CS 201, Fall 2015 — Oct 2nd — Exam 1 Name: Solution

Question 1. [5 points] What output is printed by the following code (which begins on the left and continues on the right)?

```
public class Q1 {
  private int value;

  public Q1(int v) {
    value = v;
  }

  public static void main(
      String[] args) {
    Q1 a = new Q1(17);
    Q1 b = new Q1(42);
    System.out.printf("%d,%d\n",
      a.value, b.value);

    a = b;
    System.out.printf("%d,%d\n",
      a.value, b.value);
  }
}
```

17, 42
42, 42

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

```
public class Q2 {
  private int value;

  public Q2(int v) {
    value = v;
  }

  public static void mystery(
      Q2 x, Q2 y) {
    Q2 tmp = x;
    x = y;
    y = tmp;
  }

  public static void main(
      String[] args) {
    Q2 a = new Q2(17);
    Q2 b = new Q2(42);

    mystery(a, b);

    System.out.printf("%d,%d\n",
      a.value, b.value);
  }

}
```

assignments to x and y parameters do not affect a and b in main

17, 42

Question 3. [10 points] Write statements to read two sets of (x,y) coordinates from the user, calculate the distance between the two sets of coordinates, and print the result. The distance between two points is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. You can use `Math.sqrt` to compute the square root of a `double` value.

```
double x1, y1, x2, y2;
Scanner keyboard = new Scanner(System.in);
x1 = keyboard.nextDouble();
y1 = keyboard.nextDouble();
x2 = keyboard.nextDouble();
y2 = keyboard.nextDouble();
double xdiff = x2 - x1, ydiff = y2 - y1;
double dist = Math.sqrt(xdiff * xdiff + ydiff * ydiff);
System.out.println(dist);
```

Question 4. [10 points] Identify, explain, and correct all of the problems with the following class definition, which begins on the left and continues on the right.

```
public class Point {
   double x;
   double y;

   Point() {
     x = 0;
     y = 0;
   }

   Point(x,y) {
     x = x;
     y = y;
   }
```

```
private getX() {
  return(x);
}

private getY() {
  return(y);
}
}
```

instance fields should be private

constructors are usually public

need datatypes (double)

need return type (double)

need to disambiguate params vs. fields

```
this.x = x;
this.y = y;
```

Question 5. [10 points] Consider the following method:

```
       public static int countLines(String fileName) {
1        FileReader fr = new FileReader(fileName);
2        BufferedReader br = new BufferedReader(fr);
         try {
3        int count = 0;
4        while (true) {
5          String line = br.readLine();
6          if (line == null) { break; }
7          count++;
8        }
         } finally {
9        br.close();
         }
10       return count;
       }
```

(a) At which lines in this method could an `IOException` or `FileNotFoundException` occur?

1, 5, and 9

(b) As written, this method neither declares itself as throwing `IOException` nor attempts to handle `IOException` using `try/catch`. Which of these do you think would make more sense? Explain briefly.

The method should throw IOException. try/catch only makes sense if it is possible to do something to recover from the exception.

(c) Explain how the method might open a file, but then not make an attempt to close it.

br.readLine() could throw an IOException

(d) Explain how to guarantee that the method will make an attempt to close the file if it is opened.

add try/finally as shown above

Question 6. [10 points] Implement the `CountBy` class so that the following JUnit tests will pass. The idea is that when a `CountBy` object is created, its internal counter is set to zero, and a parameter to the constructor sets an integer value that is the multiple by which the object will count. Each time the `next` method is called, the internal counter should be increased by the multiple. Each time the `get` method is called, it should return the object's current counter value.

```
CountBy by2 = new CountBy(2);
CountBy by7 = new CountBy(7);

assertEquals(0, by2.get());
assertEquals(0, by7.get());

by2.next();
assertEquals(2, by2.get());

by7.next();
by7.next();
assertEquals(14, by7.get());
```

```
public class CountBy {
    private int incr, count;

    public CountBy(int incr) {
        this.incr = incr;
        this.count = 0;
    }

    public int get() {
        return count;
    }

    public void next() {
        count += incr;
    }
}
```

Question 7. [10 points] For the `Point3D` class shown below, circle the fields and methods that a concrete class `Vector` extending `Point3D` would inherit from `Point3D`.

Also, list all the methods that subclass `Vector` has to instantiate to be considered a valid concrete class.

```
public abstract class Point3D {
  public    int x;
  public    int y;
  public    int z;

  public A() {
    x = 0;
    y = 0;
    x = 0;
  }

  public A(int x, int y, int z) {
    this.x = x;
    this.y = y;
    this.z = z;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
  }

  public int getZ() {
    return z:
  }
```

inherited
by Vector
(except constructors —
subclasses must
define their
own constructor(s))

```
  private int getX_2() {
    return x*x;
  }

  private int getY_2() {
    return y*y;
  }

  private int getZ_2() {
    return z*z;
  }
```

private methods
aren't inherited

```
  public abstract double getLength();

  public abstract double getNormX();

  public abstract double getNormY();

  public abstract double getNormZ();
}

public class Vector
    extends Point3D {
  // what methods must
  // this class define?
}
```

Vector must
implement these
concretely.