CS 201, Fall 2014 — Oct 3rd — Exam 1 Name: Solution

Question 1. [10 points] Consider the following class (which begins on the left and continues on the right):

```
public class Box {
  private int val;

  public Box(int v)
    { this.val = v; }
```

```
  public int getVal()
    { return this.val; }

  public void setVal(int v)
    { this.val = v; }
}
```

(a) Consider the following statements:

```
Box b = new Box(44);
missing code
System.out.println(b.getVal());
```

What code can be substituted for *missing code* so that the code prints the output 55?

b.setVal(55);

(b) Consider the following statements (which could occur in any class, and not necessarily in the Box class):

```
Box b = new Box(121);
System.out.printf("Value is %d\n", b.val);
```

Does this code compile? If not, explain why not. If it does compile, what output does it produce?

It will not compile (if outside of the Box class) because the val field is private, and can't be accessed directly by methods that aren't in the Box class.

Question 2. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q2 {
  private int val;

  public Q2(int v)
    { this.val = v; }

  public void setVal(int v)
    { this.val = v; }

  public int getVal()
    { return this.val; }
```

```
  public static void main(String[] a) {
    Q2 obj1 = new Q2(4);
    Q2 obj2 = new Q2(5);

    System.out.printf("%d,%d\n",
      obj1.getVal(), obj2.getVal());

    obj2 = obj1;  ← make obj2 point to same object as obj1
    obj2.setVal(7);

    System.out.printf("%d,%d\n",
      obj1.getVal(), obj2.getVal());
  }
}
```

$\boxed{\begin{matrix}4,5\\7,7\end{matrix}}$

Question 3. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

a will point to same array that arr points to

```
public class Q3 {
  public static void f(int[] a) {
    int[] b;
    b = a;  ← make b point to same array a points to
    b[0] = 55;
  }
```

```
  public static void main(String[] a) {
    int[] arr = new int[2];
    arr[0] = 11;
    arr[1] = 22;
    f(arr);
    System.out.printf("%d,%d\n",
      arr[0], arr[1]);
  }
}
```

$\boxed{55,22}$

Question 4. [10 points] Consider the following method:

```
public static String getLongestLine(String fileName) throws IOException {
  String longest = null;
  FileReader fr = new FileReader(fileName);
  BufferedReader br = new BufferedReader(fr);
try {
  boolean done = false;
  while (!done) {
    String line = br.readLine();
    if (line == null) {
      done = true;
    } else if (longest == null || line.length() > longest.length()) {
      longest = line;
    }
  }
} finally {
  br.close();
}
  return longest;
}
```

This method is intended to return the longest line in a text file as a `String`.

(a) It is possible for this method to open the named file but make no attempt to close it. Briefly explain how this could happen.

If an IOException is thrown by the call to readLine(), br.close() will never be reached.

(b) Show how to modify the method so that if the file is opened, it is guaranteed to make an attempt to close the file. (Suggestion: annotate the method showing where to add or modify code.)

See above.

Question 5. [15 points] Write a class called `TrafficLight` to simulate a traffic light. An object which is an instance of the `TrafficLight` class should have three "states": green, yellow, and red. The object should start in the green state. When the `nextState()` method is called, it should switch to the next state in the sequence (green to yellow, yellow to red, or red back to green.) The `toString()` method should return a string describing the current state.

Example JUnit test showing the expected behavior:

```
public void testTrafficLight() {
  TrafficLight trafficLight = new TrafficLight();

  assertEquals("green", trafficLight.toString());
  trafficLight.nextState();
  assertEquals("yellow", trafficLight.toString());
  trafficLight.nextState();
  assertEquals("red", trafficLight.toString());
  trafficLight.nextState();
  assertEquals("green", trafficLight.toString());
}
```

```
public class TrafficLight {
    private int state ();

    public TrafficLight() {
        state = 0;
    }

    public void nextState () {
        state = (state + 1) % 3;
    }

    public String toString () {
        if (state == 0) {
            return "green";
        } else if (state == 1) {
            return "yellow";
        } else {
            return "red";
        }
    }
}
```

Question 6. [15 points] Consider the following partially-specified class (which begins on the left and continues on the right):

```
public class Name implements
    Comparable<Name> {

  private char firstInitial;
  private String lastName;

  public Name(char fi, String ln) {
    firstInitial = fi;
    lastName = ln;
  }
```

```
  public char getFirstInitial()
   { return firstInitial; }

  public String getLastName()
    { return lastName; }

  public int compareTo(Name o) {
    TODO
  }
}
```

Show the code that should replace TODO to implement the `compareTo` method. This method should compare first by last name, then by first initial (in the case where the last names are the same.) Example JUnit test:

```
Name tGreen = new Name('T', "Green");
Name jSmith = new Name('J', "Smith");
Name rSmith = new Name('R', "Smith");

assertTrue(jSmith.compareTo(tGreen) > 0);
assertTrue(jSmith.compareTo(rSmith) < 0);
```

Hint: the `String` class implements the `Comparable` interface.

```
int  cmp;
cmp =  this.lastName.compareTo(o.lastName);
if (cmp != 0) {
    return cmp;
}
if (this.firstInitial < o.firstInitial) {
    return -1;
} else if ( this. firstInitial > o.firstInitial) {
    return 1;
} else {
    return 0;
}
```