Your answers may be different! That's OK, as long as they are correct!

Question 1. (a)

```
/** An instance is ... */
public class OutOfSpaceException
                        extends RuntimeException {
   /** Constructor: an instance with message m*/
   public OutOfSpaceException(String m) {
      super(m);
   }
   /** Constructor: an instance with no message */
   public OutOfSpaceException() {
      super();
   }
}
```

(b)
```
/** A Shelf of books */
public class Shelf extends Repository {
   private Vector<Book> contents; // books on the shelf
    //                 size is <= the capacity of this repository

   /** Constr: book shelf with capacity c and no books */
   public Shelf(int c) {
      super(c);
      contents= new Vector<Book>();
   }

   /** = no. of books on this shelf */
   public int noItems() {
      return contents.size();
   }

   /** Add book b to this shelf and return its location.
      Throw OutOfSpaceException if there is no room*/
   public int add(Book b) {
      if (contents.size() == capacity()) {
         throw new OutOfSpaceException();
      }
      contents.add(b);
      return contents.size()-1;
   }

   /** = the book at location i of this Shelf.
         (null if i is not the location of a book) */
   public Book get(int i) {
      if (i < 0 || contents.size() <= i)
          return null;
      return contents.get(i);
   }
}
```

Question 2. (a)
```
/** A book that is also on the web */
public class WebBook extends Book {
   private String url; // Where the book is on the web

   /** A book b on the web at URL url.
      Throw an IllegalArgumentException if ... */
   public WebBook(Object b, String url) {
      super(b);
      this.url= url;
      if (!(url.endsWith(".pdf")))
         throw new IllegalArgumentException(
                        "url does not end in pdf: " + url);
   }

   /** = the url for this book */
   public String getUrl() {
      return url;
   }
}
```

(b)
```
/** = the url for b (null if it doesn't have one) */
    public static String url(Book b) {
        if (b instanceof WebBook) {
           return ((WebBook)b).getUrl();
        }
        return null;
    }
```

Question 3.
```
int t= h;
int j= k;
int i= k+1;
// inv: b[h..t-1] < 0, b[t..i-1] unknown,
//      b[i..j] = 0, and b[j+1..k] > 0
while (t < i) {
   if (b[i-1] < 0) {
      Swap b[i-1] and b[t];    t= t+1;
   } else if (b[i-1] == 0) {
      i= i-1;
   } else {
      Swap b[i-1] and b[j];   i= i-1;   j= j-1;
   }
}
return new Point(i, j);
```

Question 4. Below are two nice solutions. In one, at each iteration of the main loop, all elements of one row are created; in the other, all elements with the same label are created. We don't care which you did.


```
/** = the bricks placed in the
GUI. See final for spec. */
public Brick[][] placeBricks1(int m) {
    Brick[][] b= new Brick[m][m];
    // inv: bricks on rows 0..r-1 have been created
           and added to the GUI
    for (int r= 0; r < m; r= r+1) {
        // Create and add bricks b[r][0..m-1]
        for (int c= 0; c < m; c= c+1){
           int lab= c < r ? r : c; //label on brick b[r][c]
           Color col=
                 (lab%2 == 0 ? Color.red : Color.green);
           b[r][c]= new Brick(20*r, 20*c, 20, col, lab);
           add(b[r][c]);
        }
    }
    return b;
}
```

```
/**= the bricks placed in the GUI. See final for spec. */
public Brick[][] placeBricks2(int m) {
     Brick[][] b= new Brick[m][m];
     // inv: bricks with a label in 0..n-1 have been
             created and added to the GUI
    for (int n= 0; n < m; n= n+1) {
         // Create/add bricks with label n
         // Create/add bricks b[n][0..n]
         Color col= n%2 == 0 ? Color.red : Color.green;
         for (int c= 0; c <= n; c= c+1) {
             b[n][c]= new Brick(20*n, 20*c, 20, col, n);
             add(b[n][c]);
         }

         // Create/add bricks b[0..n-1][n]
         for (int c= 0; c < n; c= c+1) {
             b[c][n]= new Brick(20*c, 20*n, 20, col, n);
             add(b[c][n]);
         }
   }
    return b;
}
```

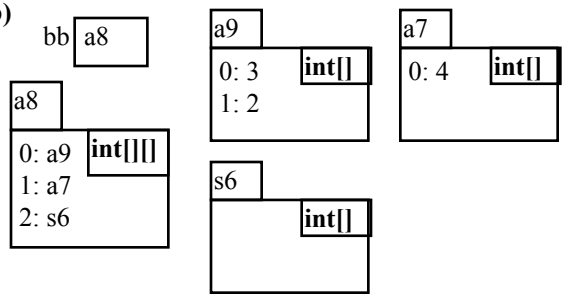
Question 5.

```
/** = weight of heaviest Rhino in r's ancestry tree.
      (if r is null, use 0 as the weight) */
public static double heaviest(Rhino r) {
   if (r == null)
       return 0;
   return Math.max(r.weight,
      Math.max(heaviest(r.mother), heaviest(r.father)));
}

/** = the heaviest Rhino in this rhino's ancestry tree. */
public Rhino heaviest() {
    Rhino h= this;
    if (mother != null) {
        Rhino m= mother.heaviest();
        if (h.weight < m.weight) h= m;
    }
    if (father != null) {
        Rhino f= father. heaviest();
        if (h.weight < f.weight) h= f;
    }
    return h;
}
```

Question 6 (a) Evaluation the expression; if it is true, execute the statement and then repeat the process (if the expression is false, there is nothing more to do; execution of the while-statement is done).

(b)



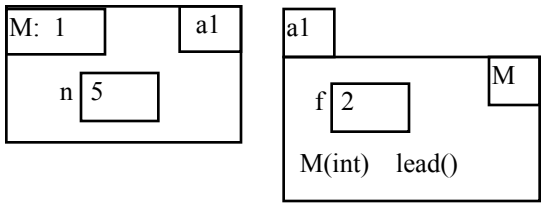
(c)

```
/** = number of elements in b */
public static int size(Object[] b) {
    int k= 0;
    try {
        while (true) {
            Object x= b[k];
            k= k+1;
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        return k;
    }
}
```

Question 7. (a)

```
public void testMConstructor() {
     M m= new M(4);
     assertEquals(4, m.f);
     m= new M(5);
     assertEquals(2, m.f);
}
```

(b) The value of the expression is `a1`.



(c)