1. int h = b.length;
   int k = 0;
   // inv: c[0..k-1] contains the odd elements of
   //      b[h..b.length-1], in reverse order.
   while (h != 0) {
      h = h - 1;
      if (b[h] % 2 == 1) {
         c[k] = b[h];
         k = k + 1;
      }
   }
   // post: c[0..k-1] contains the odd elements of b,
   //       in reverse order.

2. (a) public class NotLowerCaseLetterException
       extends RuntimeException{
      /** Constructor: instance with empty detail message */
      public NotLowerCaseLetterException() {
      }
      /** Constructor: an instance with detail message s */
      public NotLowerCaseLetterException(String s) {
         super(s);
      }
   }
   (b) /** An instance maintains a list of Strings that
       contain only letters in 'a'..'z' */
   public class LettersStringList {
      // The list of Strings. Each String contains only
      // letters in 'a'..'z'
      private Vector<String> list;
      /** Throw a NotLowerCaseLetterException with a
         suitable detail message if s contains a char that is
         not in 'a'..'z'. */
      private static void check(String s) {
         for (int k = 0; k != s.length(); k = k + 1) {
            if (s.charAt(k) < 'a' || s.charAt(k) > 'z') {
               throw new NotLowerCaseLetterException(
                  "Char " + s.charAt(k) + " not in a..z");
            }
         }
      }
      /** Constructor: instance with empty list of Strings. */
      public LettersStringList() {
         list = new Vector<String>();
      }
      /** Append s to the list. If all the letters of s are not
         in 'a'..'z', throw a NotLowerCaseLetterException. */
      public void append(String s) {
         check(s);
         list.add(s);
      }
      /** If s contains only chars in 'a'..'z', append s to the
         list; else print "Mistake in parameter." */
      public void appendMessage(String s) {
         try {
            append(s);
         } catch (NotLowerCaseLetterException e) {
            System.out.println("Mistake in parameter.");
         }
      }
   }

3. (a) Make a class abstract so that instances of it can not
    be created. To make the class abstract, insert keyword
    abstract between public and class.
   (b) Evaluate expression e and store its value in variable x.
   (c) 
      meth1: 1
      p
      a1
      a0
   Argument of first call: a0
   Argument of second call: a1

4. Power of Now: 28
   Item@58d6b0 [this is the one given by DrJava]
   Truth Wins Out: 10
   Cost of Item: 78
   Book is: 10
   Book is: Power of Now
   Are books the same? false
   Are books the same? false

5. /** = if s is "", 0;
      otherwise: the number of times the first
      char of s appears at the beginning of s.
      E.g. for s = "xxxxxxxxxx", the answer is 3. */
   public static int numberOfFirst(String s) {
      if (s.length() == 0)
         return 0;
      if (s.length() == 1 || s.charAt(0) != s.charAt(1))
         return 1;
      return 1 + numberOfFirst(s.substring(1));
   }
   /** = the compression of s. */
   public static String compress(String s) {
      if (s.length() == 0)
         return "";
      if (s.length() == 1 || s.charAt(0) != s.charAt(1))
         return 1;
      return 1 + numberOfFirst(s.substring(1));
   }

6. Note: In this answer, we omit the conventional getter
   methods for fields. If you omitted them too, that is OK.
   /** An instance contains info about a Passport */
   public class Passport {

/** number of passports created thus far */
private static int numberOfPassports = 0;
/** Contains all Passports ever created */
private static Vector v = new Vector<Passport>();
private String p; /** The name of the person */
private String s; /** State in which p lives (two char state designation, like NY */
private int n; /** p's passport number */
/** = a Passport for person p (null if there is none). */
public Passport getPassport(String p) {}

/** Constructor: an instance for person p with passport number n. The person lives in state s.
Precondition. No passport exists for p.
Precondition: s is a two-char state design., e.g. NY */
private Passport(String p, String s, String n) {}
/** = the existing Passport for person p in state s, or, if there is none, a newly created one for p in state s.
Precondition: s is a two-char state design., e.g. NY */
public Passport assign(String p, String s) {}
/** = representation of the Passport */
public String toString() {
}

7. (a) A JPanel is associated with a FlowLayout manager. It lays out the components in a horizontal row, in the order in which they were added to the JPanel. However, if the window is too narrow to contain all the components, they flow into successive rows.

(b) The value of keyword this is the name (on the tab of) the object in which it appears. So, if the call

```
   JButton.addActionListener(this);
```

appears in an object named A0, the value of the argument in the call is A0.

c)

```
   b   a1

   a1
0 a2 0
1 a9
       2
       4
       6
       7
```

8. Selection sort. We give assertions as formulas; you can translate them easily into diagrams.

/** Sort b[p..q] (so the postcondition is: b[p..q] is sorted) */
public static void sort(int[] b, int p, int q) {
    int k = p;
    // inv: b[p..k-1] is sorted and
    // b[p..k-1] <= b[k..q]
    while (k < q) {
        int m = index of min of b[k..q];
        Swap b[k] and b[m];
        k = k + 1;
    }
    // post: b[p..q] is sorted