CMPS 1600 Introduction to Computer Science II – Spring 14

11/17/14

1. Homework

Programming portion (problems 1 and 2) due **Thursday 1/23/14** at 11:55pm on Blackboard.

Written portion (problem 3) due **Friday 1/24/14** at the beginning of class.

Please zip the (Eclipse) project directory for this homework, and use the following naming convention for the name of the project (and directory): `lastName_firstName_hw1`. **In order to receive any credit for the programming portions, you are required to thoroughly comment and test your code.**

1. **Methods (8 points)**
   Write a class called `Methods` that contains the following static methods, and tests them in the `main` method of this class.

   (a) (2 points) Write a `private` static method that takes an integer array as argument, and prints the array. *(Note: The length of an array* `A` *can be accessed with* `A.length`*)*

   (b) (4 points) Write a public static method `arrayAdd` that takes two integer arrays as arguments, and returns an integer array that in its i-th position contains the sum of the elements in the i-th position in the input arrays. Find a reasonable way to handle the case when the input arrays do not have the same length, and document it in the comments. For testing, use your method above to print the array.

   (c) (2 points) Yes, Java also supports recursion... :-) Write a *recursive* private static method that prints an array. The arguments to your method should be the array, **as well as an additional index variable**. This index variable will allow you to specify a recursive case. Can you conclude the overall printout with a newline?

2. **Rectangle (8 points)**

   (a) (4 points) Implement a class `Rectangle` that represents an axis-parallel rectangle in the plane. The rectangle should provide the functionality (in the form of methods) to return its area, to return its circumference, and to be translated. Provide the necessary attributes (i.e., variables) to ensure this functionality, as well as all corresponding getter and setter methods. Make your implementation as clean as possible, with proper use of access modifiers (private vs. public) for attributes and methods. Also add a `toString` method.

   (b) (2 points) Add a static counter to your `Rectangle` class that counts the number of rectangle objects that have been created. For this you have to increment this counter in the constructor. We did exactly the same for the `Point` class in class. Now, use this static counter to assign a unique ID to each rectangle object that has been created. For this you will have to introduce an additional ID attribute, as well as a corresponding getter method (a setter method does not make sense).

Flip over to back page ⟹

(c) (2 points) Test your `Rectangle` class in the main method of a separate `RectangleTester` class. Make sure to test all the functionality.

(d) **(1 point extra credit)** Add functionality to intersect two rectangles, by adding a method `intersect` that takes another `Rectangle` as argument, and intersects both rectangles. The result should be stored in the current `Rectangle` object. How can one handle an empty intersection?

3. **Mystery (4 points)**
Consider the following code:

```
public class Mystery {

  public static int mystery1(int a){
    int b=a;
    a = a*a;
    return b;
  }

  public static Point mystery2(Point a){
    Point b=a;
    a.translate(a);
    return b;
  }

  public static void main(String[] args) {
    int x = 5;
    Point p = new Point(5,5);
    x=mystery1(x);
    p=mystery2(p);
    System.out.println("x="+x+", p="+p);
  }
}
```

What is being printed, and why? Explain your answer by drawing pictures of the variables in memory during different times: (1) in the `main` method right before the calls to the mystery methods, (2) in `mystery1`, (3) in `mystery2`, and (3) at the end.