SI@UCF Java GUI Test #1 Solutions
7/10/2015

1) (12 pts) Jimmy's box for crayons has dimensions `L x W x H`. Each crayon he puts in has a height of exactly `H`. Thus, he stands each crayon up in the box forming rows of crayons, as if the cross section of each crayon was a square of side length `S`. (Imagine filling a rectangle of dimensions `L x W` with squares with dimensions `S x S`.) Complete the program below so that it correctly calculates the maximum number of crayons that can fit in Jimmy's box.

```
public class crayons {

    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        System.out.println("What is the length of your box?");
        int length = stdin.nextInt() ;
        System.out.println("What is the width of your box?")
        int width = stdin.nextInt() ;
        System.out.println("What is the diameter of each crayon?")
        int side = stdin.nextInt() ;

        int lengthCrayon = length/side;
        int widthCrayon = width/side;
        int numCrayons = lengthCrayon*widthCrayon;

        System.out.println(numCrayons+ "crayons will fit in the box.")
    }
}
```

Grading: 3 pts all reads, 3 pts both int divs, 2 pts mult, 1 pt print

2) (10 pts) Complete the segment of code below so that it reads in 10 lowercase alphabetic strings and prints out the shortest of those strings. If there is a tie, print the string that comes first alphabetically, of the shortest strings.

```
Scanner stdin = new Scanner(System.in);
String best = stdin.next();

for (int i=1; i<10; i++) {

    String cur = stdin.next();                  // 1 pt
    if (cur.length() < best.length())           // 3 pts
        best = cur;                             // 1 pt
    else if (cur.length() == best.length() &&   // 4 pts
             cur.compareTo(best) < 0)
        best = cur;                             // 1 pt
}
System.out.println("The winner is "+best);
```

3) (20 pts) The Florida State Lottery has come up with a very strange system for awarding prizes, in hopes of getting mathematics enthusiasts to play. (Math enthusiasts normally stay away from lotteries, since they understand the idea of expected value.)The player chooses any integer, ***n***, from 1 to 10^9 and a second random integer, ***modval***, from 10 to 99 is chosen by the computer (after ***n*** has been selected). If the sum of the squares of the digits of ***n*** is divisible by ***modval***, then you win! Complete the program below to simulate playing this game once. If the player wins, your program should print out **"You win!"**, otherwise it should print **"You lose."**

```
public class Lotto {

    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        Random r = new Random();

        System.out.println("What value of n do you choose?\n");
        int n = stdin.nextInt();


        int modval = 10 + r.nextInt(90);     // 3 pts

        int digitSqSum = 0;                  // 1 pt
        while (n > 0) {                      // 2 pts
            int digit = n%10;                // 3 pts
            digitSqSum += (digit*digit);     // 3 pts
            n /= 10;                         // 3 pts
        }

        if (digitSqSum%modval == 0)          // 2 pts
            System.out.println("You win!");  // 1 pt
        else                                 // 1 pt
            System.out.println("You lose."); // 1 pt









    }
}
```

Consider creating a class to store a lego brick. Each lego brick has a width and a length (both positive integers) and we stipulate that the width be no greater than the length. We define the size of a lego brick to be the product of its length and width. A lego brick fits on top of another lego brick if at least both bricks have one equal dimension and the other dimension of the first brick is less than or equal to the other dimension of the second brick. For example, a 3x3 brick fits on a 3x4 brick, and a 2x4 brick fits on a 1x4 brick, but a 3x5 brick does not fit on a 3x4 brick and a 1x2 brick doesn't fit on a 2x3 brick. To lego bricks are equal if and only if both dimensions match. According to this specification, complete the LegoBrick class below.

```
public class LegoBrick {
  private int width;
  private int length;

  // Precondition: dim1 > 0 and dim2 > 0
  // Postcondition: LegoBrick is created with width <= length and
  //                matching dim1 and dim2.
  public LegoBrick(int dim1, int dim2) {
  }

  // Postcondition: Returns true iff this/current LegoBrick fits
  //                on top of other.
  public boolean fitsOn(LegoBrick other) {
  }

  // Postcondition: returns true iff this equals other.
  public boolean equals(LegoBrick other) {
  }

  // Returns the size of this LegoBrick.
  public int size() {
  }

  public String toString() {
    return width+"x"+length;
  }
}
```

4) (10 pts) Write a constructor for the LegoBrick class that creates a LegoBrick object with the dimensions given in the two input parameters. Note that the input parameters may be given in any order. (Thus, if the input parameters were 5 and 3, then the width must be assigned to 3 and the length to 5.)

```
public LegoBrick(int dim1, int dim2) {
    width = Math.min(dim1, dim2); // 5 pts for each case.
    length = Math.max(dim1, dim2);
}
```

5) (10 pts) Add a method fitsOn that returns true iff this LegoBrick fits on top of LegoBrick other, as previously defined.

```
public boolean fitsOn(LegoBrick other) {

    if (this.width == other.width && this.length <= other.length)
        return true;  // 3 pts this case
    if (this.length == other.length && this.width <= other.width)
        return true;  // 3 pts this case
    if (other.width == this.length)
        return true;  // 3 pts this case
    return false;     // 1 pt fall through

}
```

6) (5 pts) Add a method equals to the LegoBrick class that returns true iff this LegoBrick equals other.

```
public boolean equals(LegoBrick other) {

    return this.width == other.width && this.length == other.length;
    // 1 pt return, 2 pts and, 1 pt each equality
}
```

7) (5 pts) Add a method size to the LegoBrick class that returns the size of this LegoBrick.

```
public int size() {
    return this.width*this.length;
    // 1 pt return, 2 pts mult, 1 pt each component
}
```

8) (12 pts) Complete the draw method shown below so that as long as it's properly invoked with all the appropriate support code, it will draw a green snowman shaped figure of three circles with radii 100, 60 and 20, respectively, from bottom to top.

```
public void draw(Graphics g) {

    g.setColor(this.backgroundColor);          // 0 pts
    g.fillRect(0, 0, 500, 500);                // 0 pts

    g.setColor(Color.GREEN);                   // 1 pt
    g.fillOval(150, 250, 200, 200);            // 3 pts
    g.fillOval(190, 130, 120, 120);            // 3 pts
    g.fillOval(230, 90, 40, 40);               // 3 pts

// Note - many coordinates are possible. The last two coordinates for each
// method call must match exactly. The first two coordinates can be different
// but the differences between them must match the differences above
// (+40, +40 for the first parameter and -80, -40 for the second parameter.)


}
```

9) (12 pts) In the posted solution for the pong game the following segment of code appears in the update method:

```
if (keyPressed[1]) {
    if (magY+magHeight < 500)
        magY += 5;
}
```

(a) If the inner if were removed, what may occur to the Magenta paddle?

It would potentially scroll off the bottom of the screen, since the maximum y value of the paddle would be unbounded without that inner if. (4 pts)

(b) What is the purpose of adding magHeight to magY instead of just comparing magY to 500? (Note: 500 is the height of the window.)

magY represents the y value of the top left corner of the rectangular paddle, when we add the height to this value, we get the y value of the bottom left corner of the paddle. We don't want the paddle to go off the screen at all, which means preventing *the bottom of the panel* from exceeding the maximum Y value of 500. (5 pts)

(c) What direction (up, right, down, left) does keyPressed[1] correspond to?

Down (3 pts, all or nothing)

10) (4 pts) What color are Red Hot candies? **Red Hot (4 pts give to all)**