2016 SI@UCF Java GUI Test #1 Redo Test #1

Name: SOLUTION
6/21/16

1) (15 pts) Assuming all of the support code is set up, complete the draw method below so that it prints 10 red letter X's side by side using the drawLine method only. The four corners of the first X should be at (0, 50), (50, 50), (0, 150) and (50, 150). The two right corners of the i^{th} X should serve as the left corners of the $(i+1)^{st}$ X.

```
public void draw(Graphics g) {

    for (int i=0; i<10; i++) {              // 3 pts
        g.drawLine(50*i, 50, 50+50*i, 150); // 6 pts
        g.drawLine(50*i,150, 50+50*i, 50);  // 6 pts
    }
}
```

2) (5 pts) Describe, in English, what the following code segment accomplishes. Note: The prompts for entering information are intentionally vague, so they don't give away the purpose of this segment of code.

```
Scanner stdin = new Scanner(System.in);
int x1, y1, x2, y2;

System.out.println("Enter...");
x1 = stdin.nextInt();
y1 = stdin.nextInt();
System.out.println("Enter...");
x2 = stdin.nextInt();
y2 = stdin.nextInt();
int d = Math.abs(x1 - x2) + Math.abs(y1 - y2);
System.out.println("Answer = "+d);
```

Takes input from the user in the form of four integers, which correspond to two points. It then calculates the Manhattan distance, or distance travelled between them if you can only move vertically and horizontally.

3) (15 pts) A simple formula for the height of an object dropped from Y feet high t seconds after its dropped (on Earth) is $f(t) = Y - 16t^2$. Fill in the program below so that it prints out a chart of the height of an object after each second, starting with t = 0. The user will enter the initial height. The last line should print a height of 0, regardless of what the mathematical equation says, since the object won't fall below the ground.

```
public class Drop {

    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        int height;

        System.out.println("Enter the initial height.");
        height = stdin.nextInt();

        int initialHeight = height;                    (5 pts)

        int t = 0;

        System.out.println("Time\tHeight");
        while (height > 0) {



            System.out.println(t+"\t"+height);

            t++;
            Height = initialHeight - 16*t*t;           (5 pts)



        }

     if (height<0)                                     (5 pts)
          height = 0;
     System.out.println(t+"\t"+height);



    }
}
```

4) (10 pts) What is the output of the following code segment?

```
int a=3, b=1;
while (a < 8) {
  b = b + a;
  a = a + 1;
  System.out.println("a = "+a+" b = "+b);
}
```

a = 4 b = 4 (2 pts)
a = 5 b = 8 (2 pts)
a = 6 b = 13 (2 pts)
a = 7 b = 19 (2 pts)
a = 8 b = 26 (2 pts)

Consider creating a class to store a "spinner" object. A spinner is wheel that can be spun with several different labeled wedges, from 0 to n-1, where n represents the number of sectors on the spinner. An incomplete version of the class is included below. The next five problems (5 – 9) will be to add methods to this class.

```
public class Spinner {

  private int numSectors;
  private Random r;

  public Spinner(int n) {
    numSectors = n;
    r = new Random();
  }

}
```

5) (10 pts) Add a default constructor to the Spinner class that creates a Random object, and then uses it to set the number of sectors for the Spinner to a random integer in between 1 and 10, inclusive.

```
public Spinner() {

      r = new Random();                    (5 pts)
      numSectors = r.nextInt(10) + 1;      (5 pts)




}
```

6) (5 pts) Add a method double to the Spinner class that doubles the number of sectors on the spinner.

```
public void double() {

      numSectors *=2;                      (5 pts)


}
```

7) (10 pts) Add a method compareTo to the Spinner class so that it returns -1 if this spinner has fewer sectors than other, 0 if both Spinners have an equal number of sectors, and 1 if this spinner has more sectors than other.

```
public int compareTo(Spinner other) {

      if (this.numSectors > other.numSectors)          (3 pts)
            return 1;
      else if (this.numSectors == other.numSectors)    (3 pts)
            return 0;
      else                                             (4 pts)
            return -1;
}
```

8) (5 pts) Add a method spin to the Spinner class that returns a random sector of the spinner, in between 0 and numSectors-1.

```
public int spin() {

      return r.nextInt(numSectors);                    (5 pts)

}
```

9) (15 pts) Write a method spinsUntil that takes in a single integer, x, and simulates spinning the spinner until that number, x, is spun. If x < 0 or x >= numSectors, the method should not do a simulation and return -1 to indicate the impossibility of the task. Otherwise, the method should run the simulation using the spin method and return how many times the spinner needed to be spun to land on x. The minimum value it should return in these cases is 1, if we get lucky an land on the value x the very first time.

```
public int spinsUntil(int x) {

      if (x<0 || x>=numSectors)            (5 pts)
            return -1;


      int count = 0;

      while(this.spin()!= x)               (10 pts)
            count++

      return count;









}
```

10) (5 pts) Write a single assignment statement that "implements" the Distance Formula in Java to find the straight line distance between the points (x1, y1) and (x2, y2).

The formula is $D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$. Assume that variables x1, y1, x2, y2 and d have all been declared and the first four store meaningful values. Write a single statement that assigns d to the distance between the points represent by (x1, y1) and (x2, y2).

```
D = Math.sqrt( Math.pow(x2-x1,2) + Math.pow(y2-y1,2) )  (5 pts)
```

10) (5 pts) In what shape are the individual pieces of the popular cereal Froot Loops?

Loops