CS100J Spring 2008. Answers to final

1 .
```
int k= b.length-2;
int big1= Math.max(b[k], b[k+1]);
int big2= Math.min(b[k], b[k+1]);
// inv: big1 is the largest int in b[k..b.length–1],
//      big2 is second largest int in b[k..b.length–1],
//          0 <= k <= b.length-2
while (k != 0) {
    k= k-1;
    if (b[k] > big1)
       { big2= big1; big1= b[k]; }
    else if (b[k] > big2)
       { big2= b[k]; }
  }
```

2.
```
public class OutOfSpaceException extends
                                  RuntimeException {
    /** Constructor: Instance with no detail message */
    public OutOfSpaceException()
       { super(); }

    /** Constructor: Instance with detail message m */
    public OutOfSpaceException(String m)
       { super(m); }
}

/** append v to the list, but
    Throw an OutOfSpaceException if there is no space */
public void add(int v) throws OutOfSpaceException {
    if (n == list.length)
       { throw new OutOfSpaceException();       }
    list[n]= v;
    n= n+1;
}

/** If there is room in the list for v, then append it.
    Otherwise print error message "no space" */
public void messageAdd(int v) {
    try { add(v); }
    catch (OutOfSpaceException e) {
       System.out.println("no space");
    }
}
```

3. (a)
```
/** An instance is a time of day */
public class Time implements Komparable {
  private int hr;  // The hour of the day, in range 0..23
  private int min; // The minute of the hours, 0..59

  /** Constructor: Time of day given in minutes m,
                   n range 0..60*24-1 */
  public Time(int m)
     { hr= m/60; min= m%60; }

  /** = "This object comes before t".
      Throw a ClassCastException … */
  public boolean less(Object t) {
     if (!(t instanceof Time))
        throw new ClassCastException();
     Time tm= (Time) t;
     return hr < tm.hr ||
          (hr == tm.hr && min < tm.min);
  }
}
```

3.b. Komparable[] b= {**new** Time(60), **new** Time(121)};

4.
```
Robin - 2008.05.08 : 9.0
Sound@14a3c6
Starling - 2007.01.01 : 10.0
Length of sound: 59.0
Bird is: Robin
Bird is: Robin
Birds are the same?false
Birds are the same?false
```

5.
```
/** = the complement of n, formed by replacing
      each decimal digit of n by 10-n.

      Precondition: n > 0 and no digit of n is 0 */
public static int complement(int n) {
  if (n < 10)
     return 10 - n;
  return complement(n/10) * 10 + (10 - n%10);
}
```

6.
```
import java.util.*;

/** An instance is a birth day */
public class BirthDay {
  /** a list of all Birthday objects */
  public static Vector<BirthDay> birthdays=
                                  new Vector<BirthDay>();

  private int month; // month of the birthday
  private int day;   // day of the month of the birthday

  /** Constructor: instance with month m and day d.
      Pre: this is a valid date: m in 1..12 and d in 1..30
           except when m = 2, when d in 1..28. */
  public BirthDay(int m, int d) {
    month= m; day= d;
    birthdays.add(this);

  }
  /** Constructor: nstance with the last day of month m.
    Precondition: this is a valid date: m in 1..12. */
  public BirthDay(int m) {
    this(m, m != 2 ? 30 : 28);
  }

  /** = the month */
  public int getMonth() {  return month;  }

  /** = the day */
  public int getDay() { return day; }

  /** = the birthday, in the form month:day */
```

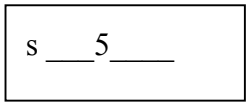
```
   public String toString()
      { return month + "." + day; }

/** An instance is a birthday shifted a bit */
public class VirtualBirthday extends BirthDay {
   private int shift; // The virtual birthday is the real
                      // birthday shifted shift days

   /** Const: instance with birth day m.d, shifted s days.
       Pre: s is  -10..10 and the real and virtual birthdays
            are in the same year.*/
   public VirtualBirthday(int m, int d, int s) {
      super(m, d);
      shift= s;
   }

   /** = the virtual birthday, in the form month:day */
   public String toString() {
      int month= getMonth();
      int day= getDay() + shift;
      if (day > 30) {
          month= month + 1; day= day - 30;
      }
      if (day < 1) {
         month= month - 1; day= 30 + day;
      }
      return month + "." + day;
   }
}
```

7a. Test1
Test2
Test3

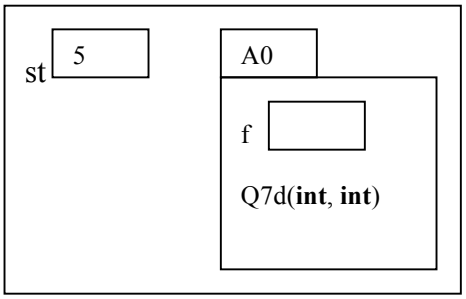
7b. The layout manager is a FlowLayoutManager. The components appear in the JPanel in the order in which they were added, but they flow to the next line(s) if the JPanel is too narrow.

7c. Keyword **this** refers to the (name of) the object in which it appears.

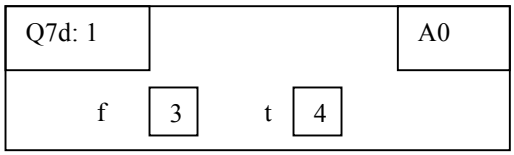
7d.



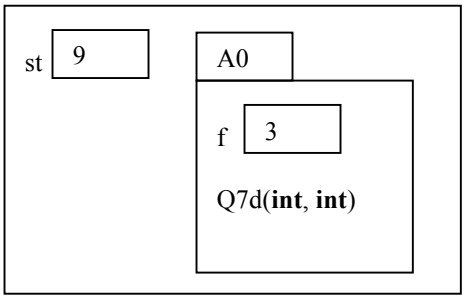
Step 1. Draw a new object of class Odd (it goes in Odd's file drawer):



Step 2. Execute the constructor call Oddd(3, 4). The frame for the call just after it is drawn is this:



Executing the call changes object A0 and static variable st so that everything looks like this:



Step 3. Yield as value of the new-expression the name of the newly created object, in this case, A0.

8. Algorithm bsearch. We give assertions as formulas; you can translate them easily into diagrams.

```
/** = a value j that satisfies b[0..j] <= v < b[j+1..].
      Precondition: b is sorted. */
public static int bsearch(int[] b, int v) {
      int j= -1; int k= b.length;

      // inv: b[0..j] <= v < b[k..b.length-1]
      while (j != k-1) {
         int e= (j+k)/2;
         if (b[e] <= v) j= e;
         else k= e;
      }
      return j;
   }
```