```java
// Example 1 from page 3 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example1 {
  // This is a one-line comment; it extends to the end of the line.
  /* This is a delimited comment,
     extending over several lines.
  */
  int /* This delimited comment extends over part of a line */ x = 117;
}
```

```java
// Example 2 from page 3 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example2 {                                  // Class declaration
  int x;

  Example2(int x) {
    this.x = x;                                 // One-line body
  }

  int sum(int y) {                              // Multi-line body
    if (x > 0) {                                // If statement
      return x + y;                             // Single statement
    } else if (x < 0) {                         // Nested if-else, block statement
      int res = -x + y;
      return res * 117;
    } else { // x == 0                          // Terminal else, block statement
      int sum = 0;
      for (int i=0; i<10; i++) {                // For loop
        sum += (y - i) * (y - i);
      }
      return sum;
    }
  }

  static boolean checkdate(int mth, int day) {
    int length;
    switch (mth) {                              // Switch statement
    case 2:                                     // Single case
      length = 28; break;
    case 4: case 6: case 9: case 11:            // Multiple case
      length = 30; break;
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
      length = 31; break;
    default:
      return false;
    }
    return (day >= 1) && (day <= length);
  }
}
```

```java
// Example 4 from page 7 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example4 {
  public static void main(String[] args) {
    int i1 = 1000111222, i2 = 40000, i3 = -1;
    floatdouble(i1, i1);                           // L W: 1.00011123E9 1.000111222E9
    bytecharshort((byte)i2, (char)i2, (short)i2);  // C C C: 64 40000 -25536
    bytecharshort((byte)i3, (char)i3, (short)i3);  // C C C: -1 65535 -1
    intint((int)1.9, (int)-1.9);                   // C C: 1 -1
    intint((int)1.5, (int)-1.5);                   // C C: 1 -1
    intint((int)2.5, (int)-2.5);                   // C C: 2 -2
  }

  private static void floatdouble(float f, double d) {
    System.out.println(f + " " + d);
  }

  private static void bytecharshort(byte b, char c, short s) {
    System.out.println(b + " " + (int)c + " " + s);
  }

  private static void intint(int i1, int i2) {
    System.out.println(i1 + " " + i2);
  }
}
```

```java
// Example 6 from page 9 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example6 {
  public static void main(String[] args) {
    int a, b, c;
    int x = 1, y = 2, z = 3;
    int ratio = z/x;
    final double PI = 3.141592653589;
    boolean found = false;
    final int maxyz;
    if (z > y) maxyz = z; else maxyz = y;
    System.out.println(maxyz);
  }
}
```

```java
// Example 7 from page 9 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Scope {
                      // no x in scope
  void m1(int x) {    // declaration of parameter x (#1)
    x = 7;            // x #1 in scope
  }                   //
                      // no x in scope
  void m2(int v2) {   //
    x = 7;            // x #5 in scope
  }                   //
                      // no x in scope
  void m3(int v3) {   //
    x = 7;            // x #5 in scope
    int x;            // declaration of variable x (#2)
    x = 7;            // x #2 in scope
  }                   //
                      // no x in scope
  void m4(int v4) {   //
    x = 7;            // x #5 in scope
    {                 //
      int x;          // declaration of variable x (#3)
      x = 7;          // x #3 in scope
    }                 //
    x = 7;            // x #5 in scope
    {                 //
      int x;          // declaration of variable x (#4)
      x = 7;          // x #4 in scope
    }                 //
    x = 7;            // x #5 in scope
  }                   //
                      // no x in scope
  int x;              // declaration of field x (#5)
  { x = 7; }          // x #5 in scope
}
```

```java
// Example 8 from page 11 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example8 {
  public static void main(String[] args) {
    equalityAndPlus();
  }

  static void equalityAndPlus() {
    String s1 = "abc";
    String s2 = s1 + "";        // New object, but contains same text as s1
    String s3 = s1;             // Same object as s1
    String s4 = s1.toString();  // Same object as s1
    // The following statements print false, true, true, true, true:
    System.out.println("s1 and s2 identical objects: " + (s1 == s2));
    System.out.println("s1 and s3 identical objects: " + (s1 == s3));
    System.out.println("s1 and s4 identical objects: " + (s1 == s4));
    System.out.println("s1 and s2 contain same text: " + (s1.equals(s2)));
    System.out.println("s1 and s3 contain same text: " + (s1.equals(s3)));
    // These two statements print 35A and A1025 because (+) is left associative:
    System.out.println(10 + 25 + "A");  // Same as (10 + 25) + "A"
    System.out.println("A" + 10 + 25);  // Same as ("A" + 10) + 25
  }
}
```

```java
// Example 9 from page 11 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example9 {
  public static void main(String[] args) {
    String res = "";
    for (int i=0; i<args.length; i++)
      res += args[i];
    System.out.println(res);
  }
}
```

```java
// Example 10 from page 11 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example10 {
  public static void main(String[] args) {
    System.out.println(ecount(""));
    System.out.println(ecount("hjsdafj yew eqwh ge Ee"));
    System.out.println(ecount("hjsdafj yew fqwh gf Ef"));
  }

  static int ecount(String s) {
    int ecount = 0;
    for (int i=0; i<s.length(); i++)
      if (s.charAt(i) == 'e')
        ecount++;
    return ecount;
  }
}
```

```java
// Example 11 from page 11 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example11 {
  public static void main(String[] args) {
    String[] ss0 = { "", "abde", "abdf", "b" };
    System.out.println(sorted(ss0));
    String[] ss1 = { };
    System.out.println(sorted(ss1));
    String[] ss2 = { "jhkjfgsad " };
    System.out.println(sorted(ss2));
    String[] ss3 = { "Baaaa", "Abbbb" };
    System.out.println(sorted(ss3));
  }

  static boolean sorted(String[] a) {
    for (int i=1; i<a.length; i++)
      if (a[i-1].compareTo(a[i]) > 0)
        return false;
    return true;
  }
}
```

```java
// Example 12 from page 11 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example12 {
  public static void main(String[] args) {
    Point p1 = new Point(10, 20);
    Point p2 = new Point(30, 40);
    System.out.println("p1 is " + p1);        // Prints: p1 is (10, 20)
    System.out.println("p2 is " + p2);        // Prints: p2 is (30, 40)
    p2.move(7, 7);
    System.out.println("p2 is " + p2);        // Prints: p2 is (37, 47)
  }
}
```

```java
// Example 13 from page 13 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example13 {
  public static void main(String[] args) {
    String res = String.format("|%1$s|%1$7s|%1$-7s|", "Oslo");
    System.out.println(res);
  }
}
```

```java
// Example 14 from page 13 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Random;

class Example14 {
  public static void main(String[] args) {
    Random rnd = new Random();                       // Random number generator
    for (int i=0; i<3; i++) {
      for (int j=0; j<5; j++)
        System.out.format("%4d", rnd.nextInt(1000)); // Random integer 0..999
      System.out.format("%n");                       // Newline
    }
  }
}
```

```java
// Example 15 from page 15 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.GregorianCalendar;

class Example15 {
  public static void main(String[] args) {
    GregorianCalendar date = new GregorianCalendar(2004, 8, 14, 12, 9, 28);
    System.out.format("%1$tF %1$tR%n", date);
  }
}
```

```java
// Example 16 from page 15 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Date;
import java.util.Locale;

class Example16 {
  public static void main(String[] args) {
    Date now = new Date();
    System.out.format("%tc%n", now);                       // default locale
    System.out.format(Locale.US, "%tc%n", now);            // en_US locale
    System.out.format(Locale.GERMANY, "%tc%n", now);       // de_DE locale
  }
}
```

```java
// Example 17 from page 15 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Date;
import java.util.Locale;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;

class Example17 {
  public static void main(String[] args)
    throws UnsupportedEncodingException
  {
    PrintWriter wr =
      new PrintWriter(new OutputStreamWriter(System.out, "ISO-8859-1"));
    double d = 1234567.9;
    wr.format(Locale.US,      "%,.2f%n", d);        // en_US locale
    wr.format(Locale.GERMANY, "%,.2f%n", d);        // de_DE locale
    wr.format(Locale.FRANCE,  "%,.2f%n", d);        // fr_FR locale
    wr.flush();
  }
}
```

```java
// Example 18 from page 17 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example18 {
  public static void main(String[] args) {
    // Roll a die, count frequencies
    int[] freq = new int[6];                    // all initialized to 0
    for (int i=0; i<1000; i++) {
      int die = (int)(1 + 6 * Math.random());
      freq[die-1] += 1;
    }
    for (int c=1; c<=6; c++)
      System.out.println(c + " came up " + freq[c-1] + " times");

    // Create an array of the strings "A0", "A1", ..., "A19"
    String[] number = new String[20];           // all initialized to null
    for (int i=0; i<number.length; i++)
      number[i] = "A" + i;
    for (int i=0; i<number.length; i++)
      System.out.println(number[i]);
  }
}
```

```java
// Example 19 from page 17 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example19 {
  public static void main(String[] args) {
    Number[] a = new Integer[10];       // Length 10, element type Integer
    Double d = new Double(3.14);        // Type Double,  class Double
    Integer i = new Integer(117);       // Type Integer, class Integer
    Number n = i;                       // Type Number,  class Integer
    a[0] = i;                           // OK, Integer is subtype of Integer
    a[1] = n;                           // OK, Integer is subtype of Integer
    System.out.println("stored a[0] and a[1]");
    a[2] = d;                           // NO, Double not subtype of Integer
    System.out.println("should never get here");
  }
}
```

```java
// Example 20 from page 17 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example20 {
  public static void main(String[] args) {
    System.out.println("August 31 is legal: " + checkdate(8, 31));
    System.out.println("April 31 is legal:  " + checkdate(4, 31));
  }

  static int[] days = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

  static boolean checkdate(int mth, int day)
  { return (mth >= 1) && (mth <= 12) && (day >= 1) && (day <= days[mth-1]); }
}
```

```java
// Example 21 from page 17 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Random;

class Example21 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example21 <length>\n");
    else {
      System.out.println("Timing character replacement in a string:");
      Random rnd = new Random();
      int length = Integer.parseInt(args[0]);
      char[] cbuf = new char[length];
      for (int i=0; i<length; i++)
        cbuf[i] = (char)(65 + rnd.nextInt(26));
      String s = new String(cbuf);
      for (int i=0; i<10; i++) {
        Timer t = new Timer();
        String res = replaceCharChar(s, 'A', 'H');
        System.out.print(t.check() + " ");
      }
      System.out.println();
    }
  }

  static String replaceCharChar(String s, char c1, char c2) {
    char[] res = new char[s.length()];
    for (int i=0; i<s.length(); i++)
      if (s.charAt(i) == c1)
        res[i] = c2;
      else
        res[i] = s.charAt(i);
    return new String(res);
  }

  private final static class Timer {
    private long starttime;

    public Timer() { starttime = System.currentTimeMillis(); }

    double check() {
      return (System.currentTimeMillis()-starttime)/1000.0;
    }
  }
}
```

```java
// Example 22 from page 19 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example22 {
  public static void main(String[] args) {
    // Two ways to create a rectangular array of the form
    // 0.0 0.0
    // 0.0 0.0
    // 0.0 0.0

    double[][] r1 = new double[3][2];

    double[][] r2 = new double[3][];
    for (int i=0; i<3; i++)
      r2[i] = new double[2];

    // Three ways to create a lower triangular array of the form
    // 0.0
    // 0.0 0.0
    // 0.0 0.0 0.0

    double[][] t1 = new double[3][];
    for (int i=0; i<3; i++)
      t1[i] = new double[i+1];

    // A nested array initializer creating an array of the above shape
    double[][] t2 = { { 0.0 }, { 0.0, 0.0 }, { 0.0, 0.0, 0.0 } };

    // Creating an anonymous array of the above shape
    double[][] t3 = new double[][] { { 0.0 }, { 0.0, 0.0 }, { 0.0, 0.0, 0.0 } };
  }
}
```

```java
// Example 23 from page 19 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example23 {
  public static void main(String[] args) {
    String ecoliK12MG1655  =
       "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC"
      +"TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA"
      +"TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC"
      +"ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG"
      +"CCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA"
      +"GTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC"
      +"AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTGGCGATGATTG"
      +"AAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTTGCCGAACTTTT";
    codonfreq(ecoliK12MG1655);
  }

  static void codonfreq(String s) {
    // The translation from nucleotide ACGTacgt to index 0123
    int[] fromNuc = new int[128];
    for (int i=0; i<fromNuc.length; i++)
      fromNuc[i] = -1;
    fromNuc['a'] = fromNuc['A'] = 0; fromNuc['c'] = fromNuc['C'] = 1;
    fromNuc['g'] = fromNuc['G'] = 2; fromNuc['t'] = fromNuc['T'] = 3;
    // Count frequencies of codons (triples of nucleotides)
    int[][][] freq = new int[4][4][4];
    for (int i=0; i+2<s.length(); i+=3) {
      int nuc1 = fromNuc[s.charAt(i)];
      int nuc2 = fromNuc[s.charAt(i+1)];
      int nuc3 = fromNuc[s.charAt(i+2)];
      freq[nuc1][nuc2][nuc3] += 1;
    }
    // The translation from index 0123 to nucleotide ACGT
    final char[] toNuc = { 'A', 'C', 'G', 'T' };
    for (int i=0; i<4; i++)
      for (int j=0; j<4; j++) {
        for (int k=0; k<4; k++)
          System.out.print(" "+toNuc[i]+toNuc[j]+toNuc[k]
                           +": " + freq[i][j][k]);
        System.out.println();
      }
  }
}
```

```java
// Example 24 from page 21 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Point {
  int x, y;

  Point(int x, int y) { this.x = x; this.y = y; }

  Point(Point p) { this(p.x, p.y); }    // calls the above constructor

  void move(int dx, int dy) { x += dx; y += dy; }

  public String toString() { return "(" + x + ", " + y + ")"; }
}
```

```java
// Example 25 from page 21 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.ArrayList;

class SPoint {
  static ArrayList<SPoint> allpoints = new ArrayList<SPoint>();
  int x, y;

  SPoint(int x, int y) { allpoints.add(this); this.x = x; this.y = y; }
  void move(int dx, int dy) { x += dx; y += dy; }
  public String toString() { return "(" + x + ", " + y + ")"; }
  int getIndex() { return allpoints.indexOf(this); }
  static int getSize() { return allpoints.size(); }
  static SPoint getPoint(int i) { return allpoints.get(i); }
}
```

```java
// Example 26 from page 21 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class TLC {                                        // top-level class
  static int sf;
  int nf;

  static class SMC {                               // static member class
    static int ssf = sf + TLC.sf;                  // can have static members
    int snf = sf + TLC.sf;                         // cannot use non-static TLC members
  }

  class NMC {                                      // non-static member (inner) class
    int nnf1 = sf + nf;                            // can use non-static TLC members
    int nnf2 = TLC.sf + TLC.this.nf;               // cannot have static members
  }

  void nm() {                                      // non-static method in TLC
    class NLC {                                    // local (inner) class in method
      int m(int p) { return sf+nf+p; }             // can use non-static TLC members
    }                                              // cannot have static members
  }

  // According to the Java Language Specification, 2nd edition,
  // section 8.1.2, every local class is an inner class, and thus
  // static members are disallowed in local classes, even inside
  // static methods:

  static void sm() {                               // static method in TLC
    class SLC {                                    // local class
      // static int ssf = 78;                      // cannot have static members
    }                                              // cannot use non-static TLC members
  }
}

class Example26 {
  public static void main(String[] args) {
    TLC.SMC sio = new TLC.SMC();
    TLC.sf = 10;
    TLC oo = new TLC();
    oo.nf = 5;
    TLC.NMC io1 = oo.new NMC();
    System.out.println("io1.nnf1 = " + io1.nnf1);
    oo.nf = 7;
    TLC.NMC io2 = oo.new NMC();
    System.out.println("io1.nnf1 = " + io1.nnf1);
    System.out.println("io2.nnf1 = " + io2.nnf1);
  }
}
```

```java
// Example 27 from page 23 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


abstract class Vessel {
  double contents;
  abstract double capacity();
  void fill(double amount) { contents = Math.min(contents + amount, capacity()); }
}

class Tank extends Vessel {
  double length, width, height;

  Tank(double length, double width, double height)
  { this.length = length; this.width = width; this.height = height; }

  double capacity() { return length * width * height; }

  public String toString()
  { return "tank (" + length + ", " + width + ", " + height + ")"; }
}

class Cube extends Tank {
  Cube(double side) { super(side, side, side); }
  public String toString() { return "cube (" + length + ")"; }
}

class Barrel extends Vessel {
  double radius, height;
  Barrel(double radius, double height) { this.radius = radius; this.height = height; }
  double capacity() { return height * Math.PI * radius * radius; }
  public String toString() { return "barrel (" + radius + ", " + height + ")"; }
}
```

```java
// Example 28 from page 23 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example28 {
   public static void main(String[] args) {
     Vessel v1 = new Barrel(3, 10);
     Vessel v2 = new Tank(10, 20, 12);
     Vessel v3 = new Cube(4);
     Vessel[] vs = { v1, v2, v3 };
     v1.fill(90); v1.fill(10); v2.fill(100); v3.fill(80);
     double sum = 0;
     for (int i=0; i<vs.length; i++)
       sum += vs[i].capacity();
     System.out.println("Total capacity is " + sum);
     for (int i=0; i<vs.length; i++)
       System.out.println("vessel number " + i + ": " + vs[i]);
   }
}
```

```java
// Example 30 from page 25 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class B                          // one non-static field vf, one static sf
{ int vf; static int sf; B(int i) { vf = i; sf = i+1; } }

class C extends B                // two non-static fields vf, one static sf
{ int vf; static int sf; C(int i) { super(i+20); vf = i; sf = i+2; } }

class D extends C                // three non-static fields vf
{ int vf; D(int i) { super(i+40); vf = i; sf = i+4; } }
```

```java
// Example 32 from page 25 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Access {
  private static int x;

  static class SI {
    private static int y = x;       // access private x from enclosing class
  }

  static void m() {
    int z = SI.y;                   // access private y from nested class
  }
}
```

```java
// Example 33 from page 27 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example33 {
  double m(int i) { return i; }
  boolean m(boolean b) { return !b; }
  static double m(int x, double y) { return x + y + 1; }
  static double m(double x, double y) { return x + y + 3; }

  public static void main(String[] args) {
    System.out.println(m(10, 20));              // Prints 31.0
    System.out.println(m(10, 20.0));            // Prints 31.0
    System.out.println(m(10.0, 20));            // Prints 33.0
    System.out.println(m(10.0, 20.0));          // Prints 33.0
  }
}
```

```java
// Example 34 from page 27 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class C1 {
  static void m1(double d) { System.out.println("11d"); }
  void m1(int i)      { System.out.println("11i"); }
  void m2(int i)      { System.out.println("12i"); }
}

class C2 extends C1 {
  static void m1(double d) { System.out.println("21d"); }
  void m1(int i)      { System.out.println("21i"); }
  void m2(double d)   { System.out.println("22d"); }
  void m2(Integer ii) { System.out.println("22ii"); }
  void m3(int i)      { System.out.println("23i"); }
  void m4(Integer ii) { System.out.println("24ii"); }
}

class Example34 {
  public static void main(String[] args) {
    int i = 17;
    Integer ii = new Integer(i);
    double d = 17.0;
    C2 c2 = new C2();                              // Type C2, object class C2
    C1 c1 = c2;                                    // Type C1, object class C2
    c1.m1(i); c2.m1(i); c1.m1(d); c2.m1(d);        // Prints 21i 21i 11d 21d
    c1.m2(i);                                      // Prints 12i
    c2.m2(i);                                      // Prints 12i
    c2.m2(ii);                                     // Prints 22ii
    c2.m3(ii);                                     // Prints 23i,  with unboxing
    c2.m4(i);                                      // Prints 24ii, with boxing
  }
}
```

```java
// Example 35 from page 27 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example35 {
  public static void main(String[] args) {
    System.out.println(max(69, 42));                        // max(int,int)
    System.out.println(max(2, 5, 7, 11, 3));                // max(int,int[])
    System.out.println(max(2, new int[] { 5, 7, 11, 3 })); // max(int,int[])
  }

  static int max(int x, int y) {
    return x > y ? x : y;
  }

  static int max(int x1, int... xr) {
    int res = x1;
    for (int x : xr)
      res = max(res, x);
    return res;
  }
}
```

```java
// Example 36 from page 29 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Point {
  int x, y;

  Point(int x, int y) { this.x = x; this.y = y; }

  Point(Point p) { this(p.x, p.y); }    // calls the above constructor

  void move(int dx, int dy) { x += dx; y += dy; }

  public String toString() { return "(" + x + ", " + y + ")"; }
}
```

```java
// Example 38 from page 29 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example38 {
  static double[] ps = new double[6];

  static {                               // static initializer block
    double sum = 0;
    for (int i=0; i<ps.length; i++)      // fill with increasing random numbers
      ps[i] = sum += Math.random();
    for (int i=0; i<ps.length; i++)      // scale so last ps element is 1.0
      ps[i] /= sum;
  }

  static int roll() {
    double p = Math.random();
    int i = 0;
    while (p > ps[i])
      i++;
    return i+1;
  }

  public static void main(String[] args) {
    for (int i=0; i<10; i++)
      System.out.println(roll());
  }
}
```

```java
// Example 39 from page 31 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class TLC {                                      // top-level class
  static int sf;
  int nf;

  static class SMC {                             // static member class
    static int ssf = sf + TLC.sf;                // can have static members
    int snf = sf + TLC.sf;                       // cannot use non-static TLC members
  }

  class NMC {                                    // non-static member (inner) class
    int nnf1 = sf + nf;                          // can use non-static TLC members
    int nnf2 = TLC.sf + TLC.this.nf;             // cannot have static members
  }

  void nm() {                                    // non-static method in TLC
    class NLC {                                  // local (inner) class in method
      int m(int p) { return sf+nf+p; }           // can use non-static TLC members
    }                                            // cannot have static members
  }

  // According to the Java Language Specification, 2nd edition,
  // section 8.1.2, every local class is an inner class, and thus
  // static members are disallowed in local classes, even inside
  // static methods:

  static void sm() {                             // static method in TLC
    class SLC {                                  // local class
      // static int ssf = 78;                    // cannot have static members
    }                                            // cannot use non-static TLC members
  }
}

class Example39 {
  public static void main(String[] args) {
    TLC.SMC sio = new TLC.SMC();
    TLC.sf = 10;
    TLC oo = new TLC();
    oo.nf = 5;
    TLC.NMC io1 = oo.new NMC();
    System.out.println("io1.nnf1 = " + io1.nnf1);
    oo.nf = 7;
    TLC.NMC io2 = oo.new NMC();
    System.out.println("io1.nnf1 = " + io1.nnf1);
    System.out.println("io2.nnf1 = " + io2.nnf1);
  }
}
```

```java
// Example 40 from page 31 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Iterator;

class Example40 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example40 <string>\n");
    else {
      Iterator<String> seq = suffixes(args[0]);
      while (seq.hasNext())
        System.out.println(seq.next());
    }
  }

  // Create and return an iterator over all non-empty suffixes of s
  static Iterator<String> suffixes(final String s) {
    class SuffixIterator implements Iterator<String> {
      int startindex=0;
      public boolean hasNext() { return startindex < s.length(); }
      public String next() { return s.substring(startindex++); }
      public void remove() { throw new UnsupportedOperationException(); }
    }
    return new SuffixIterator();
  }
}
```

```java
// Example 41 from page 31 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Iterator;

class Example41 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example41 <string>\n");
    else {
      Iterator<String> seq = suffixes(args[0]);
      while (seq.hasNext())
        System.out.println(seq.next());
    }
  }

  // Create and return an iterator over all non-empty suffixes of s
  static Iterator<String> suffixes(final String s) {
    return
      new Iterator<String>() {
          int startindex=0;
          public boolean hasNext() { return startindex < s.length(); }
          public String next() { return s.substring(startindex++); }
          public void remove() { throw new UnsupportedOperationException(); }
        };
  }
}
```

```java
// Example 44 from page 33 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class TLC {                                     // top-level class
  static int sf;
  int nf;

  static class SMC {                            // static member class
    static int ssf = sf + TLC.sf;               // can have static members
    int snf = sf + TLC.sf;                      // cannot use non-static TLC members
  }

  class NMC {                                   // non-static member (inner) class
    int nnf1 = sf + nf;                         // can use non-static TLC members
    int nnf2 = TLC.sf + TLC.this.nf;            // cannot have static members
  }

  void nm() {                                   // non-static method in TLC
    class NLC {                                 // local (inner) class in method
      int m(int p) { return sf+nf+p; }          // can use non-static TLC members
    }                                           // cannot have static members
  }

  // According to the Java Language Specification, 2nd edition,
  // section 8.1.2, every local class is an inner class, and thus
  // static members are disallowed in local classes, even inside
  // static methods:

  static void sm() {                            // static method in TLC
    class SLC {                                 // local class
      // static int ssf = 78;                   // cannot have static members
    }                                           // cannot use non-static TLC members
  }
}

class Example44 {
  public static void main(String[] args) {
    TLC.SMC sio = new TLC.SMC();
    TLC.sf = 10;
    TLC oo = new TLC();
    oo.nf = 5;
    TLC.NMC io1 = oo.new NMC();
    System.out.println("io1.nnf1 = " + io1.nnf1);
    oo.nf = 7;
    TLC.NMC io2 = oo.new NMC();
    System.out.println("io1.nnf1 = " + io1.nnf1);
    System.out.println("io2.nnf1 = " + io2.nnf1);
  }
}
```

```java
// Example 45 from page 37 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example45 {
  public static void main(String[] args) {
    int max = 2147483647;
    int min = -2147483648;
    println(max+1);                             // Prints -2147483648
    println(min-1);                             // Prints  2147483647
    println(-min);                              // Prints -2147483648
    print(   10/3); println(   10/(-3));        // Prints  3 -3
    print((-10)/3); println((-10)/(-3));        // Prints -3  3
    print(   10%3); println(   10%(-3));        // Prints  1  1
    print((-10)%3); println((-10)%(-3));        // Prints -1 -1
  }

  static void print(int i)   { System.out.print(i + " "); }

  static void println(int i) { System.out.println(i + " "); }
}
```

```java
// Example 46 from page 37 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example46 {
  public static void main(String[] args) {
    System.out.println(leapyear(1900));
    System.out.println(leapyear(1999));
    System.out.println(leapyear(2000));
    System.out.println(leapyear(2001));
    System.out.println(leapyear(2004));
    System.out.println(leapyear(2100));
  }

  // Returns true if y is a leap year
  static boolean leapyear(int y)
  { return y % 4 == 0 && y % 100 != 0 || y % 400 == 0; }
}
```

```java
// Example 47 from page 37 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example47 {
  public static void main(String[] args) throws Exception {
    int a = 0x3;                        // Bit pattern   0011
    int b = 0x5;                        // Bit pattern   0101
    println4(a);                        // Prints        0011
    println4(b);                        // Prints        0101
    println4(~a);                       // Prints        1100
    println4(~b);                       // Prints        1010
    println4(a & b);                    // Prints        0001
    println4(a ^ b);                    // Prints        0110
    println4(a | b);                    // Prints        0111
  }

  static void println4(int n) {
    for (int i=3; i>=0; i--)
      System.out.print(n >> i & 1);
    System.out.println();
  }
}
```

```java
// Example 48 from page 39 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example48 {
  public static void main(String[] args) {
    double d;
    d = 12;                     // widening conversion from int to double

    byte b1 = 123, b2;          // narrowing int to byte
    b2 = 123 + 1;               // legal  : 123+1 is a compile-time constant
 // b2 = b1  + 1;               // illegal: (b1 + 1) has type int
    b2 = (byte)(b1 + 1);        // legal  : (byte)(b1 + 1) has type byte
 // b2 = 123 + 5;               // illegal: 123 + 5 is larger than 127

    int x = 0;
    x += 1.5;                   // add 1 to x
    d += 1.5;                   // add 1.5 to d
    System.out.println(x);
    System.out.println(d);
  }
}
```

```java
// Example 49 from page 39 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example49 {
  public static void main(String[] args) {
    Point p1 = new Point(10, 20);
    System.out.println("p1 is " + p1);        // Prints: p1 is (10, 20)
    Point p2 = p1;                            // p1 and p2 are same object
    p2.move(8, 8);
    System.out.println("p2 is " + p2);        // Prints: p2 is (18, 28)
    System.out.println("p1 is " + p1);        // Prints: p1 is (18, 28)
  }
}
```

```java
// Example 50 from page 39 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example50 {
  public static void main(String[] args) {
    double[] a = { 365.0, 24.0, 60.0, 60.0 };
    System.out.println(multiply(a));
  }

  static double multiply(double[] xs) {
    double prod = 1.0;
    for (int i=0; i<xs.length; i++)
      prod *= xs[i];                    // Equivalent to: prod = prod * xs[i]
    return prod;
  }
}
```

```java
// Example 51 from page 39 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example51 {
  public static void main(String[] args) {
    System.out.println(absolute(-12));
    System.out.println(absolute(12));
    System.out.println(absolute(0));
  }

  // Returns the absolute value of x (always non-negative)
  static double absolute(double x)
  { return (x >= 0 ? x : -x); }
}
```

```java
// Example 52 from page 39 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example52 {
  public static void main(String[] args) {
    // Instance test
    Number n1 = new Integer(17);
    Number n2 = new Double(3.14);
    // The following statements print: false, true, false, true
    System.out.println("n1   is a Double: " + (n1 instanceof Double));
    System.out.println("n2   is a Double: " + (n2 instanceof Double));
    System.out.println("null is a Double: " + (null instanceof Double));
    System.out.println("n2   is a Number: " + (n2 instanceof Number));
  }
}
```

```java
// Example 53 from page 41 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example53 {
  public static void main(String[] args) {
    C c1 = new C(100);                  // c1 has type C; object has class C
    B b1 = c1;                          // b1 has type B; object has class C
    print(C.sf,  B.sf);                 // Prints 102 121
    print(c1.sf, b1.sf);                // Prints 102 121
    print(c1.vf, b1.vf);                // Prints 100 120
    C c2 = new C(200);                  // c2 has type C; object has class C
    B b2 = c2;                          // b2 has type B; object has class C
    print(c2.sf, b2.sf);                // Prints 202 221
    print(c2.vf, b2.vf);                // Prints 200 220
    print(c1.sf, b1.sf);                // Prints 202 221
    print(c1.vf, b1.vf);                // Prints 100 120
    D d3 = new D(300);                  // d3 has type D; object has class D
    C c3 = d3;                          // c3 has type C; object has class D
    B b3 = d3;                          // b3 has type B; object has class D
    print(D.sf,  C.sf,  B.sf);          // Prints 304 304 361
    print(d3.sf, c3.sf, b3.sf);         // Prints 304 304 361
    print(d3.vf, c3.vf, b3.vf);         // Prints 300 340 360
  }

  static void print(int x, int y) { System.out.println(x+" "+y); }
  static void print(int x, int y, int z) { System.out.println(x+" "+y+" "+z); }
}
```

```java
// Example 54 from page 41 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Point {
  int x, y;

  Point(int x, int y) { this.x = x; this.y = y; }

  Point(Point p) { this(p.x, p.y); }    // calls the above constructor

  void move(int dx, int dy) { x += dx; y += dy; }

  public String toString() { return "(" + x + ", " + y + ")"; }
}
```

```java
// Example 55 from page 41 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.ArrayList;

class SPoint {
  static ArrayList<SPoint> allpoints = new ArrayList<SPoint>();
  int x, y;

  SPoint(int x, int y) { allpoints.add(this); this.x = x; this.y = y; }
  void move(int dx, int dy) { x += dx; y += dy; }
  public String toString() { return "(" + x + ", " + y + ")"; }
  int getIndex() { return allpoints.indexOf(this); }
  static int getSize() { return allpoints.size(); }
  static SPoint getPoint(int i) { return allpoints.get(i); }
}
```

```java
// Example 56 from page 43 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example56 {
  public static void main(String[] args) {
    System.out.println("Number of points created: " + SPoint.getSize());
    SPoint p = new SPoint(12, 123);
    SPoint q = new SPoint(200, 10);
    SPoint r = new SPoint(99, 12);
    SPoint s = p;
    q = null;
    System.out.println("Number of points created: " + SPoint.getSize());
    System.out.println("Number of points created: " + q.getSize());
    System.out.println("r is point number " + r.getIndex());
    for (int i=0; i<SPoint.getSize(); i++)
      System.out.println("SPoint number " + i + " is " + SPoint.getPoint(i));
  }
}
```

```java
// Example 57 from page 43 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example57 {
  public static void main(String[] args) {
    Point p = new Point(10, 20);
    int[] a = new int[5];
    int d = 8;
    System.out.println("p is " + p);          // Prints: p is (10, 20)
    System.out.println("a[3] is " + a[3]);    // Prints: a[3] is 0
    m(p, d, a);
    System.out.println("p is " + p);          // Prints: p is (18, 28)
    System.out.println("d is " + d);          // Prints: d is 8
    System.out.println("a[3] is " + a[3]);    // Prints: a[3] is 22
  }

  static void m(Point pp, int dd, int[] aa) {
    pp.move(dd, dd);
    dd = 117;
    aa[3] = 22;
  }
}
```

```java
// Example 58 from page 45 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example58 {
  double m(int i) { return i; }
  boolean m(boolean b) { return !b; }
  static double m(int x, double y) { return x + y + 1; }
  static double m(double x, double y) { return x + y + 3; }

  public static void main(String[] args) {
    System.out.println(m(10, 20));              // Prints 31.0
    System.out.println(m(10, 20.0));            // Prints 31.0
    System.out.println(m(10.0, 20));            // Prints 33.0
    System.out.println(m(10.0, 20.0));          // Prints 33.0
  }
}
```

```java
// Example 59 from page 45 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class C1 {
  static void m1(double d) { System.out.println("11d"); }
  void m1(int i)      { System.out.println("11i"); }
  void m2(int i)      { System.out.println("12i"); }
}

class C2 extends C1 {
  static void m1(double d) { System.out.println("21d"); }
  void m1(int i)      { System.out.println("21i"); }
  void m2(double d)   { System.out.println("22d"); }
  void m2(Integer ii) { System.out.println("22ii"); }
  void m3(int i)      { System.out.println("23i"); }
  void m4(Integer ii) { System.out.println("24ii"); }
}

class Example59 {
  public static void main(String[] args) {
    int i = 17;
    Integer ii = new Integer(i);
    double d = 17.0;
    C2 c2 = new C2();                               // Type C2, object class C2
    C1 c1 = c2;                                     // Type C1, object class C2
    c1.m1(i); c2.m1(i); c1.m1(d); c2.m1(d);         // Prints 21i 21i 11d 21d
    c1.m2(i);                                       // Prints 12i
    c2.m2(i);                                       // Prints 12i
    c2.m2(ii);                                      // Prints 22ii
    c2.m3(ii);                                      // Prints 23i,  with unboxing
    c2.m4(i);                                       // Prints 24ii, with boxing
  }
}
```

```java
// Example 60 from page 45 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class D1 {
  D1() { m2(); }
  void m1() { System.out.println("D1.m1 "); }
  void m2() { System.out.print("D1.m2 "); m1(); }
}

class D2 extends D1 {
  int f;
  D2() { f = 7; }
  void m1() { System.out.println("D2.m1:" + f); }
}

class Example60 {
  public static void main(String[] args) {
    System.out.println("Executing: D1 d1 = new D1()");
    D1 d1 = new D1();           // Prints D1.m2 D1.m1
    System.out.println("Executing: D1 d2 = new D2()");
    D1 d2 = new D2();           // Prints D1.m2 D2.m1:0
    System.out.println("Executing: d1.m2()");
    d1.m2();                    // Prints D1.m2 D1.m1
    System.out.println("Executing: d2.m2()");
    d2.m2();                    // Prints D1.m2 D2.m1:7
} }
```

```java
// Example 61 from page 47 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example61 {
  public static void main(String[] args) {
    int offset = 10;
    class Pair {
      public final int fst, snd;
      public Pair(int fst, int snd) { this.fst = fst; this.snd = snd; }
      public String toString() { return String.format("(%d,%d)", fst, snd); }
    }
    {
      Pair p1 = new Pair(10, 10+offset);
      System.out.println(p1);
    }
    {
      Pair p1 = new Pair(200, 300);
      System.out.println(p1);
    }
  }
}
```

```java
// Example 62 from page 47 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example62 {
  public static void main(String[] args) {
    System.out.println("Infinite loop!  Stop it by pressing ctrl-C\n\n");
    int i=0;
    while (i<10);
      i++;
  }
}
```

```java
// Example 63 from page 47 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example63 {
  public static void main(String[] args) {
    System.out.println(absolute(-12));
    System.out.println(absolute(12));
    System.out.println(absolute(0));
  }

  // Behaves the same as absolute on page~\pageref{pgm-absolute-conditional}:
  static double absolute(double x) {
    if (x >= 0)
      return x;
    else
      return -x;
  }
}
```

```java
// Example 64 from page 47 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example64 {
  public static void main(String[] args) {
    System.out.println("Thursday is " + wdayno1("Thursday"));
  }

  static int wdayno1(String wday) {
    if      (wday.equals("Monday"))    return 1;
    else if (wday.equals("Tuesday"))   return 2;
    else if (wday.equals("Wednesday")) return 3;
    else if (wday.equals("Thursday"))  return 4;
    else if (wday.equals("Friday"))    return 5;
    else if (wday.equals("Saturday"))  return 6;
    else if (wday.equals("Sunday"))    return 7;
    else return -1;                              // Here used to mean ‘not found’
  }
}
```

```java
// Example 65 from page 49 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example65 {
  public static void main(String[] args) {
    System.out.println("44 is " + findCountry(44));
  }

  static String findCountry(int prefix) {
    switch (prefix) {
    case 1:   return "North America";
    case 44:  return "Great Britain";
    case 45:  return "Denmark";
    case 299: return "Greenland";
    case 46:  return "Sweden";
    case 7:   return "Russia";
    case 972: return "Israel";
    default:  return "Unknown";
    }
  }
}
```

```java
// Example 67 from page 49 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example67 {
  private static final HashMap<String,Integer> wdayNumber = new HashMap<String,Integer>();

  static {
    int wdayno = 0;
    String[] wdays =
      { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };
    for (String wday : wdays)
      wdayNumber.put(wday, wdayno++);
  }

  public static void main(String[] args) {
    if (args.length != 2)
      System.out.println("Usage: Example67 <weekday> <hours>\n");
    else {
      double hours = Double.parseDouble(args[1]);
      Integer index = wdayNumber.get(args[0]);
      switch (index != null ? index : -1) {
      case 0:
        System.out.format("Monday: pay is %.2f%n", 10+7.42*hours);
        break;
      case 1: case 2: case 3: case 4:
        System.out.format("Workday: pay is %.2f%n", 7.42*hours);
        break;
      case 5: case 6:
        System.out.format("Weekend: pay is %.2f%n", 20+1.25*7.42*hours);
        break;
      default:
        System.out.format("Unknown weekday: %s%n", args[0]);
      }
    }
  }
}
```

```java
// Example 68 from page 51 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example68 {
  public static void main(String[] args) {
    for (int i=1; i<=4; i++) {
      for (int j=1; j<=i; j++)
        System.out.print("*");
      System.out.println();
    }
  }
}
```

```java
// Example 69 from page 51 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example69 {
  public static void main(String[] args) {
    int[] iarr = { 2, 3, 5, 7, 11 };
    int sum = 0;
    for (int i : iarr)
      sum += i;
    System.out.println(sum);
  }
}
```

```java
// Example 70 from page 51 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example70 {
  public static void main(String[] args) {
    for (int i : fromTo(13, 17))
      System.out.println(i);
  }

  public static Iterable<Integer> fromTo(final int m, final int n) {
    class FromToIterator implements Iterator<Integer> {
      private int i = m;
      public boolean hasNext() { return i <= n; }
      public Integer next() {
        if (i <= n)
          return i++;
        else
          throw new NoSuchElementException();
      }
      public void remove() { throw new UnsupportedOperationException(); }
    }
    class FromToIterable implements Iterable<Integer> {
      public Iterator<Integer> iterator() {
        return new FromToIterator();
      }
    }
    return new FromToIterable();
  }
}
```

```java
// Example 71 from page 51 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example71 {
  public static void main(String[] args) {
    Iterable<Integer> ible = fromTo(13, 17);
    for (Iterator<Integer> iter = ible.iterator(); iter.hasNext(); /* none */) {
      int i = iter.next();
      System.out.println(i);
    }
  }

  public static Iterable<Integer> fromTo(final int m, final int n) {
    class FromToIterator implements Iterator<Integer> {
      private int i = m;
      public boolean hasNext() { return i <= n; }
      public Integer next() {
        if (i <= n)
          return i++;
        else
          throw new NoSuchElementException();
      }
      public void remove() { throw new UnsupportedOperationException(); }
    }
    class FromToIterable implements Iterable<Integer> {
      public Iterator<Integer> iterator() {
        return new FromToIterator();
      }
    }
    return new FromToIterable();
  }
}
```

```java
// Example 72 from page 53 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example72 {
  public static void main(String[] args) {
    System.out.println("Thursday is " + wdayno2("Thursday"));
  }

  static int wdayno2(String wday) {
    int i=0;
    while (i < wdays.length && ! wday.equals(wdays[i]))
      i++;
    // Now i >= wdays.length or wday equal to wdays[i]
    if (i < wdays.length)
      return i+1;
    else
      return -1;                                // Here used to mean ‘not found’
  }

  static final String[] wdays =
  { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };
}
```

```java
// Example 73 from page 53 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example73 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage:  Example73 <double>\n");
    else {
      double[] arr = { 2, 3, 5, 5, 7, 11, 13 };
      int k = binarySearch(arr, Double.parseDouble(args[0]));
      if (k >= 0)
        System.out.format("Found at %d%n", k);
      else
        System.out.format("Not found, belongs at %d%n", ~k);
    }
  }

  // Array arr must be sorted in increasing order and may contain
  // duplicates.

  // If we imagine that arr[-1]==minus infinity and
  // arr[arr.length]==plus infinity, then the following invariant
  // holds at beginning and end of the loop body:
  //    arr[a-1] < x < arr[b+1]

  // The negation of the loop condition is (a>b); and in fact a==b+1
  // and b==a-1 just after the loop.  It follows that
  //    arr[b] < x < arr[a]
  // just after the loop.

  // So in case x is not found in the sorted array, we can insert x at
  // arr[a], moving up the elements arr[a...], and rest assured that
  // the resulting array remains sorted.

  // Note that when x is found, then its index i>=0 is returned.  If x
  // is not found, then the one's complement ~a of a is returned;
  // since a>=0, the one's complement is negative.  Hence we can
  // easily distinguish success (x found) from failure (x not found).

  public static int binarySearch(double[] arr, double x) {
    int a=0, b=arr.length-1;
    while (a<=b) { // Loop invariant: arr[a-1] < x < arr[b+1]
      int i = (a+b)/2;
      if (arr[i] < x)      a = i+1;
      else if (arr[i] > x) b = i-1;
      else                 return i;    // because arr[i] == x
    }
    // Now a>b, in fact a==b+1 and b==a-1, and arr[b] < x < arr[a]
    return ~a;
  }
}
```

```java
// Example 74 from page 53 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example74 {
  public static void main(String[] args) {
    Iterable<Integer> ible = fromTo(13, 17);
    Iterator<Integer> iter = ible.iterator();
    while (iter.hasNext()) {
      int i = iter.next();
      System.out.println(i);
    }
  }

  public static Iterable<Integer> fromTo(final int m, final int n) {
    class FromToIterator implements Iterator<Integer> {
      private int i = m;
      public boolean hasNext() { return i <= n; }
      public Integer next() {
        if (i <= n)
          return i++;
        else
          throw new NoSuchElementException();
      }
      public void remove() { throw new UnsupportedOperationException(); }
    }
    class FromToIterable implements Iterable<Integer> {
      public Iterator<Integer> iterator() {
        return new FromToIterator();
      }
    }
    return new FromToIterable();
  }
}
```

```java
// Example 75 from page 53 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example75 {
  public static void main(String[] args) {
    System.out.println("Counting sum of eyes until 5 or 6 comes up (10000 dice).");
    int[] wait = new int[1000];
    for (int i=0; i<10000; i++)
      wait[waitsum()]++;
    System.out.println("sum: frequency");
    for (int w=5; w<20; w++)
      System.out.println(w + ": " + wait[w]);
  }

  // Roll a die and compute sum until five or six comes up
  static int waitsum() {
    int sum = 0, eyes;
    do {
      eyes = (int)(1 + 6 * Math.random());
      sum += eyes;
    } while (eyes < 5);
    return sum;
  }
}
```

```java
// Example 76 from page 55 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example76 {
  public static void main(String[] args) {
    System.out.println("Thursday is " + wdayno3("Thursday"));
  }

  static int wdayno3(String wday) {
    for (int i=0; i < wdays.length; i++)
      if (wday.equals(wdays[i]))
        return i+1;
    return -1;                                  // Here used to mean ‘not found’
  }

  static final String[] wdays =
  { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };
}
```

```java
// Example 77 from page 55 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example77 {
  public static void main(String[] args) {
    double[] a = { 365.0, 24.0, 60.0, 60.0 };
    System.out.println(shortcutmultiply(a));
  }

  static double shortcutmultiply(double[] xs) {
    double prod = 1.0;
    for (int i=0; i<xs.length; i++) {
      prod *= xs[i];
      if (prod == 0.0)
        break;
    }
    return prod;
  }
}
```

```java
// Example 78 from page 55 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example78 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example78 <string>\n");
    else {
      String q = args[0];
      System.out.println(q + " substring of hjsdfk: " + substring1(q, "hjsdfk"));
    }
  }

  // Decide whether query is a substring of target (using continue)

  static boolean substring1(String query, String target) {
    nextposition:
      for (int j=0; j<=target.length()-query.length(); j++) {
        for (int k=0; k<query.length(); k++)
          if (target.charAt(j+k) != query.charAt(k))
            continue nextposition;
        return true;
      }
    return false;
  }
}
```

```java
// Example 79 from page 55 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example79 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example79 <string>\n");
    else {
      String q = args[0];
      System.out.println(q + " substring of hjsdfk: " + substring2(q, "hjsdfk"));
    }
  }

  // Decide whether query is a substring of target (using break)

  static boolean substring2(String query, String target) {
    for (int j=0; j<=target.length()-query.length(); j++)
      thisposition: {
        for (int k=0; k<query.length(); k++)
          if (target.charAt(j+k) != query.charAt(k))
            break thisposition;
        return true;
      }
    return false;
  }
}
```

```java
// Example 80 from page 57 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example80 {
  public static void main(String[] args) {
    try {
      System.out.println(args[0] + " is weekday number " + wdayno4(args[0]));
    } catch (WeekdayException x) {
      System.out.println("Weekday problem: " + x);
    } catch (Exception x) {
      System.out.println("Other problem: " + x);
    }
  }

  // Behaves the same as wdayno3 above, but throws Exception instead of
  // returning bogus weekday number:
  static int wdayno4(String wday) throws WeekdayException {
    for (int i=0; i < wdays.length; i++)
      if (wday.equals(wdays[i]))
        return i+1;
    throw new WeekdayException(wday);
  }

  static final String[] wdays =
  { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };
}
```

```java
// Example 81 from page 57 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example81 {
  public static void main(String[] args) {
    try {
      System.out.println(args[0] + " is weekday number " + wdayno4(args[0]));
    } catch (WeekdayException x) {
      System.out.println("Weekday problem: " + x);
    } catch (Exception x) {
      System.out.println("Other problem: " + x);
    }
  }

  // Behaves the same as wdayno3 above, but throws Exception instead of
  // returning bogus weekday number:
  static int wdayno4(String wday) throws WeekdayException {
    for (int i=0; i < wdays.length; i++)
      if (wday.equals(wdays[i]))
        return i+1;
    throw new WeekdayException(wday);
  }

  static final String[] wdays =
  { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };
}
```

```java
// Example 82 from page 57 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example82 {
  public static void main(String[] args) throws IOException {
    double[] xs = readRecord("foo");
    for (int i=0; i<xs.length; i++)
      System.out.println(xs[i]);
  }

  static double[] readRecord(String filename) throws IOException {
    Reader freader         = new FileReader(filename);
    BufferedReader breader = new BufferedReader(freader);
    double[] res = new double[3];
    try {
      res[0] = new Double(breader.readLine()).doubleValue();
      res[1] = new Double(breader.readLine()).doubleValue();
      res[2] = new Double(breader.readLine()).doubleValue();
    } finally {
      breader.close();
    }
    return res;
  }
}
```

```java
// Example 83 from page 59 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example83 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java -ea Example83 <integer>\n");
    else {
      int x = Integer.parseInt(args[0]);
      System.out.println("Integer square root of " + x + " is " + sqrt(x));
    }
  }

  // Modified for Java from C code on Paul Hsieh's square root page

  static int sqrt(int x) {  // Algorithm by Borgerding, Hsieh, Ulery
    if (x < 0)
      throw new IllegalArgumentException("sqrt: negative argument");
    int temp, y = 0, b = 0x8000, bshft = 15, v = x;;
    do {
      if (v >= (temp = (y<<1)+b << bshft--)) {
        y += b; v -= temp;
      }
    } while ((b >>= 1) > 0);
    assert (long)y * y <= x && (long)(y+1)*(y+1) > x;
    return y;
  }
}
```

```java
// Example 84 from page 59 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

class Example84 {
  public static void main(String[] args) throws IOException {
    if (args.length != 2)
      System.out.println("Usage: java -ea Example84 <textfile> <linewidth>\n");
    else {
      Iterator wordIter =
        new ReaderIterator(new BufferedReader(new FileReader(args[0])));
      int lineWidth = Integer.parseInt(args[1]);
      format(wordIter, lineWidth,
             new PrintWriter(new OutputStreamWriter(System.out)));
    }
  }

  // This method formats a sequence of words (Strings obtained from
  // the iterator wordIter) into lines of text, padding the inter-word
  // gaps with extra spaces to obtain lines of length lineWidth, and
  // thus a straight right margin.
  //
  // There are the following exceptions:
  //  * if a word is longer than lineWidth, it is put on a line by
  //    itself (producing a line longer than lineWidth)
  //  * a single word may appear on a line by itself (producing a line
  //     shorter than lineWidth) if adding the next word to the line
  //     would make it longer than lineWidth
  //  * the last line of the output is not padded with extra spaces.
  //
  // The algorithm for padding with extra spaces ensures that the
  // spaces are evenly distributed over inter-word gaps, using modulo
  // arithmetics.  An assert statement asserts that the resulting
  // output line has the correct length unless the line contains only
  // a single word or is the last line of the output.

  public static void format(Iterator wordIter, int lineWidth, PrintWriter pw) {
    lineWidth = Math.max(0, lineWidth);
    WordList curLine = new WordList();
    while (wordIter.hasNext()) {
      while (wordIter.hasNext() && curLine.length() < lineWidth) {
        String word = (String)wordIter.next();
        if (!word.equals(""))
          curLine.addLast(word);
      }
      int wordCount = curLine.size();
      if (wordCount > 0) {
        int extraSpaces = lineWidth - curLine.length();
        if (wordCount > 1 && extraSpaces < 0) { // last word goes on next line
          int lastWordLength = curLine.getLast().length();
          extraSpaces += 1 + lastWordLength;
          wordCount -= 1;
        } else if (!wordIter.hasNext())         // last line, do not pad
          extraSpaces = 0;
        // Pad holes with evenly distributed extra spaces
        int holes = wordCount - 1;
        int spaces = holes/2;
        StringBuilder sbuf = new StringBuilder();
        sbuf.append(curLine.removeFirst());
        for (int i=1; i<wordCount; i++) {
          spaces += extraSpaces;
          appendSpaces(sbuf, 1 + spaces / holes);
          spaces %= holes;
          sbuf.append(curLine.removeFirst());
        }
        String res = sbuf.toString();
        assert res.length()==lineWidth || wordCount==1 || !wordIter.hasNext();
        pw.println(res);
      }
    }
    pw.flush();
  }

  static void appendSpaces(StringBuilder sbuf, int count) {
    for (int i=0; i<count; i++)
```

```
      sbuf.append(' ');
  }
}

// A word list with a fast length method, and invariant assertions

class WordList {
  private LinkedList<String> strings = new LinkedList<String>();
  // Invariant: length equals word lengths plus inter-word spaces
  private int length = -1;

  public int length() { return length; }

  public int size() { return strings.size(); }

  public void addLast(String s) {
    strings.addLast(s);
    length += 1 + s.length();
    assert length == computeLength() + strings.size() - 1;
  }

  public String removeFirst() {
    String res = strings.removeFirst();
    length -= 1 + res.length();
    assert length == computeLength() + strings.size() - 1;
    return res;
  }

  public String getLast() {
    return strings.getLast();
  }

  private int computeLength() {  // For checking the invariant only
    int sum = 0;
    for (String s : strings)
      sum += s.length();
    return sum;
  }
}

// A String-producing Iterator, created from a Reader using StreamTokenizer

class ReaderIterator implements Iterator {
  private StreamTokenizer stok; // equals null if at end or error

  public ReaderIterator(Reader rd) {
    stok = new StreamTokenizer(rd);
    stok.resetSyntax();
    stok.whitespaceChars(' ', ' '); stok.whitespaceChars('\n', '\n');
    stok.whitespaceChars('\r', '\r'); stok.whitespaceChars('\t', '\t');
    stok.wordChars('!', '~');
    try {
      stok.nextToken();
    } catch (IOException e) {
      stok = null;
    }
  }

  public boolean hasNext() {
    return stok != null && stok.ttype != StreamTokenizer.TT_EOF;
  }

  public Object next() {
    String res = null;
    if (hasNext())
      try {
        res = stok.sval;
        stok.nextToken();
      } catch (IOException e) {
        stok = null;
      }
    return res;
  }

  public void remove() {
    throw new UnsupportedOperationException();
  }
}
```

```
// Example 85 from page 61 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.awt.Color;
import java.awt.Graphics;

interface Colored { Color getColor(); }
interface Drawable { void draw(Graphics g); }
interface ColoredDrawable extends Colored, Drawable {}
```

```
// Example 86 from page 61 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.awt.Color;
import java.awt.Graphics;

class ColoredPoint extends Point implements Colored {
  Color c;
  ColoredPoint(int x, int y, Color c) { super(x, y); this.c = c; }
  public Color getColor() { return c; }
}

class ColoredDrawablePoint extends ColoredPoint implements ColoredDrawable {
  ColoredDrawablePoint(int x, int y, Color c) { super(x, y, c); }
  public void draw(Graphics g) { g.fillRect(x, y, 1, 1); }
}

class ColoredRectangle implements ColoredDrawable {
  int x1, x2, y1, y2;   // (x1, y1) upper left, (x2, y2) lower right corner
  Color c;

  ColoredRectangle(int x1, int y1, int x2, int y2, Color c)
  { this.x1 = x1; this.y1 = y1; this.x2 = x2; this.y2 = y2; this.c = c; }

  public Color getColor() { return c; }

  public void draw(Graphics g) { g.drawRect(x1, y1, x2-x1, y2-y1); }
}
```

```java
// Example 87 from page 61 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Graphics;

class Example87 {
  static void printcolors(Colored[] cs) {
    for (int i=0; i<cs.length; i++)
      System.out.println(cs[i].getColor().toString());
  }

  static void draw(Graphics g, ColoredDrawable[] cs) {
    for (int i=0; i<cs.length; i++) {
      g.setColor(cs[i].getColor());
      cs[i].draw(g);
    }
  }

  public static void main(String[] args) {
    final ColoredDrawable[] cs =
          { new ColoredDrawablePoint(3, 4, Color.red),
            new ColoredRectangle(50, 100, 60, 110, Color.green) };
    printcolors(cs);
    Frame f = new Frame();
    // Anonymous local inner class, subclass of java.awt.Canvas
    f.add(new Canvas() {
        static final long serialVersionUID = 50L;
        public void paint(Graphics g) {
          draw(g, cs);
        }
        public Dimension getPreferredSize() { return new Dimension(400, 300); }
      });
    f.pack(); f.setVisible(true);
  }
}
```

```java
// Example 88 from page 63 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example88 {
  public static void main(String[] args) {
    if (args.length != 3)
      System.out.println("Usage: Example88 yyyy mm dd\n");
    else {
      try {
        MyDate d = new MyDate(Integer.parseInt(args[0]),
                              Month.toMonth(Integer.parseInt(args[1])),
                              Integer.parseInt(args[2]));
        System.out.println(d + " is " + d.weekday() + " in week " + d.weekNumber());
      } catch (Exception e) {
        System.out.println(e);
      }
    }
  }
}

enum Day {
  Mon, Tue, Wed, Thu, Fri, Sat, Sun;

  private final static Day[] day = values();        // Cache the array

  public static Day toDay(int n) {
    return day[n];
  }

  public int toInt() {
    return ordinal();
  }
}

enum Month {
  Jan(31), Feb(28), Mar(31), Apr(30), May(31), Jun(30),
  Jul(31), Aug(31), Sep(30), Oct(31), Nov(30), Dec(31);

  private final int days;

  private Month(int days) {
    this.days = days;
  }

  private final static Month[] month = values();    // Cache the array

  public int days(int year) {
    return this == Feb && MyDate.leapYear(year) ? 29 : days;
  }

  public static Month toMonth(int n) {
    return month[n-1];
  }

  public int toInt() {
    return ordinal()+1;
  }

  public Month succ() {
    return toMonth(toInt()+1);
  }
}

class MyDate {
  final int yy /* 0-9999 */, dd /* 1-31 */;
  final Month mm;

  public MyDate(int yy, Month mm, int dd) throws Exception {
    if (ok(yy, mm, dd)) {
      this.yy = yy; this.mm = mm; this.dd = dd;
    } else
      throw new Exception("Illegal date ("+yy+","+mm+","+dd+")");
  }

  public static boolean leapYear(int y) {
    return y % 4 == 0 && y % 100 != 0 || y % 400 == 0;
```

```
  }

  public boolean leapYear() {
    return leapYear(yy);
  }

  // An alternative to method days(int) in enum type Month:

  public static int monthDays(int y, Month m) {
    switch (m) {
    case Apr: case Jun: case Sep: case Nov:
      return 30;
    case Feb:
      return leapYear(y) ? 29 : 28;
    default:
      return 31;
    }
  }

  public int monthDays() {
    return mm.days(yy);
  }

  public static int yearDays(int y) {
    return leapYear(y) ? 366 : 365;
  }

  public int yearDays() {
    return yearDays(yy);
  }

  public static boolean ok(int y, Month m, int d) {
    return 1 <= d && d <= m.days(y);
  }

  // ISO week numbers: the week is from Monday to Sunday.  Week 1 is
  // the first week having a Thursday.

  public static int weekNumber(int y, Month m, int d) {
    int yday = dayInYear(y, m, d);
    int wday = weekday(y, m, d).toInt();
    int week = (yday - wday + 10)/7;
    if (week == 0)
      return (yday + yearDays(y-1) - wday + 10)/7;
    else
      return week;
  }

  public int weekNumber() {
    return weekNumber(yy, mm, dd);
  }

  // Translated from Emacs's calendar.el:
  // Reingold: Number of the day within the year:

  public static int dayInYear(int y, Month m, int d) {
    int monthno = m.toInt() - 1;
    int monthadjust =
      monthno > 1 ? (27 + 4 * monthno) / 10 - (leapYear(y) ? 1 : 0) : 0;
    return d - 1 + 31 * monthno - monthadjust;
  }

  public int dayInYear() {
    return dayInYear(yy, mm, dd);
  }

  // Reingold: Find the number of days elapsed from the (imagined)
  // Gregorian date Sunday, December 31, 1 BC to the given date.

  public static int toDaynumber(int y, Month m, int d) {
    int prioryears = y - 1;
    return
      dayInYear(y, m, d)
      + 1 + 365 * prioryears
      + prioryears / 4 - prioryears / 100 + prioryears / 400;
  }
```

```
  public int toDaynumber() {
    return toDaynumber(yy, mm, dd);
  }

  // Reingold et al: from absolute day number to year, month, date:

  public static MyDate fromDaynumber(int n) {
    try {
      int d0 = n - 1;
      int n400 = d0 / 146097;
      int d1 = d0 % 146097;
      int n100 = d1 / 36524;
      int d2 = d1 % 36524;
      int n4 = d2 / 1461;
      int d3 = d2 % 1461;
      int n1 = d3 / 365;
      int d = 1 + d3 % 365;
      int y = 400 * n400 + 100 * n100 + n4 * 4 + n1 + 1;
      if (n100 == 4 || n1 == 4) {
        return new MyDate(y-1, Month.Dec, 31);
      } else {
        Month m = Month.Jan;
        int mdays;
        while ((mdays = m.days(y)) < d) {
          d -= mdays;
          m = m.succ();
        }
        return new MyDate(y, m, d);
      }
    } catch (Exception e) {
      throw new RuntimeException("MyDate.fromDaynumber: impossible");
    }
  }

  // Day of the week: 0=Mon, 1=Tue, ..., 6=Sun

  public static Day weekday(int y, Month m, int d) {
    return Day.toDay((toDaynumber(y, m, d)+6) % 7);
  }

  public Day weekday() {
    return weekday(yy, mm, dd);
  }

  public String toString() { // ISO format such as 2004-07-11
    return String.format("%4d-%02d-%02d", yy, mm.toInt(), dd);
  }
}
```

```java
// Example 89 from page 65 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class WeekdayException extends Exception {
  static final long serialVersionUID = 50L;
  public WeekdayException(String wday) {
    super("Illegal weekday: " + wday);
  }
}
```

```java
// Example 90 from page 65 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


// To exercise all paths through the try-catch-finally statement in
// method m, run this program with each of these arguments:
// 101 102 103 201 202 203 301 302 303 411 412 413 421 422 423 431 432 433
// like this:
//    java Example90 101
//    java Example90 102
//    etc

class Example90 {
  public static void main(String[] args) throws Exception {
    if (args.length != 1)
      System.out.println("Usage: java Example90 <integer>\n");
    else
      System.out.println(m(Integer.parseInt(args[0])));
  }

  static String m(int a) throws Exception {
    try {
      System.out.print("try ... ");
      if (a/100 == 2) return "returned from try";
      if (a/100 == 3) throw new Exception("thrown by try");
      if (a/100 == 4) throw new RuntimeException("thrown by try");
    } catch (RuntimeException x) {
      System.out.print("catch ... ");
      if (a/10%10 == 2) return "returned from catch";
      if (a/10%10 == 3) throw new Exception("thrown by catch");
    } finally {
      System.out.println("finally");
      if (a%10 == 2) return "returned from finally";
      if (a%10 == 3) throw new Exception("thrown by finally");
    }
    return "terminated normally with " + a;
  }
}
```

```java
// Example 91 from page 67 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Incrementer extends Thread {
  public int i;
  public void run() {
    for (;;) {                                    // Forever
      i++;                                        // increment i
      yield();
    }
} }

class Example91 {
  public static void main(String[] args) throws IOException {
    Incrementer u = new Incrementer();
    u.start();
    System.out.println("Repeatedly press Enter to get the current value of i:");
    for (;;) {
      System.in.read();                           // Wait for keyboard input
      System.out.println(u.i);
} } }
```

```java
// Example 92 from page 69 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Printer extends Thread {
  static Object mutex = new Object();
  public void run() {
    for (;;) {
      synchronized (mutex) {
        System.out.print("-");
        Util.pause(100,300);
        System.out.print("/");
      }
      Util.pause(200);
} } }

class Example92 {
  public static void main(String[] args)
  { new Printer().start(); new Printer().start(); }
}

// Pseudo-random numbers and sleeping threads

class Util {
  public static void pause(int length)
  { try { Thread.sleep(length); } catch (InterruptedException x) {} }

  public static void pause(int a, int b) { pause(random(a, b)); }

  public static int random(int a, int b)
  { return (int)(a + (b-a+1) * Math.random()); }
}
```

```java
// Example 93 from page 69 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Bank {
  private int account1 = 10, account2 = 20;
  synchronized public void transfer(int amount) {
    int new1 = account1 - amount;
    Util.pause(10);
    account1 = new1; account2 = account2 + amount;
    System.out.println("Sum is " + (account1+account2));
} }

class Clerk extends Thread {
  private Bank bank;
  public Clerk(Bank bank) { this.bank = bank; }
  public void run() {
    for (;;) {                                    // Forever
      bank.transfer(Util.random(-10, 10));        //   transfer money
      Util.pause(200, 300);                       //   then take a break
} } }

class Example93 {
  public static void main(String[] args) {
    Bank bank = new Bank();
    new Clerk(bank).start(); new Clerk(bank).start();
} }
```

```java
// Example 94 from page 71 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Buffer {
  private int contents;
  private boolean empty = true;
  public int get() {
    synchronized (this) {
      while (empty)
        try { this.wait(); } catch (InterruptedException x) {};
      empty = true;
      this.notify();
      return contents;
  } }
  public void put(int v) {
    synchronized (this) {
      while (!empty)
        try { this.wait(); } catch (InterruptedException x) {};
      empty = false;
      contents = v;
      this.notify();
  } }
}

class Example94 {
  public static void main(String[] args) {
    final Buffer buf = new Buffer();

    class Producer extends Thread {
      public void run() {
        for (int i=1; true; i++) {
          buf.put(i);
          Util.pause(10, 100);
    } } }

    class Consumer extends Thread {
      public void run() {
        for (;;)
          System.out.println("Consumed " + buf.get());

    } }

    new Producer().start();
    new Consumer().start(); new Consumer().start();
} }
```

```java
// Example 95 from page 71 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.awt.Canvas;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Graphics;

class AnimatedCanvas extends Canvas implements Runnable {
  static final long serialVersionUID = 50L;
  private int scale = 50, step = 5;

  AnimatedCanvas() { Thread u = new Thread(this); u.start(); }

  public void run() {                           // from Runnable
    for (;;) { // forever sleep and repaint
      try { Thread.sleep(100); } catch (InterruptedException e) { }
      if (scale <= 0 || scale >= 100)
        step = -step;
      scale += step;
      repaint();
    }
  }

  public void paint(Graphics g) {               // from Canvas
    Dimension size = getSize();
    g.fillRect(scale*(size.width-20)/100,                    0, 20, 20);
    g.fillRect((100-scale)*(size.width-20)/100, size.height-20, 20, 20);
  }

  public Dimension getPreferredSize() { return new Dimension(400, 100); }
  public Dimension getMinimumSize() { return getPreferredSize(); }
}

class Example95 {
  public static void main(String[] args) {
    Canvas anim1 = new AnimatedCanvas();
    Canvas anim2 = new AnimatedCanvas();
    Frame f = new Frame("Animation with threads");
    f.add(anim1, "North"); f.add(anim2, "South");
    f.pack(); f.setVisible(true);
  }
}
```

```java
// Example 97 from page 75 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example97 {
  public static void main(String[] args) {
    for (int i=0; i<=100; i++)
      System.out.println(i + "! = " + fact(i));
  }

  static double fact(int n) {
    double res = 0.0;
    for (int i=1; i<=n; i++)
      res += Math.log(i);
    return Math.exp(res);
  }
}
```

```java
// Example 98 from page 75 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example98 {
  public static void main(String[] args) {
    printGaussians(100);
  }

  // From http://www.taygeta.com/random/gaussian.html 2001-09-21:
  // The most basic form of the transformation looks like:
  //               y1 = sqrt( - 2 ln(x1) ) cos( 2 pi x2 )
  //               y2 = sqrt( - 2 ln(x1) ) sin( 2 pi x2 )

  // We start with two independent random numbers, x1 and x2, which
  // come from a uniform distribution (in the range from 0 to 1). Then
  // apply the above transformations to get two new independent random
  // numbers which have a Gaussian distribution with zero mean and a
  // standard deviation of one.

  static void printGaussians(int n) {
    for (int i=0; i<n; i+=2) {
      double x1 = Math.random(), x2 = Math.random();
      print(Math.sqrt(-2 * Math.log(x1)) * Math.cos(2 * Math.PI * x2));
      print(Math.sqrt(-2 * Math.log(x1)) * Math.sin(2 * Math.PI * x2));
    }
  }

  static void print(double d) {
    System.out.println(d);
  }
}
```

```java
// Example 99 from page 75 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example99 {
  public static void main(String[] args) {
    mathtest();
  }

  static void mathtest() {
    print("Illegal arguments, NaN results:");
    print(Math.sqrt(-1));                // NaN
    print(Math.log(-1));                 // NaN
    print(Math.pow(-1, 2.5));            // NaN
    print(Math.acos(1.1));               // NaN
    print("Infinite results:");
    print(Math.log(0));                  // -Infinity
    print(Math.pow(0, -1));              // Infinity
    print(Math.exp(1000.0));             // Infinity (overflow)
    print("Infinite arguments:");
    double infinity = Double.POSITIVE_INFINITY;
    print(Math.sqrt(infinity));          // Infinity
    print(Math.log(infinity));           // Infinity
    print(Math.exp(-infinity));          // 0.0
    print("NaN arguments and special cases:");
    double nan = Math.log(-1);
    print(Math.sqrt(nan));               // NaN
    print(Math.pow(nan, 0));             // 1.0 (special case)
    print(Math.pow(0, 0));               // 1.0 (special case)
    print(Math.round(nan));              // 0   (special case)
    print(Math.round(1E50));             // 9223372036854775807 (Long.MAX_VALUE)
    // For all (x, y) except (0.0, 0.0):
    // sign(cos(atan2(y, x))) == sign(x) && sign(sin(atan2(y, x))) == sign(y)
    for (double x=-100; x<=100; x+=0.125) {
      for (double y=-100; y<=100; y+=0.125) {
        double r = Math.atan2(y, x);
        if (!(sign(Math.cos(r))==sign(x) && sign(Math.sin(r))==sign(y)))
          print("x = " + x + "; y = " + y);
      }
    }
  }

  static int sign(double x) {
    final double tolerance = 1E-14;
    if (x < -tolerance)
      return -1;
    else if (x > +tolerance)
      return +1;
    else
      return 0;
  }

  static void print(String d) {
    System.out.println(d);
  }

  static void print(double d) {
    System.out.println(d);
  }

  static void print(long d) {
    System.out.println(d);
  }
}
```

```java
// Example 100 from page 77 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Example100 {
  public static void main(String[] args) {
    StringBuilder res = new StringBuilder();
    for (int i=0; i<args.length; i++)
      res.append(args[i]);
    System.out.println(res.toString());
  }
}
```

```java
// Example 101 from page 77 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Random;

class Example101 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example101 <length>\n");
    else {
      System.out.println("Timing character replacement in a string:");
      Random rnd = new Random();
      int length = Integer.parseInt(args[0]);
      char[] cbuf = new char[length];
      for (int i=0; i<length; i++)
        cbuf[i] = (char)(65 + rnd.nextInt(26));
      String s = new String(cbuf);
      for (int i=0; i<10; i++) {
        Timer t = new Timer();
        String res = replaceCharString(s, 'A', "HA");
        System.out.print(t.check() + " ");
      }
      System.out.println();
    }
  }

  static String replaceCharString(String s, char c1, String s2) {
    StringBuilder res = new StringBuilder();
    for (int i=0; i<s.length(); i++)
      if (s.charAt(i) == c1)
        res.append(s2);
      else
        res.append(s.charAt(i));
    return res.toString();
  }

  private final static class Timer {
    private long starttime;

    public Timer() { starttime = System.currentTimeMillis(); }

    double check() {
      return (System.currentTimeMillis()-starttime)/1000.0;
    }
  }
}
```

```java
// Example 102 from page 77 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Random;

class Example102 {
  public static void main(String[] args) {
    if (args.length != 1)
      System.out.println("Usage: java Example102 <length>\n");
    else {
      System.out.println("Timing character replacement, abusing a string buffer:");
      Random rnd = new Random();
      int length = Integer.parseInt(args[0]);
      char[] cbuf = new char[length];
      for (int i=0; i<length; i++)
        cbuf[i] = (char)(65 + rnd.nextInt(26));
      String s = new String(cbuf);
      for (int i=0; i<10; i++) {
        StringBuilder sb = new StringBuilder(s);
        Timer t = new Timer();
        replaceCharString(sb, 'A', "HA");
        System.out.print(t.check() + " ");
      }
      System.out.println();
    }
  }

  // In-place replacement in a StringBuilder; very inefficient and strange

  static void replaceCharString(StringBuilder sb, char c1, String s2) {
    int i = 0;                                   // Inefficient
    while (i < sb.length()) {                    // Inefficient
      if (sb.charAt(i) == c1) {                  // Inefficient
        sb.replace(i, i+1, s2);                  // Inefficient
        i += s2.length();                        // Inefficient
      } else                                     // Inefficient
        i += 1;                                  // Inefficient
    }                                            // Inefficient
  }

  private final static class Timer {
    private long starttime;

    public Timer() { starttime = System.currentTimeMillis(); }

    double check() {
      return (System.currentTimeMillis()-starttime)/1000.0;
    }
  }
}
```

```java
// Example 103 from page 77 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.text.DecimalFormat;

class Example103 {
  public static void main(String[] args) {
    System.out.println("Printing temperature table");
    DecimalFormat ff = new DecimalFormat("#0"), cf = new DecimalFormat("0.0");
    System.out.println("Fahrenheit   Celsius");
    for (double f=100; f<=400; f+=10) {
      double c = 5 * (f - 32) / 9;
      System.out.println(padLeft(ff.format(f), 10) + padLeft(cf.format(c), 10));
    }
  }

  static String padLeft(String s, int width) {
    StringBuilder res = new StringBuilder();
    for (int i=width-s.length(); i>0; i--)
      res.append(' ');
    return res.append(s).toString();
  }
}
```

```java
// Example 104 from page 79 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example104 {
  public static void main(String[] args) {
    ArrayList cool = new ArrayList();
    cool.add(new Person("Kristen"));
    cool.add(new Person("Bjarne"));
    cool.add(new Exception("Larry"));   // Wrong, but no compiletime check
    cool.add(new Person("Anders"));
    Person p = (Person)(cool.get(2));   // Compiles OK, but fails at runtime
  }

  private static class Person {
    private static int counter = 0;
    private final String name;
    private final int serialNumber;

    public Person(String name) {
      this.name = name;
      this.serialNumber = counter++;
    }
  }
}
```

```java
// Example 105 from page 79 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example105 {
  public static void main(String[] args) {
    ArrayList<Person> cool = new ArrayList<Person>();
    cool.add(new Person("Kristen"));
    cool.add(new Person("Bjarne"));
    //   cool.add(new Exception("Larry"));   // Wrong, detected at compile-time
    cool.add(new Person("Anders"));
    Person p = cool.get(2);                  // No explicit cast or check needed
  }

  private static class Person {
    private static int counter = 0;
    private final String name;
    private final int serialNumber;

    public Person(String name) {
      this.name = name;
      this.serialNumber = counter++;
    }
  }
}
```

```java
// Example 106 from page 79 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.Date;

class Pair<T,U> {
  public final T fst;
  public final U snd;
  public Pair(T fst, U snd) {
    this.fst = fst;
    this.snd = snd;
  }
}

class Example106 {
  public static void main(String[] args) {
    Pair<String,Integer> p1 = new Pair<String,Integer>("Niels", 1947);
    Pair<Double,Integer> p2 = new Pair<Double,Integer>(2.718, 1);
    Pair<Date,String> p3 = new Pair<Date,String>(new Date(), "now");
    System.out.println("(" + p1.fst + ", " + p1.snd + ")");
    System.out.println("(" + p2.fst + ", " + p2.snd + ")");
    System.out.println("(" + p3.fst + ", " + p3.snd + ")");
  }
}
```

```java
// Example 107 from page 81 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

/*

Since it is illegal to create an array by new T[SIZE] in class Log<T>,
we must pass a suitable array to the constructor.

Similarly, method getAll() could not return an array of type T[],
because it is not allowed to create such an array.  Also, it cannot
use res.toArray(log) to create an array with the right element type,
as that would overwrite the log array...

*/

class Log<T> {
  private final int size;
  private static int instanceCount = 0;
  private int count = 0;
  private T[] log;
  public Log(T[] log) {
    this.log = log;
    this.size = log.length;
    instanceCount++;
  }
  public static int getInstanceCount() { return instanceCount; }
  public void add(T msg) { log[count++ % size] = msg; }
  public int getCount() { return count; }
  public T getLast() {
    // Return the last log entry, or null if nothing logged yet
    return count==0 ? null : log[(count-1)%size];
  }
  public void setLast(T value) {
    // Update the last log entry, or create one if nothing logged yet
    if (count==0)
      log[count++] = value;
    else
      log[(count-1)%size] = value;
  }
  public ArrayList<T> getAll() {
    ArrayList<T> res = new ArrayList<T>();
    int stop = Math.min(count, size);
    for (int i=0; i<stop; i++)
      res.add(log[(count-stop+i) % size]);
    return res;
  }
}

class Example107 {
  public static void main(String[] args) {
    Log<String> log1 = new Log<String>(new String[5]);
    log1.add("Reboot");
    log1.add("Coffee");
    Log<Date> log2 = new Log<Date>(new Date[5]);
    log2.add(new Date());                                          // now
    log2.add(new Date(new Date().getTime() + 60*60*1000));      // now + 1 hour
    ArrayList<Date> dts = log2.getAll();
    // Printing both logs:
    for (String s : log1.getAll())
      System.out.print(s + "    ");
    System.out.println();
    for (Date dt : dts)
      System.out.print(dt + "   ");
    System.out.println();
  }
}
```

```java
// Example 108 from page 81 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

/*
   This generic interface represents a function from A to R; in C# one
   would use a generic delegate instead.

   In MyLinkedList method equals(Object), the cast to MyList<T> is
   unchecked and may fail at runtime.
*/

interface Mapper<A,R> {
  R call(A x);
}

// A generic linked-list class with elements of type T

interface MyList<T> extends Iterable<T> {
  int getCount();                       // Number of elements
  T get(int i);                         // Get element at index i
  void set(int i, T item);              // Set element at index i
  void add(T item);                     // Add element at end
  void insert(int i, T item);           // Insert element at index i
  void removeAt(int i);                 // Remove element at index i
  <U> MyList<U> map(Mapper<T,U> f);     // Map f over all elements
}

class MyLinkedList<T> implements MyList<T> {
  protected int size;               // Number of elements in the list
  protected Node<T> first, last;    // Invariant: first==null iff last==null

  protected static class Node<U> {
    public Node<U> prev, next;
    public U item;

    public Node(U item) {
      this.item = item;
    }

    public Node(U item, Node<U> prev, Node<U> next) {
      this.item = item; this.prev = prev; this.next = next;
    }
  }

  public MyLinkedList() {
    first = last = null;
    size = 0;
  }

  public MyLinkedList(T... arr) {
    this();
    for (T x : arr)
      add(x);
  }

  public int getCount() {
    return size;
  }

  public T get(int i) {
    return getItem(i).item;
  }

  public void set(int i, T item) {
    getItem(i).item = item;
  }

  private Node<T> getItem(int n) {
    if (n < 0 || n >= size)
      throw new IndexOutOfBoundsException();
    else if (n < size/2) {              // Closer to front
      Node<T> node = first;
      for (int i=0; i<n; i++)
        node = node.next;
      return node;
    } else {                            // Closer to end
      Node<T> node = last;
      for (int i=size-1; i>n; i--)
        node = node.prev;
      return node;
    }
  }

  public void add(T item) {
    insert(size, item);
  }

  public void insert(int i, T item) {
    if (i == 0) {
      if (first == null) // and thus last == null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, null, first);
        first.prev = tmp;
        first = tmp;
      }
      size++;
    } else if (i == size) {
      if (last == null) // and thus first = null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, last, null);
        last.next = tmp;
        last = tmp;
      }
      size++;
    } else {
      Node<T> node = getItem(i);
      // assert node.prev != null;
      Node<T> newnode = new Node<T>(item, node.prev, node);
      node.prev.next = newnode;
      node.prev = newnode;
      size++;
    }
  }

  public void removeAt(int i) {
    Node<T> node = getItem(i);
    if (node.prev == null)
      first = node.next;
    else
      node.prev.next = node.next;
    if (node.next == null)
      last = node.prev;
    else
      node.next.prev = node.prev;
    size--;
  }

  public boolean equals(Object that) {
    return equals((MyList<T>)that);             // Unchecked cast
  }

  public boolean equals(MyList<T> that) {
    if (that != null && this.size == that.getCount()) {
      Node<T> thisnode = this.first;
      Iterator<T> thatiter = that.iterator();
      while (thisnode != null) {
        if (!thatiter.hasNext())
          throw new RuntimeException("Impossible: MyLinkedList<T>.equals");
        // assert next() will succeed (because of the above size test)
        if (!thisnode.item.equals(thatiter.next()))
          return false;
        thisnode = thisnode.next;
      }
      // assert !hasNext(); // because of the size test
      return true;
    } else
      return false;
  }
```

```java
  public <U> MyList<U> map(Mapper<T,U> f) {
    MyLinkedList<U> res = new MyLinkedList<U>();
    for (T x : this)
      res.add(f.call(x));
    return res;
  }

  public Iterator<T> iterator() {
    return new MyLinkedListIterator();
  }

  private class MyLinkedListIterator implements Iterator<T> {
    Node<T> next;                  // Node holding current element, or null

    public MyLinkedListIterator() {
      next = first;
    }

    public T next() {
      if (next != null) {
        T res = next.item;
        next = next.next;
        return res;
      } else
        throw new NoSuchElementException();
    }

    public boolean hasNext() {
      return next != null;
    }

    public void remove() {
      throw new UnsupportedOperationException();
    }
  }
}

class SortedList<T extends Comparable<T>> extends MyLinkedList<T> {
  // Sorted insertion
  public void insert(T x) {
    Node<T> node = first;
    while (node != null && x.compareTo(node.item) > 0)
      node = node.next;
    if (node == null)           // x > all elements; insert at end
      add(x);
    else {                      // x <= node.item; insert before node
      Node<T> newnode = new Node<T>(x);
      if (node.prev == null)    // insert as first element
        first = newnode;
      else
        node.prev.next = newnode;
      newnode.next = node;
      newnode.prev = node.prev;
      node.prev = newnode;
    }
  }
}

interface Printable {
  void print(PrintWriter fs);
}

class PrintableMyLinkedList<T extends Printable>
  extends MyLinkedList<T> implements Printable
{
  public void print(PrintWriter fs) {
    boolean firstElement = true;
    for (T x : this) {
      x.print(fs);
      if (firstElement)
        firstElement = false;
      else
        fs.print(", ");
    }
  }
}
```

```java
class MyString implements Comparable<MyString> {
  public final String s;
  public MyString(String s) {
    if (s == null)
      throw new RuntimeException("null string");
    else
      this.s = s;
  }
  public int compareTo(MyString that) {
    return s.compareTo(that.s);
  }
}

class Example108 {
  public static void main(String[] args) {
    MyLinkedList<Double> dLst
      = new MyLinkedList<Double>(7.0, 9.0, 13.0, 0.0);
    for (double d : dLst)
      System.out.print(d + " ");
    System.out.println();
    MyList<Integer> iLst =
      dLst.map(new Mapper<Double, Integer>() {
        public Integer call(Double d) {
          return d < 0 ? -1 : d > 0 ? +1 : 0;
        }
      });
    for (int i : iLst)
      System.out.print(i + " ");
    System.out.println();
    MyList<String> sLst =
      dLst.map(new Mapper<Double, String>() {
        public String call(Double d) {
          return "s" + d;
        }
      });
    for (String s : sLst)
      System.out.print(s + " ");
    System.out.println();
    // Testing SortedList<MyString>
    SortedList<MyString> sortedLst = new SortedList<MyString>();
    sortedLst.insert(new MyString("New York"));
    sortedLst.insert(new MyString("Rome"));
    sortedLst.insert(new MyString("Dublin"));
    sortedLst.insert(new MyString("Riyadh"));
    sortedLst.insert(new MyString("Tokyo"));
    for (MyString s : sortedLst)
      System.out.print(s.s + "   ");
    System.out.println();
  }
}
```

```java
// Example 109 from page 83 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

/*
   This generic interface represents a function from A to R; in C# one
   would use a generic delegate instead.

   In MyLinkedList method equals(Object), the cast to MyList<T> is
   unchecked and may fail at runtime.
*/

interface Mapper<A,R> {
  R call(A x);
}

// A generic linked-list class with elements of type T

interface MyList<T> extends Iterable<T> {
  int getCount();                         // Number of elements
  T get(int i);                           // Get element at index i
  void set(int i, T item);                // Set element at index i
  void add(T item);                       // Add element at end
  void insert(int i, T item);             // Insert element at index i
  void removeAt(int i);                   // Remove element at index i
  <U> MyList<U> map(Mapper<T,U> f);       // Map f over all elements
}

class MyLinkedList<T> implements MyList<T> {
  protected int size;               // Number of elements in the list
  protected Node<T> first, last;    // Invariant: first==null iff last==null

  protected static class Node<U> {
    public Node<U> prev, next;
    public U item;

    public Node(U item) {
      this.item = item;
    }

    public Node(U item, Node<U> prev, Node<U> next) {
      this.item = item; this.prev = prev; this.next = next;
    }
  }

  public MyLinkedList() {
    first = last = null;
    size = 0;
  }

  public MyLinkedList(T... arr) {
    this();
    for (T x : arr)
      add(x);
  }

  public int getCount() {
    return size;
  }

  public T get(int i) {
    return getItem(i).item;
  }

  public void set(int i, T item) {
    getItem(i).item = item;
  }

  private Node<T> getItem(int n) {
    if (n < 0 || n >= size)
      throw new IndexOutOfBoundsException();
    else if (n < size/2) {              // Closer to front
      Node<T> node = first;
      for (int i=0; i<n; i++)
        node = node.next;
      return node;
    } else {                            // Closer to end
      Node<T> node = last;
      for (int i=size-1; i>n; i--)
        node = node.prev;
      return node;
    }
  }

  public void add(T item) {
    insert(size, item);
  }

  public void insert(int i, T item) {
    if (i == 0) {
      if (first == null) // and thus last == null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, null, first);
        first.prev = tmp;
        first = tmp;
      }
      size++;
    } else if (i == size) {
      if (last == null) // and thus first = null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, last, null);
        last.next = tmp;
        last = tmp;
      }
      size++;
    } else {
      Node<T> node = getItem(i);
      // assert node.prev != null;
      Node<T> newnode = new Node<T>(item, node.prev, node);
      node.prev.next = newnode;
      node.prev = newnode;
      size++;
    }
  }

  public void removeAt(int i) {
    Node<T> node = getItem(i);
    if (node.prev == null)
      first = node.next;
    else
      node.prev.next = node.next;
    if (node.next == null)
      last = node.prev;
    else
      node.next.prev = node.prev;
    size--;
  }

  public boolean equals(Object that) {
    return equals((MyList<T>)that);             // Unchecked cast
  }

  public boolean equals(MyList<T> that) {
    if (that != null && this.size == that.getCount()) {
      Node<T> thisnode = this.first;
      Iterator<T> thatiter = that.iterator();
      while (thisnode != null) {
        if (!thatiter.hasNext())
          throw new RuntimeException("Impossible: MyLinkedList<T>.equals");
        // assert next() will succeed (because of the above size test)
        if (!thisnode.item.equals(thatiter.next()))
          return false;
        thisnode = thisnode.next;
      }
      // assert !hasNext(); // because of the size test
      return true;
    } else
      return false;
  }
```

```
  public <U> MyList<U> map(Mapper<T,U> f) {
    MyLinkedList<U> res = new MyLinkedList<U>();
    for (T x : this)
      res.add(f.call(x));
    return res;
  }

  public Iterator<T> iterator() {
    return new MyLinkedListIterator();
  }

  private class MyLinkedListIterator implements Iterator<T> {
    Node<T> next;                  // Node holding current element, or null

    public MyLinkedListIterator() {
      next = first;
    }

    public T next() {
      if (next != null) {
        T res = next.item;
        next = next.next;
        return res;
      } else
        throw new NoSuchElementException();
    }

    public boolean hasNext() {
      return next != null;
    }

    public void remove() {
      throw new UnsupportedOperationException();
    }
  }
}

class SortedList<T extends Comparable<T>> extends MyLinkedList<T> {
  // Sorted insertion
  public void insert(T x) {
    Node<T> node = first;
    while (node != null && x.compareTo(node.item) > 0)
      node = node.next;
    if (node == null)           // x > all elements; insert at end
      add(x);
    else {                      // x <= node.item; insert before node
      Node<T> newnode = new Node<T>(x);
      if (node.prev == null)    // insert as first element
        first = newnode;
      else
        node.prev.next = newnode;
      newnode.next = node;
      newnode.prev = node.prev;
      node.prev = newnode;
    }
  }
}

interface Printable {
  void print(PrintWriter fs);
}

class PrintableMyLinkedList<T extends Printable>
  extends MyLinkedList<T> implements Printable
{
  public void print(PrintWriter fs) {
    boolean firstElement = true;
    for (T x : this) {
      x.print(fs);
      if (firstElement)
        firstElement = false;
      else
        fs.print(", ");
    }
  }
}
```

```
class MyString implements Comparable<MyString> {
  public final String s;
  public MyString(String s) {
    if (s == null)
      throw new RuntimeException("null string");
    else
      this.s = s;
  }
  public int compareTo(MyString that) {
    return s.compareTo(that.s);
  }
}

class Example109 {
  public static void main(String[] args) {
    MyLinkedList<Double> dLst
      = new MyLinkedList<Double>(7.0, 9.0, 13.0, 0.0);
    for (double d : dLst)
      System.out.print(d + " ");
    System.out.println();
    MyList<Integer> iLst =
      dLst.map(new Mapper<Double, Integer>() {
        public Integer call(Double d) {
          return d < 0 ? -1 : d > 0 ? +1 : 0;
        }
      });
    for (int i : iLst)
      System.out.print(i + " ");
    System.out.println();
    MyList<String> sLst =
      dLst.map(new Mapper<Double, String>() {
        public String call(Double d) {
          return "s" + d;
        }
      });
    for (String s : sLst)
      System.out.print(s + " ");
    System.out.println();
    // Testing SortedList<MyString>
    SortedList<MyString> sortedLst = new SortedList<MyString>();
    sortedLst.insert(new MyString("New York"));
    sortedLst.insert(new MyString("Rome"));
    sortedLst.insert(new MyString("Dublin"));
    sortedLst.insert(new MyString("Riyadh"));
    sortedLst.insert(new MyString("Tokyo"));
    for (MyString s : sortedLst)
      System.out.print(s.s + "   ");
    System.out.println();
  }
}
```

```java
// Example 110 from page 83 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

// A constraint may involve type parameters
// A type may have multiple constraints

class ComparablePair<T extends Comparable<T>, U extends Comparable<U>>
  implements Comparable<ComparablePair<T,U>>
{
  public final T fst;
  public final U snd;

  public ComparablePair(T fst, U snd) {
    this.fst = fst; this.snd = snd;
  }

  // Lexicographic ordering
  public int compareTo(ComparablePair<T,U> that) {
    int firstCmp = this.fst.compareTo(that.fst);
    return firstCmp != 0 ? firstCmp : this.snd.compareTo(that.snd);
  }

  public String toString() {
    return "(" + fst + ", " + snd + ")";
  }
}

// Sorting soccer world champions by country and year

class Example110 {
  public static void main(String[] args) {
    ArrayList<ComparablePair<String,Integer>> lst
      = new ArrayList<ComparablePair<String,Integer>>();
    lst.add(new ComparablePair<String,Integer>("Brazil", 2002));
    lst.add(new ComparablePair<String,Integer>("Italy", 1982));
    lst.add(new ComparablePair<String,Integer>("Argentina", 1978 ));
    lst.add(new ComparablePair<String,Integer>("Argentina", 1986 ));
    lst.add(new ComparablePair<String,Integer>("Germany", 1990));
    lst.add(new ComparablePair<String,Integer>("Brazil", 1994));
    lst.add(new ComparablePair<String,Integer>("France", 1998));
    Collections.sort(lst);
    for (ComparablePair<String,Integer> pair : lst)
      System.out.println(pair);
  }
}
```

```java
// Example 111 from page 83 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Hold<T> {
  private T contents;

  public void set(Object x) {
    contents = (T)x;                    // Unchecked cast
  }

  public T get() {
    return contents;
  }
}

class Example111 {
  public static void main(String[] args) {
    Hold<Integer> h = new Hold<Integer>();
    h.set("foo");                       // Succeeds at run-time
    System.out.println("Succesfully executed  h.set(\"foo\")");
    h.get();                            // Succeeds at run-time
    System.out.println("Succesfully executed  h.get()");
    // String s = h.get();              // Illegal, rejected by compiler
    Integer i = h.get();                // Legal, but fails at run-time
    System.out.println("Succesfully executed  Integer i = h.get()");
  }
}
```

```
// Example 112 from page 85 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

/*
   This generic interface represents a function from A to R; in C# one
   would use a generic delegate instead.

   In MyLinkedList method equals(Object), the cast to MyList<T> is
   unchecked and may fail at runtime.
*/

interface Mapper<A,R> {
  R call(A x);
}

// A generic linked-list class with elements of type T

interface MyList<T> extends Iterable<T> {
  int getCount();                        // Number of elements
  T get(int i);                          // Get element at index i
  void set(int i, T item);               // Set element at index i
  void add(T item);                      // Add element at end
  void insert(int i, T item);            // Insert element at index i
  void removeAt(int i);                  // Remove element at index i
  <U> MyList<U> map(Mapper<T,U> f);      // Map f over all elements
}

class MyLinkedList<T> implements MyList<T> {
  protected int size;                // Number of elements in the list
  protected Node<T> first, last;     // Invariant: first==null iff last==null

  protected static class Node<U> {
    public Node<U> prev, next;
    public U item;

    public Node(U item) {
      this.item = item;
    }

    public Node(U item, Node<U> prev, Node<U> next) {
      this.item = item; this.prev = prev; this.next = next;
    }
  }

  public MyLinkedList() {
    first = last = null;
    size = 0;
  }

  public MyLinkedList(T... arr) {
    this();
    for (T x : arr)
      add(x);
  }

  public int getCount() {
    return size;
  }

  public T get(int i) {
    return getItem(i).item;
  }

  public void set(int i, T item) {
    getItem(i).item = item;
  }

  private Node<T> getItem(int n) {
    if (n < 0 || n >= size)
      throw new IndexOutOfBoundsException();
    else if (n < size/2) {               // Closer to front
      Node<T> node = first;
      for (int i=0; i<n; i++)
        node = node.next;
```

```
      return node;
    } else {                             // Closer to end
      Node<T> node = last;
      for (int i=size-1; i>n; i--)
        node = node.prev;
      return node;
    }
  }

  public void add(T item) {
    insert(size, item);
  }

  public void insert(int i, T item) {
    if (i == 0) {
      if (first == null) // and thus last == null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, null, first);
        first.prev = tmp;
        first = tmp;
      }
      size++;
    } else if (i == size) {
      if (last == null) // and thus first = null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, last, null);
        last.next = tmp;
        last = tmp;
      }
      size++;
    } else {
      Node<T> node = getItem(i);
      // assert node.prev != null;
      Node<T> newnode = new Node<T>(item, node.prev, node);
      node.prev.next = newnode;
      node.prev = newnode;
      size++;
    }
  }

  public void removeAt(int i) {
    Node<T> node = getItem(i);
    if (node.prev == null)
      first = node.next;
    else
      node.prev.next = node.next;
    if (node.next == null)
      last = node.prev;
    else
      node.next.prev = node.prev;
    size--;
  }

  public boolean equals(Object that) {
    return equals((MyList<T>)that);             // Unchecked cast
  }

  public boolean equals(MyList<T> that) {
    if (that != null && this.size == that.getCount()) {
      Node<T> thisnode = this.first;
      Iterator<T> thatiter = that.iterator();
      while (thisnode != null) {
        if (!thatiter.hasNext())
          throw new RuntimeException("Impossible: MyLinkedList<T>.equals");
        // assert next() will succeed (because of the above size test)
        if (!thisnode.item.equals(thatiter.next()))
          return false;
        thisnode = thisnode.next;
      }
      // assert !hasNext(); // because of the size test
      return true;
    } else
      return false;
  }
```

```java
  public <U> MyList<U> map(Mapper<T,U> f) {
    MyLinkedList<U> res = new MyLinkedList<U>();
    for (T x : this)
      res.add(f.call(x));
    return res;
  }

  public Iterator<T> iterator() {
    return new MyLinkedListIterator();
  }

  private class MyLinkedListIterator implements Iterator<T> {
    Node<T> next;                  // Node holding current element, or null

    public MyLinkedListIterator() {
      next = first;
    }

    public T next() {
      if (next != null) {
        T res = next.item;
        next = next.next;
        return res;
      } else
        throw new NoSuchElementException();
    }

    public boolean hasNext() {
      return next != null;
    }

    public void remove() {
      throw new UnsupportedOperationException();
    }
  }
}

class SortedList<T extends Comparable<T>> extends MyLinkedList<T> {
  // Sorted insertion
  public void insert(T x) {
    Node<T> node = first;
    while (node != null && x.compareTo(node.item) > 0)
      node = node.next;
    if (node == null)           // x > all elements; insert at end
      add(x);
    else {                      // x <= node.item; insert before node
      Node<T> newnode = new Node<T>(x);
      if (node.prev == null)    // insert as first element
        first = newnode;
      else
        node.prev.next = newnode;
      newnode.next = node;
      newnode.prev = node.prev;
      node.prev = newnode;
    }
  }
}

interface Printable {
  void print(PrintWriter fs);
}

class PrintableMyLinkedList<T extends Printable>
  extends MyLinkedList<T> implements Printable
{
  public void print(PrintWriter fs) {
    boolean firstElement = true;
    for (T x : this) {
      x.print(fs);
      if (firstElement)
        firstElement = false;
      else
        fs.print(", ");
    }
  }
}
```

```java
class MyString implements Comparable<MyString> {
  public final String s;
  public MyString(String s) {
    if (s == null)
      throw new RuntimeException("null string");
    else
      this.s = s;
  }
  public int compareTo(MyString that) {
    return s.compareTo(that.s);
  }
}

class Example112 {
  public static void main(String[] args) {
    MyLinkedList<Double> dLst
      = new MyLinkedList<Double>(7.0, 9.0, 13.0, 0.0);
    for (double d : dLst)
      System.out.print(d + " ");
    System.out.println();
    MyList<Integer> iLst =
      dLst.map(new Mapper<Double, Integer>() {
        public Integer call(Double d) {
          return d < 0 ? -1 : d > 0 ? +1 : 0;
        }
      });
    for (int i : iLst)
      System.out.print(i + " ");
    System.out.println();
    MyList<String> sLst =
      dLst.map(new Mapper<Double, String>() {
        public String call(Double d) {
          return "s" + d;
        }
      });
    for (String s : sLst)
      System.out.print(s + " ");
    System.out.println();
    // Testing SortedList<MyString>
    SortedList<MyString> sortedLst = new SortedList<MyString>();
    sortedLst.insert(new MyString("New York"));
    sortedLst.insert(new MyString("Rome"));
    sortedLst.insert(new MyString("Dublin"));
    sortedLst.insert(new MyString("Riyadh"));
    sortedLst.insert(new MyString("Tokyo"));
    for (MyString s : sortedLst)
      System.out.print(s.s + "   ");
    System.out.println();
  }
}
```

```java
// Example 113 from page 85 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

/*
   This generic interface represents a function from A to R; in C# one
   would use a generic delegate instead.

   In MyLinkedList method equals(Object), the cast to MyList<T> is
   unchecked and may fail at runtime.
*/

interface Mapper<A,R> {
  R call(A x);
}

// A generic linked-list class with elements of type T

interface MyList<T> extends Iterable<T> {
  int getCount();                         // Number of elements
  T get(int i);                           // Get element at index i
  void set(int i, T item);                // Set element at index i
  void add(T item);                       // Add element at end
  void insert(int i, T item);             // Insert element at index i
  void removeAt(int i);                   // Remove element at index i
  <U> MyList<U> map(Mapper<T,U> f);       // Map f over all elements
}

class MyLinkedList<T> implements MyList<T> {
  protected int size;               // Number of elements in the list
  protected Node<T> first, last;    // Invariant: first==null iff last==null

  protected static class Node<U> {
    public Node<U> prev, next;
    public U item;

    public Node(U item) {
      this.item = item;
    }

    public Node(U item, Node<U> prev, Node<U> next) {
      this.item = item; this.prev = prev; this.next = next;
    }
  }

  public MyLinkedList() {
    first = last = null;
    size = 0;
  }

  public MyLinkedList(T... arr) {
    this();
    for (T x : arr)
      add(x);
  }

  public int getCount() {
    return size;
  }

  public T get(int i) {
    return getItem(i).item;
  }

  public void set(int i, T item) {
    getItem(i).item = item;
  }

  private Node<T> getItem(int n) {
    if (n < 0 || n >= size)
      throw new IndexOutOfBoundsException();
    else if (n < size/2) {                // Closer to front
      Node<T> node = first;
      for (int i=0; i<n; i++)
        node = node.next;
      return node;
    } else {                              // Closer to end
      Node<T> node = last;
      for (int i=size-1; i>n; i--)
        node = node.prev;
      return node;
    }
  }

  public void add(T item) {
    insert(size, item);
  }

  public void insert(int i, T item) {
    if (i == 0) {
      if (first == null) // and thus last == null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, null, first);
        first.prev = tmp;
        first = tmp;
      }
      size++;
    } else if (i == size) {
      if (last == null) // and thus first = null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, last, null);
        last.next = tmp;
        last = tmp;
      }
      size++;
    } else {
      Node<T> node = getItem(i);
      // assert node.prev != null;
      Node<T> newnode = new Node<T>(item, node.prev, node);
      node.prev.next = newnode;
      node.prev = newnode;
      size++;
    }
  }

  public void removeAt(int i) {
    Node<T> node = getItem(i);
    if (node.prev == null)
      first = node.next;
    else
      node.prev.next = node.next;
    if (node.next == null)
      last = node.prev;
    else
      node.next.prev = node.prev;
    size--;
  }

  public boolean equals(Object that) {
    return equals((MyList<T>)that);             // Unchecked cast
  }

  public boolean equals(MyList<T> that) {
    if (that != null && this.size == that.getCount()) {
      Node<T> thisnode = this.first;
      Iterator<T> thatiter = that.iterator();
      while (thisnode != null) {
        if (!thatiter.hasNext())
          throw new RuntimeException("Impossible: MyLinkedList<T>.equals");
        // assert next() will succeed (because of the above size test)
        if (!thisnode.item.equals(thatiter.next()))
          return false;
        thisnode = thisnode.next;
      }
      // assert !hasNext(); // because of the size test
      return true;
    } else
      return false;
  }
```

```java
  public <U> MyList<U> map(Mapper<T,U> f) {
    MyLinkedList<U> res = new MyLinkedList<U>();
    for (T x : this)
      res.add(f.call(x));
    return res;
  }

  public Iterator<T> iterator() {
    return new MyLinkedListIterator();
  }

  private class MyLinkedListIterator implements Iterator<T> {
    Node<T> next;                  // Node holding current element, or null

    public MyLinkedListIterator() {
      next = first;
    }

    public T next() {
      if (next != null) {
        T res = next.item;
        next = next.next;
        return res;
      } else
        throw new NoSuchElementException();
    }

    public boolean hasNext() {
      return next != null;
    }

    public void remove() {
      throw new UnsupportedOperationException();
    }
  }
}

class SortedList<T extends Comparable<T>> extends MyLinkedList<T> {
  // Sorted insertion
  public void insert(T x) {
    Node<T> node = first;
    while (node != null && x.compareTo(node.item) > 0)
      node = node.next;
    if (node == null)           // x > all elements; insert at end
      add(x);
    else {                      // x <= node.item; insert before node
      Node<T> newnode = new Node<T>(x);
      if (node.prev == null)    // insert as first element
        first = newnode;
      else
        node.prev.next = newnode;
      newnode.next = node;
      newnode.prev = node.prev;
      node.prev = newnode;
    }
  }
}

interface Printable {
  void print(PrintWriter fs);
}

class PrintableMyLinkedList<T extends Printable>
  extends MyLinkedList<T> implements Printable
{
  public void print(PrintWriter fs) {
    boolean firstElement = true;
    for (T x : this) {
      x.print(fs);
      if (firstElement)
        firstElement = false;
      else
        fs.print(", ");
    }
  }
}
```

```java
class MyString implements Comparable<MyString> {
  public final String s;
  public MyString(String s) {
    if (s == null)
      throw new RuntimeException("null string");
    else
      this.s = s;
  }
  public int compareTo(MyString that) {
    return s.compareTo(that.s);
  }
}

class Example113 {
  public static void main(String[] args) {
    MyLinkedList<Double> dLst
      = new MyLinkedList<Double>(7.0, 9.0, 13.0, 0.0);
    for (double d : dLst)
      System.out.print(d + " ");
    System.out.println();
    MyList<Integer> iLst =
      dLst.map(new Mapper<Double, Integer>() {
        public Integer call(Double d) {
          return d < 0 ? -1 : d > 0 ? +1 : 0;
        }
      });
    for (int i : iLst)
      System.out.print(i + " ");
    System.out.println();
    MyList<String> sLst =
      dLst.map(new Mapper<Double, String>() {
        public String call(Double d) {
          return "s" + d;
        }
      });
    for (String s : sLst)
      System.out.print(s + " ");
    System.out.println();
    // Testing SortedList<MyString>
    SortedList<MyString> sortedLst = new SortedList<MyString>();
    sortedLst.insert(new MyString("New York"));
    sortedLst.insert(new MyString("Rome"));
    sortedLst.insert(new MyString("Dublin"));
    sortedLst.insert(new MyString("Riyadh"));
    sortedLst.insert(new MyString("Tokyo"));
    for (MyString s : sortedLst)
      System.out.print(s.s + "   ");
    System.out.println();
  }
}
```

```java
// Example 114 from page 85 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.awt.Color;

class Example114 {
  public static void main(String[] args) {
    LabelPoint<String> p1 = new LabelPoint<String>(5, 117, "home"),
      p2 = new LabelPoint<String>(2, 3, "work");
    LabelPoint<Double> p3 = new LabelPoint<Double>(10, 100, 3.1415);
    ColorLabelPoint<String,Integer> p4 =
      new ColorLabelPoint<String,Integer>(20, 30, "foo", 0x0000FF);
    ColorLabelPoint<String,Color> p5 =
      new ColorLabelPoint<String,Color>(40, 50, "bar", Color.BLUE);
    Movable[] movables = { p1, p2, p3, p4, p5 };
    LabelPoint<String>[] stringpoints;                       // Legal
    // stringpoints = new LabelPoint<String>[3];             // Illegal
  }
}

interface Movable {
  void move(int dx, int dy);
}

class LabelPoint<L> implements Movable {
  protected int x, y;
  private L lab;

  public LabelPoint(int x, int y, L lab) {
    this.x = x; this.y = y; this.lab = lab;
  }

  public void move(int dx, int dy) {
    x += dx; y += dy;
  }

  public L getLab() {
    return lab;
  }
}

class ColorLabelPoint<L, C> extends LabelPoint<L> {
  private C c;

  public ColorLabelPoint(int x, int y, L lab, C c) {
    super(x, y, lab);
    this.c = c;
  }
}
```

```java
// Example 115 from page 87 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

// Generic quicksort in functional style (the most efficient one)

class Example115 {
  public static void main(String[] args) {
    Integer[] ia = { 5, 7, 3, 9, 12, 45, 4, 8 };
    qsort(ia, new IntegerComparator(), 0, ia.length-1);
    Example115.<Integer>qsort(ia, new IntegerComparator(), 0, ia.length-1);
    Example115.qsort(ia, new IntegerComparator(), 0, ia.length-1);
    for (int i : ia)
      System.out.print(i + "   ");
    System.out.println();
    String[] sa = { "New York", "Rome", "Dublin", "Riyadh", "Tokyo" };
    Example115.qsort(sa, new StringReverseComparator(), 0, sa.length-1);
    for (String s : sa)
      System.out.print(s + "   ");
    System.out.println();
  }

  // Generic functional-style quicksort: sorts arr[a..b]

  private static <T> void qsort(T[] arr, Comparator<T> cmp, int a, int b) {
    if (a < b) {
      int i = a, j = b;
      T x = arr[(i+j) / 2];
      do {
        while (cmp.compare(arr[i], x) < 0) i++;
        while (cmp.compare(x, arr[j]) < 0) j--;
        if (i <= j) {
          T tmp = arr[i]; arr[i] = arr[j]; arr[j] = tmp;
          i++; j--;
        }
      } while (i <= j);
      qsort(arr, cmp, a, j);
      qsort(arr, cmp, i, b);
    }
  }
}

// Comparators for Integer and String

class IntegerComparator implements Comparator<Integer> {
  public int compare(Integer v1, Integer v2) {
    return v1 < v2 ? -1 : v1 > v2 ? +1 : 0;
  }
}

class StringReverseComparator implements Comparator<String> {
  public int compare(String v1, String v2) {
    return v2.compareTo(v1);
  }
}
```

```java
// Example 116 from page 87 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


// Generic quicksort in object-oriented style

class Example116 {
  public static void main(String[] args) {
    String[] sa = { "New York", "Rome", "Dublin", "Riyadh", "Tokyo" };
    qsort(sa, 0, sa.length-1);
    for (String s : sa)
      System.out.print(s + "   ");
    System.out.println();
  }

  // Generic object-oriented style quicksort: sorts arr[a..b]

  private static <T extends Comparable<T>> void qsort(T[] arr, int a, int b) {
    if (a < b) {
      int i = a, j = b;
      T x = arr[(i+j) / 2];
      do {
        while (arr[i].compareTo(x) < 0) i++;
        while (x.compareTo(arr[j]) < 0) j--;
        if (i <= j) {
          T tmp = arr[i]; arr[i] = arr[j]; arr[j] = tmp;
          i++; j--;
        }
      } while (i <= j);
      qsort(arr, a, j);
      qsort(arr, i, b);
    }
  }
}
```

```java
// Example 117 from page 87 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

/*
   This generic interface represents a function from A to R; in C# one
   would use a generic delegate instead.

   In MyLinkedList method equals(Object), the cast to MyList<T> is
   unchecked and may fail at runtime.
*/

interface Mapper<A,R> {
  R call(A x);
}

// A generic linked-list class with elements of type T

interface MyList<T> extends Iterable<T> {
  int getCount();                         // Number of elements
  T get(int i);                           // Get element at index i
  void set(int i, T item);                // Set element at index i
  void add(T item);                       // Add element at end
  void insert(int i, T item);             // Insert element at index i
  void removeAt(int i);                   // Remove element at index i
  <U> MyList<U> map(Mapper<T,U> f);       // Map f over all elements
}

class MyLinkedList<T> implements MyList<T> {
  protected int size;               // Number of elements in the list
  protected Node<T> first, last;    // Invariant: first==null iff last==null

  protected static class Node<U> {
    public Node<U> prev, next;
    public U item;

    public Node(U item) {
      this.item = item;
    }

    public Node(U item, Node<U> prev, Node<U> next) {
      this.item = item; this.prev = prev; this.next = next;
    }
  }

  public MyLinkedList() {
    first = last = null;
    size = 0;
  }

  public MyLinkedList(T... arr) {
    this();
    for (T x : arr)
      add(x);
  }

  public int getCount() {
    return size;
  }

  public T get(int i) {
    return getItem(i).item;
  }

  public void set(int i, T item) {
    getItem(i).item = item;
  }

  private Node<T> getItem(int n) {
    if (n < 0 || n >= size)
      throw new IndexOutOfBoundsException();
    else if (n < size/2) {                // Closer to front
      Node<T> node = first;
      for (int i=0; i<n; i++)
        node = node.next;
```

```java
      return node;
    } else {                              // Closer to end
      Node<T> node = last;
      for (int i=size-1; i>n; i--)
        node = node.prev;
      return node;
    }
  }

  public void add(T item) {
    insert(size, item);
  }

  public void insert(int i, T item) {
    if (i == 0) {
      if (first == null) // and thus last == null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, null, first);
        first.prev = tmp;
        first = tmp;
      }
      size++;
    } else if (i == size) {
      if (last == null) // and thus first = null
        first = last = new Node<T>(item);
      else {
        Node<T> tmp = new Node<T>(item, last, null);
        last.next = tmp;
        last = tmp;
      }
      size++;
    } else {
      Node<T> node = getItem(i);
      // assert node.prev != null;
      Node<T> newnode = new Node<T>(item, node.prev, node);
      node.prev.next = newnode;
      node.prev = newnode;
      size++;
    }
  }

  public void removeAt(int i) {
    Node<T> node = getItem(i);
    if (node.prev == null)
      first = node.next;
    else
      node.prev.next = node.next;
    if (node.next == null)
      last = node.prev;
    else
      node.next.prev = node.prev;
    size--;
  }

  public boolean equals(Object that) {
    return equals((MyList<T>)that);             // Unchecked cast
  }

  public boolean equals(MyList<T> that) {
    if (that != null && this.size == that.getCount()) {
      Node<T> thisnode = this.first;
      Iterator<T> thatiter = that.iterator();
      while (thisnode != null) {
        if (!thatiter.hasNext())
          throw new RuntimeException("Impossible: MyLinkedList<T>.equals");
        // assert next() will succeed (because of the above size test)
        if (!thisnode.item.equals(thatiter.next()))
          return false;
        thisnode = thisnode.next;
      }
      // assert !hasNext(); // because of the size test
      return true;
    } else
      return false;
  }
```

```java
  public <U> MyList<U> map(Mapper<T,U> f) {
    MyLinkedList<U> res = new MyLinkedList<U>();
    for (T x : this)
      res.add(f.call(x));
    return res;
  }

  public Iterator<T> iterator() {
    return new MyLinkedListIterator();
  }

  private class MyLinkedListIterator implements Iterator<T> {
    Node<T> next;                   // Node holding current element, or null

    public MyLinkedListIterator() {
      next = first;
    }

    public T next() {
      if (next != null) {
        T res = next.item;
        next = next.next;
        return res;
      } else
        throw new NoSuchElementException();
    }

    public boolean hasNext() {
      return next != null;
    }

    public void remove() {
      throw new UnsupportedOperationException();
    }
  }
}

class SortedList<T extends Comparable<T>> extends MyLinkedList<T> {
  // Sorted insertion
  public void insert(T x) {
    Node<T> node = first;
    while (node != null && x.compareTo(node.item) > 0)
      node = node.next;
    if (node == null)           // x > all elements; insert at end
      add(x);
    else {                      // x <= node.item; insert before node
      Node<T> newnode = new Node<T>(x);
      if (node.prev == null)    // insert as first element
        first = newnode;
      else
        node.prev.next = newnode;
      newnode.next = node;
      newnode.prev = node.prev;
      node.prev = newnode;
    }
  }
}

interface Printable {
  void print(PrintWriter fs);
}

class PrintableMyLinkedList<T extends Printable>
  extends MyLinkedList<T> implements Printable
{
  public void print(PrintWriter fs) {
    boolean firstElement = true;
    for (T x : this) {
      x.print(fs);
      if (firstElement)
        firstElement = false;
      else
        fs.print(", ");
    }
  }
}
```

```java
class MyString implements Comparable<MyString> {
  public final String s;
  public MyString(String s) {
    if (s == null)
      throw new RuntimeException("null string");
    else
      this.s = s;
  }
  public int compareTo(MyString that) {
    return s.compareTo(that.s);
  }
}

class Example117 {
  public static void main(String[] args) {
    MyLinkedList<Double> dLst
      = new MyLinkedList<Double>(7.0, 9.0, 13.0, 0.0);
    for (double d : dLst)
      System.out.print(d + " ");
    System.out.println();
    MyList<Integer> iLst =
      dLst.map(new Mapper<Double, Integer>() {
        public Integer call(Double d) {
          return d < 0 ? -1 : d > 0 ? +1 : 0;
        }
      });
    for (int i : iLst)
      System.out.print(i + " ");
    System.out.println();
    MyList<String> sLst =
      dLst.map(new Mapper<Double, String>() {
        public String call(Double d) {
          return "s" + d;
        }
      });
    for (String s : sLst)
      System.out.print(s + " ");
    System.out.println();
    // Testing SortedList<MyString>
    SortedList<MyString> sortedLst = new SortedList<MyString>();
    sortedLst.insert(new MyString("New York"));
    sortedLst.insert(new MyString("Rome"));
    sortedLst.insert(new MyString("Dublin"));
    sortedLst.insert(new MyString("Riyadh"));
    sortedLst.insert(new MyString("Tokyo"));
    for (MyString s : sortedLst)
      System.out.print(s.s + "   ");
    System.out.println();
  }
}
```

```java
// Example 118 from page 89 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class Vehicle { }

class Car extends Vehicle { }

class Sedan extends Car { }

class Shop<T> {
  private T thing;
  public T buyFrom() {
    return thing;
  }
  public void sellTo(T thing) {
    this.thing = thing;
  }
}

class Example118 {
  public static void main(String[] args) {
    Object object = new Object();
    Vehicle vehicle = new Vehicle();
    Car car = new Car();
    Sedan sedan = new Sedan();

    Shop b0;
    b0 = new Shop<Object>();
    b0 = new Shop<Vehicle>();
    b0 = new Shop<Car>();
    b0 = new Shop<Sedan>();
    Object o0 = b0.buyFrom();
    // Vehicle v4 = b0.buyFrom();           // Illegal
    // Car c4 = b0.buyFrom();               // Illegal
    // Sedan s4 = b0.buyFrom();             // Illegal
    b0.sellTo(object);
    b0.sellTo(vehicle);
    b0.sellTo(car);
    b0.sellTo(sedan);
    b0.sellTo(b0.buyFrom());

    Shop<Car> b1;
    // b1 = new Shop<Object>();             // Illegal
    // b1 = new Shop<Vehicle>();            // Illegal
    b1 = new Shop<Car>();
    // b1 = new Shop<Sedan>();              // Illegal
    Object o1 = b1.buyFrom();
    Vehicle v1 = b1.buyFrom();
    Car c1 = b1.buyFrom();
    // Sedan s1 = b1.buyFrom();             // Illegal
    // b1.sellTo(object);                   // Illegal
    // b1.sellTo(vehicle);                  // Illegal
    b1.sellTo(car);
    b1.sellTo(sedan);
    b1.sellTo(b1.buyFrom());

    Shop<?> b2;
    b2 = new Shop<Object>();
    b2 = new Shop<Vehicle>();
    b2 = new Shop<Car>();
    b2 = new Shop<Sedan>();
    Object o2 = b2.buyFrom();
    // Vehicle v2 = b2.buyFrom();           // Illegal
    // Car c2 = b2.buyFrom();               // Illegal
    // Sedan s2 = b2.buyFrom();             // Illegal
    // b2.sellTo(object);                   // Illegal
    // b2.sellTo(vehicle);                  // Illegal
    // b2.sellTo(car);                      // Illegal
    // b2.sellTo(sedan);                    // Illegal
    // b2.sellTo(b2.buyFrom());             // Illegal

    Shop<? extends Car> b3;
    // b3 = new Shop<Object>();             // Illegal
    // b3 = new Shop<Vehicle>();            // Illegal
    b3 = new Shop<Car>();
    b3 = new Shop<Sedan>();
```

```java
    Object o3 = b3.buyFrom();
    Vehicle v3 = b3.buyFrom();
    Car c3 = b3.buyFrom();
    // Sedan s3 = b3.buyFrom();               // Illegal
    // b3.sellTo(object);                     // Illegal
    // b3.sellTo(vehicle);                    // Illegal
    // b3.sellTo(car);                        // Illegal
    // b3.sellTo(sedan);                      // Illegal
    // b3.sellTo(b3.buyFrom());               // Illegal

    Shop<? super Car> b4;
    b4 = new Shop<Object>();
    b4 = new Shop<Vehicle>();
    b4 = new Shop<Car>();
    // b4 = new Shop<Sedan>();                // Illegal
    Object o4 = b4.buyFrom();
    // Vehicle v4 = b4.buyFrom();             // Illegal
    // Car c4 = b4.buyFrom();                 // Illegal
    // Sedan s4 = b4.buyFrom();               // Illegal
    // b4.sellTo(object);                     // Illegal
    // b4.sellTo(vehicle);                    // Illegal
    b4.sellTo(car);
    b4.sellTo(sedan);
    // b4.sellTo(b4.buyFrom());               // Illegal

    // Shop<Object> b5 = new Shop<Car>();     // Illegal
    Shop<Object> b5 = new Shop<Object>();
    Object o5 = b5.buyFrom();
    // Vehicle v4 = b5.buyFrom();             // Illegal
    // Car c4 = b5.buyFrom();                 // Illegal
    // Sedan s4 = b5.buyFrom();               // Illegal
    b5.sellTo(object);
    b5.sellTo(vehicle);
    b5.sellTo(car);
    b5.sellTo(sedan);
    b5.sellTo(b5.buyFrom());

  }
}
```

```java
// Example 119 from page 91 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)


class ErasedHold {
  private Object contents;

  public void set(Object x) {
    contents = x;                        // Note: no cast
  }

  public Object get() {
    return contents;
  }
}

class Example119 {
  public static void main(String[] args) {
    ErasedHold h = new ErasedHold();
    h.set("foo");                        // Succeeds at run-time
    System.out.println("Succesfully executed  h.set(\"foo\")");
    h.get();                             // Succeeds at run-time
    System.out.println("Succesfully executed  h.get()");
    // String s = h.get();               // Illegal, rejected by compiler
    Integer i = (Integer)h.get();        // Legal, but fails at run-time
    System.out.println("Succesfully executed  Integer i = h.get()");
  }
}
```

```java
// Example 120 from page 91 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class G<T> { }

class Example120 {
  public static void main(String[] args) {
    new F<Double>().M();
  }
}

class F<T> {
  public void M() {
    T[] tarr;                            // Legal declaration
    G<T>[] ctarr;                        // Legal declaration
    G<Integer>[] ciarr;                  // Legal declaration

    // tarr = new T[5];                  // Illegal generic array creation
    // ctarr = new G<T>[5];              // Illegal generic array creation
    // ciarr = new G<Integer>[5];        // Illegal generic array creation

    ArrayList<T> tlist;                  // Legal declaration
    ArrayList<G<T>> ctlist;              // Legal declaration
    ArrayList<G<Integer>> cilist;        // Legal declaration

    tlist = new ArrayList<T>();          // Legal arraylist creation
    ctlist = new ArrayList<G<T>>();      // Legal arraylist creation
    cilist = new ArrayList<G<Integer>>();  // Legal arraylist creation
  }
}
```

```java
// Example 121 from page 93 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example121 {
  public static void main(String[] args) {
    List<String> list1 = new LinkedList<String>();
    list1.add("list"); list1.add("dup"); list1.add("x"); list1.add("dup");
    traverse(list1);                    // Must print: list dup x dup

    List<String> list2 = new ArrayList<String>();
    list2.add("list"); list2.add("dup"); list2.add("x"); list2.add("dup");
    traverse(list2);                    // Must print: list dup x dup

    Set<String> set1 = new HashSet<String>();
    set1.add("set"); set1.add("dup"); set1.add("x"); set1.add("dup");
    traverse(set1);                     // May print:  x dup set

    SortedSet<String> set2 = new TreeSet<String>();
    set2.add("set"); set2.add("dup"); set2.add("x"); set2.add("dup");
    traverse(set2);                     // Must print: dup set x

    LinkedHashSet<String> set3 = new LinkedHashSet<String>();
    set3.add("set"); set3.add("dup"); set3.add("x"); set3.add("dup");
    traverse(set3);                     // Must print: set dup x

    Map<String,String> m1 = new HashMap<String,String>();
    m1.put("map", "J"); m1.put("dup", "K"); m1.put("x", "M"); m1.put("dup", "L");
    traverse(m1.keySet());              // May print:  x dup map
    traverse(m1.values());              // May print:  M L J

    SortedMap<String,String> m2 = new TreeMap<String,String>();
    m2.put("map", "J"); m2.put("dup", "K"); m2.put("x", "M"); m2.put("dup", "L");
    traverse(m2.keySet());              // Must print: dup map x
    traverse(m2.values());              // Must print: L J M

    LinkedHashMap<String,String> m3 = new LinkedHashMap<String,String>();
    m3.put("map", "J"); m3.put("dup", "K"); m3.put("x", "M"); m3.put("dup", "L");
    traverse(m3.keySet());              // Must print: map dup x
    traverse(m3.values());              // Must print: J L M
  }

  static void traverse(Collection<String> coll) {
    for (String elem : coll)
      System.out.print(elem + " ");
    System.out.println();
  }
}
```

```java
// Example 122 from page 97 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

// The HashSet is 2 to 5 times faster than Arrays.binarySearch, even
// for this list of strings of only 52 elements:

class Example122 {
  final static String[] keywordarray =
  { "abstract", "assert", "boolean", "break", "byte", "case", "catch",
    "char", "class", "const", "continue", "default", "do", "double", "else",
    "enum", "extends", "false", "final", "finally", "float", "for", "goto",
    "if", "implements", "import", "instanceof", "int", "interface", "long",
    "native", "new", "null", "package", "private", "protected", "public",
    "return", "short", "static", "strictfp", "super", "switch",
    "synchronized", "this", "throw", "throws", "transient", "true",
    "try", "void", "volatile", "while" };

  final static Set<String> keywords
    = new HashSet<String>(Arrays.asList(keywordarray));

  static boolean isKeyword1(String id)
  { return keywords.contains(id); }

  static boolean isKeyword2(String id)
  { return Arrays.binarySearch(keywordarray, id) >= 0; }

  public static void main(String[] args) {
    if (args.length != 2)
      System.out.println("Usage: java Example122 <iterations> <word>");
    else {
      final int count = Integer.parseInt(args[0]);
      final String id = args[1];
      for (int i=0; i<keywordarray.length; i++)
        if (isKeyword1(keywordarray[i]) != isKeyword2(keywordarray[i]))
          System.out.println("Error at i = " + i);
      if (isKeyword1(id) != isKeyword2(id))
        System.out.println("Error at id = " + id);

      System.out.print("HashSet.contains    ");
      Timer t1 = new Timer();
      for (int i=0; i<count; i++)
        isKeyword1(id);
      System.out.println(t1.check());

      System.out.print("Arrays.binarySearch ");
      Timer t2 = new Timer();
      for (int i=0; i<count; i++)
        isKeyword2(id);
      System.out.println(t2.check());
    }
  }

  private final static class Timer {
    private long starttime;

    public Timer() { starttime = System.currentTimeMillis(); }

    double check() {
      return (System.currentTimeMillis()-starttime)/1000.0;
    }
  }
}
```

```java
// Example 123 from page 97 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

// The method unique takes a collection of file names and returns the
// list without duplicates, but retaining the order of the given ones.

class Example123 {
  public static void main(String[] args) {
    String[] filenames =
      { "Lib.java", "Foo.java", "Bar.java", "Lib.java", "Foo.java", "Goo.java" };
    for (String filename : unique(filenames))
      System.out.println(filename);
  }

  public static String[] unique(String[] filenames) {
    LinkedHashSet<String> uniqueFiles = new LinkedHashSet<String>();
    for (String filename : filenames)
      uniqueFiles.add(filename);
    return uniqueFiles.toArray(new String[0]);
  }
}
```

```java
// Example 124 from page 97 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;
import java.io.*;

class Example124 {
  public static void main(String[] args) {
    SortedSet<String> filenames = new TreeSet<String>();
    File cwd = new File(".");           // Current working directory
    for (File f : cwd.listFiles())
      filenames.add(f.getName());
    for (String filename : filenames.subSet("P", "T"))
      System.out.println(filename);
  }
}
```

```java
// Example 125 from page 99 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

class Example125 {
  public static void main(String[] args) throws SQLException {
    String url = "jdbc:postgresql://localhost/messages";
    String usr = "nobody";
    String pwd = "";

    System.out.println("To run this example you need a data base server, a");
    System.out.println("database, and a suitable JDBC driver.");

    try { Class.forName("postgresql.Driver"); }
    catch (ClassNotFoundException e)
      { System.out.println("Cannot find the Postgresql driver"); }

    Connection conn = DriverManager.getConnection(url, usr, pwd);
    printNameAndMsg(getRows(conn, "select * from message"));
  }

  // Return the query result as an ArrayList of Map from String to Object
  static ArrayList<Map<String,Object>> getRows(Connection conn, String query)
    throws SQLException
  {
    Statement stmt = conn.createStatement();
    ResultSet rset = stmt.executeQuery(query);
    ResultSetMetaData rsmd = rset.getMetaData();
    int columncount = rsmd.getColumnCount();
    ArrayList<Map<String,Object>> queryResult
      = new ArrayList<Map<String,Object>>();
    while (rset.next()) {
      Map<String,Object> row = new HashMap<String,Object>();
      for (int i=1; i<=columncount; i++)
        row.put(rsmd.getColumnName(i), rset.getObject(i));
      queryResult.add(row);
    }
    return queryResult;
  }

  static void printNameAndMsg(Collection<Map<String,Object>> coll) {
    for (Map<String,Object> row : coll)
      System.out.println(row.get("name") + ": " + row.get("msg"));
  }
}
```

```java
// Example 126 from page 99 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example126 {
  private static final HashMap<String,Integer> wdayNumber = new HashMap<String,Integer>();

  static { // Static initializer block, executed once
    int wdayno = 0;
    String[] wdays =
      { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };
    for (String wday : wdays)
      wdayNumber.put(wday, wdayno++);
  }

  public static int wdayno5(String wday) {
    Integer res = wdayNumber.get(wday);
    return res == null ? -1 : res;
  }

  public static void main(String[] args) {
    System.out.println("Thursday is " + wdayno5("Thursday"));
  }
}
```

```java
// Example 127 from page 101 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;
import java.io.*;

// A concordance is a TreeMap mapping a word (a String) to a TreeSet
// of linenumbers (Integer objects)

class Example127 {
  public static void main(String[] args) throws IOException {
    if (args.length != 1)
      System.out.println("Usage: java Example127 <filename>");
    else {
      SortedMap<String,SortedSet<Integer>> index = buildIndex(args[0]);
      printIndex(index);
    }
  }

  // Parse alphanumeric words from the given file, and record them in
  // the index.  The resulting index is a SortedMap from String to
  // SortedSet of Integer (the line numbers).

  static SortedMap<String,SortedSet<Integer>> buildIndex(String filename)
    throws IOException
  {
    Reader r = new BufferedReader(new FileReader(filename));
    StreamTokenizer stok = new StreamTokenizer(r);
    stok.quoteChar('"'); stok.ordinaryChars('!', '/');
    stok.nextToken();
    SortedMap<String,SortedSet<Integer>> index
      = new TreeMap<String,SortedSet<Integer>>();
    // = new TreeMap<String,SortedSet<Integer>>(new IgnoreCaseComparator());
    while (stok.ttype != StreamTokenizer.TT_EOF) {
      if (stok.ttype == StreamTokenizer.TT_WORD) {
        SortedSet<Integer> ts;
        if (index.containsKey(stok.sval))       // If word has a set, get it
          ts = index.get(stok.sval);
        else {
          ts = new TreeSet<Integer>();          // Otherwise create one
          index.put(stok.sval, ts);
        }
        ts.add(stok.lineno());
      }
      stok.nextToken();
    }
    return index;
  }

  // Print the concordance index by iterating over its entries, and
  // for each entry, iterate over the value which is a set.

  static void printIndex(SortedMap<String,SortedSet<Integer>> index) {
    for (Map.Entry<String,SortedSet<Integer>> entry : index.entrySet()) {
      System.out.print(entry.getKey() + ": ");
      SortedSet<Integer> lineNoSet = entry.getValue();
      for (int lineno : lineNoSet)
        System.out.print(lineno + " ");
      System.out.println();
    }
  }
}
```

```java
// Example 128 from page 101 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Time implements Comparable<Time> {
  public final int hh, mm;           // 24-hour clock

  public Time(int hh, int mm) { this.hh = hh; this.mm = mm; }

  // Return neg if this before o; return pos if this after o; return zero if same
  public int compareTo(Time t) {
    return hh != t.hh ? hh - t.hh : mm - t.mm;
  }

  public boolean equals(Object o) {
    if (this == o)                                   // fast, frequent case
      return true;
    if (o == null || this.getClass() != o.getClass()) // null or not same type
      return false;
    Time t = (Time)o;                                // here o instanceof Time
    return hh == t.hh && mm == t.mm;
  }

  public int hashCode() { return 60 * hh + mm; }

  public String toString()
  { return (hh < 10 ? "0"+hh : ""+hh) + ":" + (mm < 10 ? "0"+mm : ""+mm); }
}

class TimeComparator implements Comparator<Time> {
  // Return neg if t1 before t2; return pos if t1 after t2; return zero if same
  public int compare(Time t1, Time t2) {
    return t1.hh != t2.hh ? t1.hh - t2.hh : t1.mm - t2.mm;
  }
}

class Example128 {
  public static void main(String[] args) {
    SortedMap<Time,String> datebook = new TreeMap<Time,String>();
    datebook.put(new Time(12, 30), "Lunch");
    datebook.put(new Time(15, 30), "Afternoon coffee break");
    datebook.put(new Time( 9,  0), "Lecture");
    datebook.put(new Time(13, 15), "Board meeting");
    SortedMap<Time,String> pm = datebook.tailMap(new Time(12, 0));
    for (Map.Entry<Time,String> entry : pm.entrySet())
      System.out.println(entry.getKey() + " " + entry.getValue());
  }
}
```

```java
// Example 129 from page 103 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example129 {
  public static void main(String[] args) {
    List<String> list1 = new LinkedList<String>();
    list1.add("list"); list1.add("dup"); list1.add("x"); list1.add("dup");
    traverse(list1);                    // Must print: list dup x dup

    List<String> list2 = new ArrayList<String>();
    list2.add("list"); list2.add("dup"); list2.add("x"); list2.add("dup");
    traverse(list2);                    // Must print: list dup x dup

    Set<String> set1 = new HashSet<String>();
    set1.add("set"); set1.add("dup"); set1.add("x"); set1.add("dup");
    traverse(set1);                     // May print:  x dup set

    SortedSet<String> set2 = new TreeSet<String>();
    set2.add("set"); set2.add("dup"); set2.add("x"); set2.add("dup");
    traverse(set2);                     // Must print: dup set x

    LinkedHashSet<String> set3 = new LinkedHashSet<String>();
    set3.add("set"); set3.add("dup"); set3.add("x"); set3.add("dup");
    traverse(set3);                     // Must print: set dup x

    Map<String,String> m1 = new HashMap<String,String>();
    m1.put("map", "J"); m1.put("dup", "K"); m1.put("x", "M"); m1.put("dup", "L");
    traverse(m1.keySet());              // May print:  x dup map
    traverse(m1.values());              // May print:  M L J

    SortedMap<String,String> m2 = new TreeMap<String,String>();
    m2.put("map", "J"); m2.put("dup", "K"); m2.put("x", "M"); m2.put("dup", "L");
    traverse(m2.keySet());              // Must print: dup map x
    traverse(m2.values());              // Must print: L J M

    LinkedHashMap<String,String> m3 = new LinkedHashMap<String,String>();
    m3.put("map", "J"); m3.put("dup", "K"); m3.put("x", "M"); m3.put("dup", "L");
    traverse(m3.keySet());              // Must print: map dup x
    traverse(m3.values());              // Must print: J L M
  }

  static void traverse(Collection<String> coll) {
    for (String elem : coll)
      System.out.print(elem + " ");
    System.out.println();
  }
}
```

```java
// Example 130 from page 103 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;
import java.io.*;

// A concordance is a TreeMap mapping a word (a String) to a TreeSet
// of linenumbers (Integer objects)

class Example130 {
  public static void main(String[] args) throws IOException {
    if (args.length != 1)
      System.out.println("Usage: java Example130 <filename>");
    else {
      SortedMap<String,SortedSet<Integer>> index = buildIndex(args[0]);
      printIndex(index);
    }
  }

  // Parse alphanumeric words from the given file, and record them in
  // the index.  The resulting index is a SortedMap from String to
  // SortedSet of Integer (the line numbers).

  static SortedMap<String,SortedSet<Integer>> buildIndex(String filename)
    throws IOException
  {
    Reader r = new BufferedReader(new FileReader(filename));
    StreamTokenizer stok = new StreamTokenizer(r);
    stok.quoteChar('"'); stok.ordinaryChars('!', '/');
    stok.nextToken();
    SortedMap<String,SortedSet<Integer>> index
      = new TreeMap<String,SortedSet<Integer>>();
    // = new TreeMap<String,SortedSet<Integer>>(new IgnoreCaseComparator());
    while (stok.ttype != StreamTokenizer.TT_EOF) {
      if (stok.ttype == StreamTokenizer.TT_WORD) {
        SortedSet<Integer> ts;
        if (index.containsKey(stok.sval))       // If word has a set, get it
          ts = index.get(stok.sval);
        else {
          ts = new TreeSet<Integer>();          // Otherwise create one
          index.put(stok.sval, ts);
        }
        ts.add(stok.lineno());
      }
      stok.nextToken();
    }
    return index;
  }

  // Print the concordance index by iterating over its entries, and
  // for each entry, iterate over the value which is a set.

  static void printIndex(SortedMap<String,SortedSet<Integer>> index) {
    for (Map.Entry<String,SortedSet<Integer>> entry : index.entrySet()) {
      System.out.print(entry.getKey() + ": ");
      SortedSet<Integer> lineNoSet = entry.getValue();
      for (int lineno : lineNoSet)
        System.out.print(lineno + " ");
      System.out.println();
    }
  }
}
```

```java
// Example 131 from page 103 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example131 {
  public static void main(String[] args) {
    for (int i : fromTo(13, 17))
      System.out.println(i);
  }

  public static Iterable<Integer> fromTo(final int m, final int n) {
    class FromToIterator implements Iterator<Integer> {
      private int i = m;
      public boolean hasNext() { return i <= n; }
      public Integer next() {
        if (i <= n)
          return i++;
        else
          throw new NoSuchElementException();
      }
      public void remove() { throw new UnsupportedOperationException(); }
    }
    class FromToIterable implements Iterable<Integer> {
      public Iterator<Integer> iterator() {
        return new FromToIterator();
      }
    }
    return new FromToIterable();
  }
}
```

```java
// Example 133 from page 105 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Time implements Comparable<Time> {
  public final int hh, mm;           // 24-hour clock

  public Time(int hh, int mm) { this.hh = hh; this.mm = mm; }

  // Return neg if this before o; return pos if this after o; return zero if same
  public int compareTo(Time t) {
    return hh != t.hh ? hh - t.hh : mm - t.mm;
  }

  public boolean equals(Object o) {
    if (this == o)                                   // fast, frequent case
      return true;
    if (o == null || this.getClass() != o.getClass()) // null or not same type
      return false;
    Time t = (Time)o;                                // here o instanceof Time
    return hh == t.hh && mm == t.mm;
  }

  public int hashCode() { return 60 * hh + mm; }

  public String toString()
  { return (hh < 10 ? "0"+hh : ""+hh) + ":" + (mm < 10 ? "0"+mm : ""+mm); }
}

class TimeComparator implements Comparator<Time> {
  // Return neg if t1 before t2; return pos if t1 after t2; return zero if same
  public int compare(Time t1, Time t2) {
    return t1.hh != t2.hh ? t1.hh - t2.hh : t1.mm - t2.mm;
  }
}

class Example133 {
  public static void main(String[] args) {
    SortedMap<Time,String> datebook = new TreeMap<Time,String>();
    datebook.put(new Time(12, 30), "Lunch");
    datebook.put(new Time(15, 30), "Afternoon coffee break");
    datebook.put(new Time( 9,  0), "Lecture");
    datebook.put(new Time(13, 15), "Board meeting");
    SortedMap<Time,String> pm = datebook.tailMap(new Time(12, 0));
    for (Map.Entry<Time,String> entry : pm.entrySet())
      System.out.println(entry.getKey() + " " + entry.getValue());
  }
}
```

```java
// Example 134 from page 105 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Time implements Comparable<Time> {
  public final int hh, mm;           // 24-hour clock

  public Time(int hh, int mm) { this.hh = hh; this.mm = mm; }

  // Return neg if this before o; return pos if this after o; return zero if same
  public int compareTo(Time t) {
    return hh != t.hh ? hh - t.hh : mm - t.mm;
  }

  public boolean equals(Object o) {
    if (this == o)                                   // fast, frequent case
      return true;
    if (o == null || this.getClass() != o.getClass()) // null or not same type
      return false;
    Time t = (Time)o;                                // here o instanceof Time
    return hh == t.hh && mm == t.mm;
  }

  public int hashCode() { return 60 * hh + mm; }

  public String toString()
  { return (hh < 10 ? "0"+hh : ""+hh) + ":" + (mm < 10 ? "0"+mm : ""+mm); }
}

class TimeComparator implements Comparator<Time> {
  // Return neg if t1 before t2; return pos if t1 after t2; return zero if same
  public int compare(Time t1, Time t2) {
    return t1.hh != t2.hh ? t1.hh - t2.hh : t1.mm - t2.mm;
  }
}

class Example134 {
  public static void main(String[] args) {
    SortedMap<Time,String> datebook = new TreeMap<Time,String>();
    datebook.put(new Time(12, 30), "Lunch");
    datebook.put(new Time(15, 30), "Afternoon coffee break");
    datebook.put(new Time( 9,  0), "Lecture");
    datebook.put(new Time(13, 15), "Board meeting");
    SortedMap<Time,String> pm = datebook.tailMap(new Time(12, 0));
    for (Map.Entry<Time,String> entry : pm.entrySet())
      System.out.println(entry.getKey() + " " + entry.getValue());
  }
}
```

```java
// Example 135 from page 105 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;
import java.io.*;

// Strings that differ only in case should be next to each other, with
// upper case before lower case.

class IgnoreCaseComparator implements Comparator<String> {
  public int compare(String s1, String s2) {
    int res = s1.compareToIgnoreCase(s2);
    return res != 0 ? res : s1.compareTo(s2);
  }
}

// A concordance is a TreeMap mapping a word (a String) to a TreeSet
// of linenumbers (Integer objects)

class Example135 {
  public static void main(String[] args) throws IOException {
    if (args.length != 1)
      System.out.println("Usage: java Example135 <filename>");
    else {
      SortedMap<String,SortedSet<Integer>> index = buildIndex(args[0]);
      printIndex(index);
    }
  }

  // Parse alphanumeric words from the given file, and record them in
  // the index.  The resulting index is a SortedMap from String to
  // SortedSet of Integer (the line numbers).

  static SortedMap<String,SortedSet<Integer>> buildIndex(String filename)
    throws IOException
  {
    Reader r = new BufferedReader(new FileReader(filename));
    StreamTokenizer stok = new StreamTokenizer(r);
    stok.quoteChar('"'); stok.ordinaryChars('!', '/');
    stok.nextToken();
    SortedMap<String,SortedSet<Integer>> index
      = new TreeMap<String,SortedSet<Integer>>(new IgnoreCaseComparator());
    while (stok.ttype != StreamTokenizer.TT_EOF) {
      if (stok.ttype == StreamTokenizer.TT_WORD) {
        SortedSet<Integer> ts;
        if (index.containsKey(stok.sval))       // If word has a set, get it
          ts = index.get(stok.sval);
        else {
          ts = new TreeSet<Integer>();          // Otherwise create one
          index.put(stok.sval, ts);
        }
        ts.add(stok.lineno());
      }
      stok.nextToken();
    }
    return index;
  }

  // Print the concordance index by iterating over its entries, and
  // for each entry, iterate over the value which is a set.

  static void printIndex(SortedMap<String,SortedSet<Integer>> index) {
    for (Map.Entry<String,SortedSet<Integer>> entry : index.entrySet()) {
      System.out.print(entry.getKey() + ": ");
      SortedSet<Integer> lineNoSet = entry.getValue();
      for (int lineno : lineNoSet)
        System.out.print(lineno + " ");
      System.out.println();
    }
  }
}
```

```java
// Example 138 from page 109 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;

class Example138 {
  public static void main(String[] args) {
    Set<Set<Integer>> ss = new HashSet<Set<Integer>>();
    ss.add(mkIntegerSet(new int[] { 2, 3 }));
    ss.add(mkIntegerSet(new int[] { 1, 3 }));
    ss.add(mkIntegerSet(new int[] { 1, 2 }));
    System.out.println("ss = " + ss);
    Set<Set<Integer>> tt = intersectionClose(ss);
    System.out.println("tt = " + tt);
  }

  static Set<Integer> mkIntegerSet(int[] a) {
    TreeSet<Integer> ts = new TreeSet<Integer>();
    for (int i=0; i<a.length; i++)
      ts.add(a[i]);
    return ts;
  }

  // Given a set ss of sets of T, compute its intersection
  // closure, that is, the least set tt such that ss is a subset of tt
  // and such that for any two sets t1 and t2 in tt, their
  // intersection is also in tt.

  // For instance, if ss is {{2,3}, {1,3}, {1,2}},
  // then tt is {{2,3}, {1,3}, {1,2}, {3}, {2}, {1}, {}}.

  // Both the argument and the result is a Set of Set of T.

  static <T> Set<Set<T>> intersectionClose(Set<Set<T>> ss) {
    LinkedList<Set<T>> worklist = new LinkedList<Set<T>>(ss);
    Set<Set<T>> tt = new HashSet<Set<T>>();
    while (!worklist.isEmpty()) {
      Set<T> s = worklist.removeLast();
      for (Set<T> t : tt) {
        Set<T> ts = new TreeSet<T>(t);
        ts.retainAll(s);                 // ts is the intersection of t and s
        if (!tt.contains(ts))
          worklist.add(ts);
      }
      tt.add(s);
    }
    return tt;
  }
}
```

```java
// Example 139 from page 109 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.util.*;
import java.io.*;

// Regular expressions, NFAs, DFAs, and dot graphs

// This file contains, in order:
//   * A class Nfa for representing an NFA (a nondeterministic finite
//     automaton), and for converting it to a DFA (a deterministic
//     finite automaton).  Most complexity is in this class.
//   * A class Dfa for representing a DFA, a deterministic finite
//     automaton, and for writing a dot input file representing the DFA.
//   * Classes for representing regular expressions, and for building an
//     NFA from a regular expression
//   * A test class that creates an NFA, a DFA, and a dot input file
//     for a number of small regular expressions.  The DFAs are
//     not minimized.

/* Class Nfa and conversion from NFA to DFA ---------------------------

  A nondeterministic finite automaton (NFA) is represented as a HashMap
  from state number (Integer) to a List of Transitions, a Transition
  being a pair of a label lab (a String, null meaning epsilon) and a
  target state (an Integer).

  A DFA is created from an NFA in two steps:

    (1) Construct a DFA whose states are composite, namely sets of NFA
        states (Set of Integer).  This is done by methods
        compositeDfaTrans and epsilonClose.

    (2) Replace composite states (Set of Integer) by simple states
        (Integer).  This is done by methods rename and mkRenamer.

  Method compositeDfaTrans works as follows:

    Create the epsilon-closure S0 (a HashSet of Integers) of the start
    state s0, and put it in a worklist.  Create an empty DFA transition
    relation, which is a HashMap from a composite state (an
    epsilon-closed HashSet of Integers) to a HashMap from a label
    (a non-null String) to a composite state.

    Repeatedly choose a composite state S from the worklist.  If it is
    not already in the keyset of the DFA transition relation, compute
    for every non-epsilon label lab the set T of states reachable by
    that label from some state s in S.  Compute the epsilon-closure
    Tclose of every such state T and put it on the worklist.  Then add
    the transition S -lab-> Tclose to the DFA transition relation, for
    every lab.

  Method epsilonClosure works as follows:

    Given a set S of states.  Put the states of S in a worklist.
    Repeatedly choose a state s from the worklist, and consider all
    epsilon-transitions s -eps-> s' from s.  If s' is in S already, then
    do nothing; otherwise add s' to S and the worklist.  When the
    worklist is empty, S is epsilon-closed; return S.

  Method mkRenamer works as follows:

    Given a Map from Set of Integer to something, create an injective
    Map from Set of Integer to Integer, by choosing a fresh Integer for
    every value of the map.

  Method rename works as follows:

    Given a Map from Set of Integer to Map from String to Set of
    Integer, use the result of mkRenamer to replace all Sets of Integers
    by Integers.

*/

class Nfa {
  private Integer startState;
  private Integer exitState;    // This is the unique accept state
  private Map<Integer,List<Transition>> trans;

  public Nfa(Integer startState, Integer exitState) {
    this.startState = startState; this.exitState = exitState;
    trans = new HashMap<Integer,List<Transition>>();
    if (!startState.equals(exitState))
      trans.put(exitState, new LinkedList<Transition>());
  }

  public Integer getStart() { return startState; }

  public Integer getExit() { return exitState; }

  public Map<Integer,List<Transition>> getTrans() { return trans; }

  public void addTrans(Integer s1, String lab, Integer s2) {
    List<Transition> s1Trans;
    if (trans.containsKey(s1))
      s1Trans = trans.get(s1);
    else {
      s1Trans = new LinkedList<Transition>();
      trans.put(s1, s1Trans);
    }
    s1Trans.add(new Transition(lab, s2));
  }

  public void addTrans(Map.Entry<Integer, List<Transition>> tr) {
    // assert !trans.containsKey(tr.getKey());
    trans.put(tr.getKey(), tr.getValue());
  }

  public String toString() {
    return "NFA start=" + startState + " exit=" + exitState + "\n" + trans;
  }

  // Construct the transition relation of a composite-state DFA from
  // an NFA with start state s0 and transition relation trans (a
  // HashMap from Integer to List of Transition).  The start state of
  // the constructed DFA is the epsilon closure of s0, and its
  // transition relation is a HashMap from a composite state (a
  // HashSet of Integers) to a HashMap from label (a String) to a
  // composite state (a HashSet of Integers).

  static Map<Set<Integer>,Map<String,Set<Integer>>>
    compositeDfaTrans(Integer s0, Map<Integer,List<Transition>> trans) {
    Set<Integer> S0 = epsilonClose(Collections.singleton(s0), trans);
    LinkedList<Set<Integer>> worklist = new LinkedList<Set<Integer>>();
    worklist.add(S0);
    // The transition relation of the DFA
    Map<Set<Integer>,Map<String,Set<Integer>>> res =
      new HashMap<Set<Integer>,Map<String,Set<Integer>>>();
    while (!worklist.isEmpty()) {
      Set<Integer> S = worklist.removeFirst();
      if (!res.containsKey(S)) {
        // The S -lab-> T transition relation being constructed for a given S
        Map<String, Set<Integer>> STrans = new HashMap<String, Set<Integer>>();
        // For all s in S, consider all transitions s -lab-> t
        for (Integer s : S) {
          // For all non-epsilon transitions s -lab-> t, add t to T
          for (Transition tr : trans.get(s)) {
            if (tr.lab != null) {            // Already a transition on lab
              Set<Integer> toState;
              if (STrans.containsKey(tr.lab))
                toState = STrans.get(tr.lab);
              else {                         // No transitions on lab yet
                toState = new HashSet<Integer>();
                STrans.put(tr.lab, toState);
              }
              toState.add(tr.target);
            }
          }
        }
        // Epsilon-close all T such that S -lab-> T, and put on worklist
        HashMap<String, Set<Integer>> STransClosed =
          new HashMap<String, Set<Integer>>();
        for (Map.Entry<String,Set<Integer>> entry : STrans.entrySet()) {
          Set<Integer> Tclose = epsilonClose(entry.getValue(), trans);
```

```
        STransClosed.put(entry.getKey(), Tclose);
        worklist.add(Tclose);
      }
      res.put(S, STransClosed);
    }
  }
  return res;
}

// Compute epsilon-closure of state set S in transition relation trans.
// Parameter S is a Set of Integer.
// Parameter trans is a HashMap from Integer to List of Transition.
// The result is a Set of Integer.

static Set<Integer> epsilonClose(Set<Integer> S,
                                 Map<Integer,List<Transition>> trans){
  LinkedList<Integer> worklist = new LinkedList<Integer>(S);
  Set<Integer> res = new HashSet<Integer>(S);
  while (!worklist.isEmpty()) {
    Integer s = worklist.removeFirst();
    for (Transition tr : trans.get(s)) {
      if (tr.lab == null && !res.contains(tr.target)) {
        res.add(tr.target);
        worklist.add(tr.target);
      }
    }
  }
  return res;
}

// Compute a renamer, which is a Map from Set of Integer to Integer,
// provided parameter states is a Collection of Set of Integer.

static Map<Set<Integer>,Integer> mkRenamer(Collection<Set<Integer>> states) {
  Map<Set<Integer>,Integer> renamer = new HashMap<Set<Integer>,Integer>();
  for (Set<Integer> k : states)
    renamer.put(k, renamer.size());
  return renamer;
}

// Using a renamer (a Map from Set of Integer to Integer), replace
// composite (Set of Integer) states with simple (Integer) states in
// the transition relation trans, which is assumed to be a Map from
// Set of Integer to Map from String to Set of Integer.  The result
// is a Map from Integer to Map from String to Integer.

static Map<Integer,Map<String,Integer>>
  rename(Map<Set<Integer>,Integer> renamer,
         Map<Set<Integer>,Map<String,Set<Integer>>> trans) {
  Map<Integer,Map<String,Integer>> newtrans =
    new HashMap<Integer,Map<String,Integer>>();
  for (Map.Entry<Set<Integer>,Map<String,Set<Integer>>>
         entry : trans.entrySet()) {
    Set<Integer> k = entry.getKey();
    Map<String,Integer> newktrans = new HashMap<String,Integer>();
    for (Map.Entry<String,Set<Integer>> tr : entry.getValue().entrySet())
      newktrans.put(tr.getKey(), renamer.get(tr.getValue()));
    newtrans.put(renamer.get(k), newktrans);
  }
  return newtrans;
}

static Set<Integer> acceptStates(Set<Set<Integer>> states,
                                 Map<Set<Integer>, Integer> renamer,
                                 Integer exit) {
  Set<Integer> acceptStates = new HashSet<Integer>();
  for (Set<Integer> state : states)
    if (state.contains(exit))
      acceptStates.add(renamer.get(state));
  return acceptStates;
}

public Dfa toDfa() {
  Map<Set<Integer>,Map<String,Set<Integer>>> cDfaTrans
    = compositeDfaTrans(startState, trans);
  Set<Integer> cDfaStart
    = epsilonClose(Collections.singleton(startState), trans);
```

```
    Set<Set<Integer>> cDfaStates = cDfaTrans.keySet();
    Map<Set<Integer>,Integer> renamer = mkRenamer(cDfaStates);
    Map<Integer,Map<String,Integer>> simpleDfaTrans
      = rename(renamer, cDfaTrans);
    Integer simpleDfaStart = renamer.get(cDfaStart);
    Set<Integer> simpleDfaAccept
      = acceptStates(cDfaStates, renamer, exitState);
    return new Dfa(simpleDfaStart, simpleDfaAccept, simpleDfaTrans);
  }

  // Nested class for creating distinctly named states when constructing NFAs

  static class NameSource {
    private int nextName = 0;
    public Integer next() { return nextName++; }
  }

  // Nested class for representing a transition from one state to another

  public static class Transition {
    String lab;
    Integer target;

    public Transition(String lab, Integer target)
    { this.lab = lab; this.target = target; }

    public String toString() {
      return "-" + lab + "-> " + target;
    }
  }
}


// Class Dfa, deterministic finite automata --------------------------

/*
  A deterministic finite automaton (DFA) is represented as a Map from
  state number (Integer) to a Map from label (a String, non-null) to a
  target state (an Integer).
*/

class Dfa {
  private Integer startState;
  private Set<Integer> acceptStates;
  private Map<Integer,Map<String,Integer>> trans;

  public Dfa(Integer startState,
             Set<Integer> acceptStates,
             Map<Integer,Map<String,Integer>> trans) {
    this.startState = startState;
    this.acceptStates = acceptStates;
    this.trans = trans;
  }

  public Integer getStart() { return startState; }

  public Set<Integer> getAccept() { return acceptStates; }

  public Map<Integer,Map<String,Integer>> getTrans() { return trans; }

  public String toString() {
    return "DFA start=" + startState + "\naccept=" + acceptStates
      + "\n" + trans;
  }

  // Write an input file for the dot program.  You can find dot at
  // http://www.research.att.com/sw/tools/graphviz/

  public void writeDot(String filename) throws IOException {
    PrintWriter out = new PrintWriter(new FileWriter(filename));
    out.println("// Format this file as a Postscript file with ");
    out.println("//    dot " + filename + " -Tps -o out.ps\n");
    out.println("digraph dfa {");
    out.println("size=\"11,8.25\";");
    out.println("rotate=90;");
    out.println("rankdir=LR;");
    out.println("n999999 [style=invis];");    // Invisible start node
```

```java
    out.println("n999999 -> n" + startState); // Edge into start state

    // Accept states are double circles
    for (Integer state : trans.keySet())
      if (acceptStates.contains(state))
        out.println("n" + state + " [peripheries=2];");

    // The transitions
    for (Map.Entry<Integer,Map<String,Integer>> entry : trans.entrySet()) {
      Integer s1 = entry.getKey();
      for (Map.Entry<String,Integer> s1Trans : entry.getValue().entrySet()) {
        String lab = s1Trans.getKey();
        Integer s2 = s1Trans.getValue();
        out.println("n" + s1 + " -> n" + s2 + " [label=" + lab + "];");
      }
    }
    out.println("}");
    out.close();
  }
}

// Regular expressions ----------------------------------------------
//
// Abstract syntax of regular expressions
//    r ::= A | r1 r2 | (r1|r2) | r*
//

abstract class Regex {
  abstract public Nfa mkNfa(Nfa.NameSource names);
}

class Sym extends Regex {
  String sym;

  public Sym(String sym) {
    this.sym = sym;
  }

  // The resulting nfa0 has form s0s -sym-> s0e

  public Nfa mkNfa(Nfa.NameSource names) {
    Integer s0s = names.next();
    Integer s0e = names.next();
    Nfa nfa0 = new Nfa(s0s, s0e);
    nfa0.addTrans(s0s, sym, s0e);
    return nfa0;
  }
}

class Seq extends Regex {
  Regex r1, r2;

  public Seq(Regex r1, Regex r2) {
    this.r1 = r1; this.r2 = r2;
  }

  // If   nfa1 has form s1s ----> s1e
  // and  nfa2 has form s2s ----> s2e
  // then nfa0 has form s1s ----> s1e -eps-> s2s ----> s2e

  public Nfa mkNfa(Nfa.NameSource names) {
    Nfa nfa1 = r1.mkNfa(names);
    Nfa nfa2 = r2.mkNfa(names);
    Nfa nfa0 = new Nfa(nfa1.getStart(), nfa2.getExit());
    for (Map.Entry<Integer,List<Nfa.Transition>> entry : nfa1.getTrans().entrySet())
      nfa0.addTrans(entry);
    for (Map.Entry<Integer,List<Nfa.Transition>> entry : nfa2.getTrans().entrySet())
      nfa0.addTrans(entry);
    nfa0.addTrans(nfa1.getExit(), null, nfa2.getStart());
    return nfa0;
  }
}

class Alt extends Regex {
  Regex r1, r2;

  public Alt(Regex r1, Regex r2) {
    this.r1 = r1; this.r2 = r2;
  }

  // If   nfa1 has form s1s ----> s1e
  // and  nfa2 has form s2s ----> s2e
  // then nfa0 has form s0s -eps-> s1s ----> s1e -eps-> s0e
  //                    s0s -eps-> s2s ----> s2e -eps-> s0e

  public Nfa mkNfa(Nfa.NameSource names) {
    Nfa nfa1 = r1.mkNfa(names);
    Nfa nfa2 = r2.mkNfa(names);
    Integer s0s = names.next();
    Integer s0e = names.next();
    Nfa nfa0 = new Nfa(s0s, s0e);
    for (Map.Entry<Integer,List<Nfa.Transition>> entry : nfa1.getTrans().entrySet())
      nfa0.addTrans(entry);
    for (Map.Entry<Integer,List<Nfa.Transition>> entry : nfa2.getTrans().entrySet())
      nfa0.addTrans(entry);
    nfa0.addTrans(s0s, null, nfa1.getStart());
    nfa0.addTrans(s0s, null, nfa2.getStart());
    nfa0.addTrans(nfa1.getExit(), null, s0e);
    nfa0.addTrans(nfa2.getExit(), null, s0e);
    return nfa0;
  }
}

class Star extends Regex {
  Regex r;

  public Star(Regex r) {
    this.r = r;
  }

  // If   nfa1 has form s1s ----> s1e
  // then nfa0 has form s0s ----> s0s
  //                    s0s -eps-> s1s
  //                    s1e -eps-> s0s

  public Nfa mkNfa(Nfa.NameSource names) {
    Nfa nfa1 = r.mkNfa(names);
    Integer s0s = names.next();
    Nfa nfa0 = new Nfa(s0s, s0s);
    for (Map.Entry<Integer,List<Nfa.Transition>> entry : nfa1.getTrans().entrySet())
      nfa0.addTrans(entry);
    nfa0.addTrans(s0s, null, nfa1.getStart());
    nfa0.addTrans(nfa1.getExit(), null, s0s);
    return nfa0;
  }
}

// Trying the RE->NFA->DFA translation on three regular expressions

class Example139 {
  public static void main(String[] args)
    throws IOException {
    Regex a = new Sym("A");
    Regex b = new Sym("B");
    Regex r = new Seq(new Star(new Alt(a, b)), new Seq(a, b));
    // The regular expression (a|b)*ab
    buildAndShow("dfa1.dot", r);
    // The regular expression ((a|b)*ab)*
    buildAndShow("dfa2.dot", new Star(r));
    // The regular expression ((a|b)*ab)((a|b)*ab)
    buildAndShow("dfa3.dot", new Seq(r, r));
  }

  public static void buildAndShow(String filename, Regex r)
    throws IOException {
    Nfa nfa = r.mkNfa(new Nfa.NameSource());
    System.out.println(nfa);
    Dfa dfa = nfa.toDfa();
    System.out.println(dfa);
    System.out.println("Writing DFA graph to file " + filename);
    dfa.writeDot(filename);
    System.out.println();
  }
}
```

```
// Example 140 from page 111 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example140 {
  public static void main(String[] args) throws IOException {
    System.out.println("Type some lines of text, end with an empty line:");
    BufferedReader r = new BufferedReader(new InputStreamReader(System.in));
    int count = 0;
    String s = r.readLine();
    while (s != null && !s.equals("")) {
      count++;
      s = r.readLine();
    }
    System.out.println("You entered " + count + " non-empty lines");
  }
}
```

```java
// Example 141 from page 113 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example141 {
  public static void main(String[] args)
    throws IOException, ClassNotFoundException {
    // Write numbers, words etc on file "f.txt" in human-readable form:
    PrintWriter pwr = new PrintWriter(new FileWriter("f.txt"));
    pwr.print(4711); pwr.print(' '); pwr.print("cool"); pwr.close();
    // Read numbers and words from human-readable text file "f.txt":
    StreamTokenizer stok = new StreamTokenizer(new FileReader("f.txt"));
    int tok = stok.nextToken();
    while (tok != StreamTokenizer.TT_EOF)
      { System.out.println(stok.sval); tok = stok.nextToken(); }
    // Write primitive values to a binary file "p.dat":
    DataOutputStream dos = new DataOutputStream(new FileOutputStream("p.dat"));
    dos.writeInt(4711); dos.writeChar(' '); dos.writeUTF("cool"); dos.close();
    // Read primitive values from binary file "p.dat":
    DataInputStream dis = new DataInputStream(new FileInputStream("p.dat"));
    System.out.println(dis.readInt()+"|"+dis.readChar()+"|"+ dis.readUTF());
    // Write an object or array to binary file "o.dat":
    ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("o.dat"));
    oos.writeObject(new int[] { 2, 3, 5, 7, 11 }); oos.close();
    // Read objects or arrays from binary file "o.dat":
    ObjectInputStream ois = new ObjectInputStream(new FileInputStream("o.dat"));
    int[] ia = (int[])(ois.readObject());
    System.out.println(ia[0]+","+ia[1]+","+ia[2]+","+ia[3]+","+ia[4]);
    // Read and write parts of file "raf.dat" in arbitrary order:
    RandomAccessFile raf = new RandomAccessFile("raf.dat", "rw");
    raf.writeDouble(3.1415); raf.writeInt(42);
    raf.seek(0); System.out.println(raf.readDouble() + " " + raf.readInt());
    // Read from a String s as if it were a text file:
    Reader r = new StringReader("abc");
    System.out.println("abc: " + (char)r.read() + (char)r.read() + (char)r.read());
    // Write to a StringBuilder as if it were a text file:
    Writer sw = new StringWriter();
    sw.write('d'); sw.write('e'); sw.write('f');
    System.out.println(sw.toString());
    // Write characters to standard output and standard error:
    System.out.println("std output"); System.err.println("std error");
    // Read characters from standard input (the keyboard):
    System.out.print("Type some characters and press Enter: ");
    BufferedReader bisr = new BufferedReader(new InputStreamReader(System.in));
    String response = bisr.readLine();
    System.out.println("You typed: '" + response + "'");
    // Read a byte from standard input (the keyboard):
    System.out.print("Type one character and press Enter: ");
    byte b = (byte)System.in.read();
    System.out.println("First byte of your input is: " + b);
  }
}
```

```java
// Example 142 from page 117 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example142 {
  public static void main(String[] args) throws IOException {
    System.out.println("Writing to file dice.txt");
    PrintWriter pw = new PrintWriter(new FileWriter("dice.txt"));
    for (int i=1; i<=1000; i++) {
      int die = (int)(1 + 6 * Math.random());
      pw.print(die); pw.print(' ');
      if (i % 20 == 0) pw.println();
    }
    pw.println();
    pw.close();                  // without this, the output file may be empty
  }
}
```

```java
// Example 143 from page 117 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example143 {
  public static void main(String[] args) throws IOException {
    fahrenheitCelsiusHtml();
    fahrenheitCelsiusText();
  }

  public static void fahrenheitCelsiusHtml() throws IOException {
    System.out.println("Printing temperature table to file temperature.html");
    PrintWriter pw = new PrintWriter(new FileWriter("temperature.html"));
    pw.println("<TABLE BORDER><TR><TH>Fahrenheit<TH>Celsius</TR>");
    for (int f=100; f<=400; f+=10) {
      double c = 5 * (f - 32.0) / 9;
      pw.format("<TR ALIGN=RIGHT><TD>%d<TD>%.1f%n", f, c);
    }
    pw.println("</TABLE>");
    pw.close();                 // Without this, the output file may be empty
  }

  public static void fahrenheitCelsiusText() throws IOException {
    System.out.println("Printing temperature table to file temperature.txt");
    PrintWriter pw = new PrintWriter(new FileWriter("temperature.txt"));
    pw.println("Fahrenheit   Celsius");
    for (int f=100; f<=400; f+=10) {
      double c = 5 * (f - 32.0) / 9;
      pw.format("%10d%10.1f%n", f, c);
    }
    pw.close();                 // Without this, the output file may be empty
  }
}
```

```java
// Example 144 from page 117 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example144 {
  public static void main(String[] args)
    throws IOException
  {
    Expr expr = mkAddition(4);
    expr.output(System.out); System.out.println();      // To standard output
    StringBuilder sb = new StringBuilder();             // To StringBuilder
    expr.output(sb);
    String s = sb.toString();                           // To String
    System.out.println(s);
    Writer wr = new FileWriter("expr.txt");
    System.out.println("Writing to file expr.txt");
    expr.output(wr); wr.append('\n');                   // To text file
    wr.close();
  }

  public static Expr mkAddition(int n) {
    if (n == 0)
      return new Cst(17);
    else {
      Expr e = mkAddition(n-1);
      return new Add(new Cst(n), new Add(e, e));
    }
  }
}

abstract class Expr {
  public abstract void output(Appendable sink) throws IOException;
}

class Cst extends Expr {
  private final int i;
  public Cst(int i) {
    this.i = i;
  }
  public void output(Appendable sink) throws IOException {
    sink.append(Integer.toString(i));
  }
}

class Add extends Expr {
  Expr e1, e2;
  public Add(Expr e1, Expr e2) {
    this.e1 = e1; this.e2 = e2;
  }
  public void output(Appendable sink) throws IOException {
    sink.append('('); e1.output(sink); sink.append('+');
    e2.output(sink); sink.append(')');
  }
}
```

```java
// Example 145 from page 119 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example145 {
  public static void main(String[] args) throws IOException {
    if (args.length != 1)
      System.out.println("Usage: java Example145 <filename>");
    else
      sumfile(args[0]);
  }

  static void sumfile(String filename) throws IOException {
    Reader r = new BufferedReader(new FileReader(filename));
    StreamTokenizer stok = new StreamTokenizer(r);
    double sum = 0;
    stok.nextToken();
    while (stok.ttype != StreamTokenizer.TT_EOF) {
      if (stok.ttype == StreamTokenizer.TT_NUMBER)
        sum += stok.nval;
      else
        System.out.println("Non-number: " + stok.sval);
      stok.nextToken();
    }
    System.out.println("The file sum is " + sum);
  }
}
```

```java
// Example 146 from page 119 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example146 {
  public static void main(String[] args) throws IOException {
    if (args.length != 1)
      System.out.println("Usage: java Example146 <filename>");
    else
      sumlines(args[0]);
  }

  static void sumlines(String filename) throws IOException {
    LineNumberReader lnr = new LineNumberReader(new FileReader(filename));
    lnr.setLineNumber(1);
    StreamTokenizer stok = new StreamTokenizer(lnr);
    stok.eolIsSignificant(true);
    stok.nextToken();
    while (stok.ttype != StreamTokenizer.TT_EOF) {
      int lineno = lnr.getLineNumber();
      //      System.out.println(stok.lineno() + " " + lineno);
      double sum = 0;
      while (stok.ttype != StreamTokenizer.TT_EOL) {
        if (stok.ttype == StreamTokenizer.TT_NUMBER)
          sum += stok.nval;
        stok.nextToken();
      }
      System.out.println("Sum of line " + lineno + " is " + sum);
      stok.nextToken();
    }
  }
}
```

```java
// Example 147 from page 123 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example147 {
  public static void main(String[] args) throws IOException {
    // Write: DataOutputStream implements DataOutput
    OutputStream os = new FileOutputStream("tmp1.dat");
    DataOutputStream daos = new DataOutputStream(os);
    writedata(daos);
    System.out.println("Wrote " + daos.size() + " bytes");
    daos.close();
    // Read: DataInputStream implements DataInput
    InputStream is = new FileInputStream("tmp1.dat");
    DataInputStream dais = new DataInputStream(is);
    readdata(dais);
    // Write and read: RandomAccessFile implements DataOutput and DataInput
    RandomAccessFile raf = new RandomAccessFile("tmp2.dat", "rw");
    writedata(raf);
    System.out.println("Wrote " + raf.getFilePointer() + " bytes");
    raf.seek(0);
    readdata(raf);
  }

  static void writedata(DataOutput out) throws IOException {
    out.writeBoolean(true);                         // Write 1 byte
    out.writeByte(120);                             // Write 1 byte
    out.writeBytes("foo");                          // Write 3 bytes
    out.writeBytes("fo");                           // Write 2 bytes
    out.writeChar('A');                             // Write 2 bytes
    out.writeChars("foo");                          // Write 6 bytes
    out.writeDouble(300.1);                         // Write 8 bytes
    out.writeFloat(300.2F);                         // Write 4 bytes
    out.writeInt(1234);                             // Write 4 bytes
    out.writeLong(12345L);                          // Write 8 bytes
    out.writeShort(32000);                          // Write 2 bytes
    out.writeUTF("foo");                            // Write 2 + 3 bytes
    out.writeUTF("Rhône");                          // Write 2 + 6 bytes
    out.writeByte(-1);                              // Write 1 byte
    out.writeShort(-1);                             // Write 2 bytes
  }

  static void readdata(DataInput in) throws IOException {
    byte[] buf1 = new byte[3];
    System.out.print(      in.readBoolean());       // Read 1 byte
    System.out.print(" " + in.readByte());          // Read 1 byte
    in.readFully(buf1);                             // Read 3 bytes
    in.readFully(buf1, 0, 2);                       // Read 2 bytes
    System.out.print(" " + in.readChar());          // Read 2 bytes
    System.out.print(" " + in.readChar()+in.readChar()+in.readChar());
    System.out.print(" " + in.readDouble());        // Read 8 bytes
    System.out.print(" " + in.readFloat());         // Read 4 bytes
    System.out.print(" " + in.readInt());           // Read 4 bytes
    System.out.print(" " + in.readLong());          // Read 8 bytes
    System.out.print(" " + in.readShort());         // Read 2 bytes
    System.out.print(" " + in.readUTF());           // Read 2 + 3 bytes
    System.out.print(" " + in.readUTF());           // Read 2 + 6 bytes
    System.out.print(" " + in.readUnsignedByte());  // Read 1 byte
    System.out.print(" " + in.readUnsignedShort()); // Read 2 bytes
    System.out.println();
  }
}
```

```java
// Example 148 from page 125 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class SC implements Serializable {
    static final long serialVersionUID = 50L;
    int ci;
}

class SO implements Serializable {
    static final long serialVersionUID = 50L;
  int i;
  SC c;

  SO(int i, SC c) { this.i = i; this.c = c; }

  void cprint() { System.out.print("i" + i + "c" + c.ci + " "); }
}

class Example148 {
  public static void main(String[] args)
    throws IOException, ClassNotFoundException {
    File f = new File("objects1.dat");
    if (!f.exists()) {
      System.out.println("Creating objects and writing them to file:");
      SC c = new SC();
      SO o1 = new SO(1, c), o2 = new SO(2, c);
      o1.c.ci = 3; o2.c.ci = 4;                 // update the shared c twice
      o1.cprint(); o2.cprint();                 // prints i1c4 i2c4
      OutputStream os = new FileOutputStream(f);
      ObjectOutputStream oos = new ObjectOutputStream(os);
      oos.writeObject(o1); oos.writeObject(o2); oos.close();
      System.out.println("\nRun the example again to read objects from file");
    } else {
      System.out.println("Reading objects from file (shared c):");
      InputStream is = new FileInputStream(f);
      ObjectInputStream ois = new ObjectInputStream(is);
      SO o1i = (SO)(ois.readObject()), o2i = (SO)(ois.readObject());
      o1i.cprint(); o2i.cprint();               // prints i1c4 i2c4
      o1i.c.ci = 5; o2i.c.ci = 6;               // update the shared c twice
      o1i.cprint(); o2i.cprint();               // prints i1c6 i2c6
      f.delete();
    }
    System.out.println();
  }
}
```

```java
// Example 149 from page 125 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example149 {
  public static void main(String[] args)
    throws IOException, ClassNotFoundException {
    File f = new File("objects2.dat");
    if (!f.exists()) {
      System.out.println("Creating objects and writing them to file:");
      SC c = new SC();
      SO o1 = new SO(1, c), o2 = new SO(2, c);
      o1.c.ci = 3; o2.c.ci = 4;                 // update the shared c twice
      o1.cprint(); o2.cprint();                 // prints i1c4 i2c4
      OutputStream os = new FileOutputStream(f);
      ObjectOutputStream oos1 = new ObjectOutputStream(os);
      oos1.writeObject(o1); oos1.flush();
      ObjectOutputStream oos2 = new ObjectOutputStream(os);
      oos2.writeObject(o2); oos2.close();
      System.out.println("\nRun the example again to read objects from file");
    } else {
      System.out.println("Reading objects from file (non-shared c):");
      InputStream is = new FileInputStream(f);
      ObjectInputStream ois1 = new ObjectInputStream(is);
      SO o1i = (SO)(ois1.readObject());
      ObjectInputStream ois2 = new ObjectInputStream(is);
      SO o2i = (SO)(ois2.readObject());
      o1i.cprint(); o2i.cprint();               // prints i1c4 i2c4
      o1i.c.ci = 5; o2i.c.ci = 6;               // update two different c's
      o1i.cprint(); o2i.cprint();               // prints i1c5 i2c6
      f.delete();
    }
    System.out.println();
  }
}
```

```java
// Example 150 from page 127 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Timer {
  private long starttime;

  public Timer() { starttime = System.currentTimeMillis(); }

  double check() {
    return (System.currentTimeMillis()-starttime)/1000.0;
  }
}

class Example150 {
  public static void main(String[] args) throws IOException {
    OutputStream os1 = new FileOutputStream("tmp1.dat");
    writeints("Unbuffered FileOutputStream: ", 1000000, os1);
    OutputStream os2 = new BufferedOutputStream(new FileOutputStream("tmp2.dat"));
    writeints("Buffered FileOutputStream:   ", 1000000, os2);
    Writer wr1 = new FileWriter("tmp1.dat");
    writeints("Unbuffered FileWriter: ", 1000000, wr1);
    Writer wr2 = new BufferedWriter(new FileWriter("tmp2.dat"));
    writeints("Buffered FileWriter:   ", 1000000, wr2);
  }

  static void writeints(String msg, int count, OutputStream os)
    throws IOException {
    Timer t = new Timer();
    for (int i=0; i<count; i++)
      os.write(i & 255);
    os.close();
    System.out.println(msg + t.check());
  }

  static void writeints(String msg, int count, Writer os)
    throws IOException {
    Timer t = new Timer();
    for (int i=0; i<count; i++)
      os.write(i & 255);
    os.close();
    System.out.println(msg + t.check());
  }
}
```

```java
// Example 151 from page 127 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;

// Compile with
//    javac -encoding Latin1 Examples.java

// On a Unix system, pipe the result through od (octal dump) to study
// the details of the encoding:
//    java Example151 | od -c | less

class Example151 {
  public static void main(String[] args)
    throws UnsupportedEncodingException
  {
    System.out.format("*** %-12s %-12s%n", "Encoding", "Canonical name");
    String s = "El Niño, süß, Ærøskøbing å, éclair, §2";
    writeIt(s, "US-ASCII");        //  7-bit US ASCII
    writeIt(s, "ISO-8859-1");      //  8-bit ISO Latin 1
    writeIt(s, "UTF-8");           //  8-bit Unicode encoding
    writeIt(s, "UTF-16");          // 16-bit Unicode encoding
    writeIt(s, "UTF-16BE");        // 16-bit Unicode encoding, big-endian
    writeIt(s, "UTF-16LE");        // 16-bit Unicode encoding, little-endian
    writeIt(s, "Cp1252");          // MS Windows codepage 1252
  }

  static void writeIt(String s, String enc)
    throws UnsupportedEncodingException
  {
    OutputStreamWriter osw = new OutputStreamWriter(System.out, enc);
    PrintWriter pw = new PrintWriter(osw);
    System.out.format("*** %-12s %-12s%n", enc, osw.getEncoding());
    pw.println(s); pw.println();
    pw.flush();
  }
}
```

```java
// Example 152 from page 129 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.*;

class Example152 {
  public static void main(String[] args) throws IOException {
    final int count = 40000;
    System.out.println("Writing " + count + " strings to random access string array file");
    Iterable<String> strGenerator =
      new Iterable<String>() {
        public Iterator<String> iterator() {
          return new Iterator<String>() {
            int i = 0;
            public boolean hasNext() { return i < count; }

            public String next() { return "string" + i++; }

            public void remove() { }
          };
        }
      };
    writeStrings("saf.dat", strGenerator);
  }

  static void writeStrings(String filename, Iterable<String> strGenerator)
    throws IOException {
    RandomAccessFile raf = new RandomAccessFile(filename, "rw");
    raf.setLength(0);                                   // truncate the file
    ArrayList<Long> offsettable = new ArrayList<Long>();
    for (String s : strGenerator) {
      offsettable.add(raf.getFilePointer());            // store string offset
      raf.writeUTF(s);                                  // write string
    }
    for (long offset : offsettable)
      raf.writeLong(offset);
    raf.writeInt(offsettable.size());                   // write string count
    raf.close();
  }
}
```

```java
// Example 153 from page 129 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.util.Random;

class Example153 {
  public static void main(String[] args) throws IOException {
    final int count = 20;
    System.out.println("Reading " + count + " random strings from string array file");
    Random rnd = new Random();
    for (int i=0; i<count; i++) {
      int j = rnd.nextInt(10000);
      try {
        System.out.println(j + ": " + readOneString("saf.dat", j));
      } catch (IOException e) { throw e; }
    }
  }

  static String readOneString(String filename, int i) throws IOException {
    final int INTSIZE = 4, LONGSIZE = 8;
    RandomAccessFile raf = new RandomAccessFile(filename, "r");
    raf.seek(raf.length() - INTSIZE);
    int N = raf.readInt();
    raf.seek(raf.length() - INTSIZE - LONGSIZE * N + LONGSIZE * i);
    long si = raf.readLong();
    raf.seek(si);
    String s = raf.readUTF();
    raf.close();
    return s;
  }
}
```

```java
// Example 154 from page 131 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

class Example154 {
  public static void main(String[] args) throws IOException {
    if (args.length > 0)
      showDir(0, new File(args[0]));
    else
      showDir(0, new File(".."));
  }

  static void showDir(int indent, File file) throws IOException {
    for (int i=0; i<indent; i++)
      System.out.print('-');
    System.out.println(file.getName());
    if (file.isDirectory()) {
      File[] files = file.listFiles();
      for (int i=0; i<files.length; i++)
        showDir(indent+4, files[i]);
    }
  }
}
```

```java
// Example 155 from page 131 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;

// The producer loop terminates by throwing an exception when the
// reader end of the pipe has died

class Example155 {
  public static void main(String[] args) throws IOException {
    PipedOutputStream outpipe = new PipedOutputStream();
    PipedInputStream inpipe = new PipedInputStream(outpipe);
    final DataOutputStream outds = new DataOutputStream(outpipe);
    DataInputStream inds = new DataInputStream(inpipe);

    // This thread outputs primes to outds -> outpipe -> inpipe -> inds
    class Producer extends Thread {
      public void run() {
        try {
          outds.writeInt(2);
          for (int p=3; true; p+=2) {
            int q=3;
            while (q*q <= p && p%q != 0)
              q+=2;
            if (q*q > p)
              { outds.writeInt(p); System.out.print("."); }
          }
        } catch (IOException e) { System.out.println("<terminated>: " + e); }
      }
    }

    new Producer().start();
    System.out.println("Press Enter for more primes, or ctrl-C to stop");
    System.out.println("Each dot represents the production of one prime");
    for (;;) {                                        // forever
      for (int n=0; n<10; n++)                        //   output 10 primes
        System.out.print(inds.readInt() + " ");       //   and
      System.in.read();                               //   wait for Enter
    }
  }
}
```

```java
// Example 156 from page 133 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.io.*;
import java.net.*;

class Example156 {
  final static int PORT = 2357;

  public static void main(String[] args) throws IOException {
    boolean server = args.length == 1 && args[0].equals("server");
    boolean client = args.length == 2 && args[0].equals("client");
    if (server) {               // Server: accept questions about primality
      ServerSocket serversock = new ServerSocket(PORT);
      for (;;) { // forever, accept connections
        Socket sock = serversock.accept();
        DataInputStream dis  = new DataInputStream(sock.getInputStream());
        DataOutputStream dos = new DataOutputStream(sock.getOutputStream());
        int query = dis.readInt();
        System.out.print("[" + query + "]");
        dos.writeBoolean(isPrime(query));
        dis.close(); dos.close();
      }
    } else if (client) {        // Client: ask questions about primality
      for (int i=1; i<1000; i++) {
        Socket sock = new Socket(args[1], PORT);
        DataOutputStream dos = new DataOutputStream(sock.getOutputStream());
        DataInputStream dis = new DataInputStream(sock.getInputStream());
        dos.writeInt(i);
        if (dis.readBoolean())
          System.out.print(i + " ");
        dos.close(); dis.close();
      }
    } else {                    // Neither server not client
      System.out.println("Start two copies of this program, possibly on different machines:
");
      System.out.println("   java Example156 server");
      System.out.println("   java Example156 client <serverhostname>");
      System.out.println("Use `java Example156 client localhost' if the");
      System.out.println("client and server run on the same machine.");
      System.out.println("You may start several clients all talking to the same server.");
    }
  }

  static boolean isPrime(int p) {
    if (p == 2)
      return true;
    if (p == 1 || p % 2 == 0)
      return false;
    for (int q=3; q*q<=p; q+=2)
      if (p % q == 0)
        return false;
    return true;
  }
}
```

```java
// Example 164 from page 139 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import java.lang.annotation.*;                        // Annotation
import static java.lang.annotation.ElementType.*;      // @Target arguments
import static java.lang.annotation.RetentionPolicy.*;  // @Retention arguments
import java.lang.reflect.*;                            // Method

// Annotation arguments are evaluated at compiletime; they must be
// constant expressions of a limited repertoire of types.

// Annotation objects are constructors are constructed at
// compile-time, and may either be discarded after that, or retained
// in the class file, or retained at run-time.

@Target({TYPE, METHOD})     // Attribute can be used on types and methods only
@Retention(RUNTIME)         // Attribute values are kept until run-time
@interface Author {
  public final int oneHour = 60 * 60 * 1000;
  public String name();
  public Month month();
  public String[] diet() default { "Coffee", "Cola", "Mars bars" };
  public int weeklyWork() default 56 * oneHour;
}

@Retention(RUNTIME)         // Attribute values are kept until run-time
@interface Authors {
  public Author[] value();
}

class Example164 {
  @Author(name="Peter", month=Month.Nov, diet={ "Dr. Pepper" })
  public void myMethod1() { }

  @Author(name="Jens", month=Month.Jul)
  public void myMethod2() { }

  @Authors({@Author(name="Ulrik", month=Month.Jul),
            @Author(name="Andrzej", month=Month.Aug, diet = { "Tea" })})
  public void myMethod3() { }

  public static void main(String[] args) {
    Class ty = Example164.class;
    for (Method mif : ty.getMethods()) {
      if (mif.getName().startsWith("myMethod")) {
        System.out.println("\nGetting the annotations of " + mif.getName());
        // This finds only annotations with RUNTIME retention
        Annotation[] annos = mif.getDeclaredAnnotations();
        System.out.println("The annotations are:");
        for (Annotation anno : annos)
          System.out.println(anno);
      }
    }
  }
}
```

```java
// Example 165 from page 141 of Java Precisely second edition (The MIT Press 2005)
// Author: Peter Sestoft (sestoft@itu.dk)

import static java.lang.System.*;                  // For field out
import static java.lang.Math.*;                    // For methods log, exp
import java.util.*;                                // For class LinkedList<T>

class Example165 {
  public static void main(String[] args) {
    double giga = exp(30.0 * log(2.0));
    out.println("2^30 = " + giga);                 // 2^30 = 1.073741823999999E9
    out.format("2^30 = %12.2f\n", giga);           // 2^30 = 1073741824.00
    String res = concat("Cop", "en", "hagen");
    out.format("result = %s\n", res);              // result = Copenhagen
    Direction dir = Direction.North;
    dir = dir.turnLeft();
    out.println(dir);                              // West
    List<Double> list = new ArrayList<Double>();   // Generic collection type
    list.add(7.2); list.add(22.4); list.add(-9.2); // Boxing to Double
    for (double d : list)                          // Unboxing to double
      out.format("%+7.1f\n", d);                   // Fixed-width formatting
  }

  static String concat(String... ss) {             // Variable-arity method
    StringBuilder sb = new StringBuilder();        // StringBuilder
    for (String s : ss)                            // ss is a String array
      sb.append(s);
    return sb.toString();
  }

  enum Direction {
    East, North, West, South;                      // Enum values

    public Direction turnLeft() {                  // Member method
      switch (this) {                              // Switch on enum value
      case East:  return Direction.North;
      case North: return Direction.West;
      case West:  return Direction.South;
      case South: return Direction.East;
      default: throw new RuntimeException("Impossible");
      }
    }
  }
}
```