CS-207: Programming II
Spring 2016
Northeastern Illinois University
Homework #4: Due 02/11/16 at 9:00 a.m.
Objects and Classes

Problem #1

- Trace through the class with the `main` method below.
- You should trace through this by hand as you will have a tracing problem on your first exam that deals with objects and arrays.
- Put the output (correctly formatted) in a text file named Homework4.txt.
- Add this to a folder named Homework4 to be submitted to D2L.

```
public class Tweet
{
    private int a;
    private int b;
    public static int c = 0;

    public Tweet(int a, int b)
    {
        this.a = a;
        this.b = b;
        c += this.a + this.b;
    }

    public int getA()
    {
        return this.a;
    }

    public int getB()
    {
        return this.b;
    }

    public void setA(int a)
    {
        this.a = a;
    }

    public void setB(int b)
    {
        this.b = b;
    }
}
```

```
public class Homework4
{
  public static void main(String[] args)
  {
    Tweet t1 = new Tweet(3, 8);
    Tweet t2 = new Tweet(-3, 7);
    Tweet[] tweets = { t1, t2, new Tweet(0, -5) };

    one(tweets);

    for (int i = 0; i < tweets.length; i++)
    {
      System.out.print("Index 0: ");
      System.out.print("a is " + tweets[i].getA());
      System.out.println(" b is " + tweets[i].getB());
    }
    System.out.println("c is " + Tweet.c);
  }

  public static void one(Tweet[] tws)
  {
    for (int i = 0; i < tws.length - 1; i++)
    {
      Tweet w = tws[i];
      Tweet x = tws[i + 1];
      w.setA(2 * x.getB());
      w.setB(-2 + x.getA());
    }
    tws[2] = tws[0];
    tws[0].setA(-4);
  }
}
```

Problem #2

1. Create a class called `MyPoint`, which models a 2D point with x and y coordinates. The class should have the following instance variables, constructor(s) and methods:

   - Two private integer instance variables named `x` and `y`.
   - A "no-arg" constructor that creates a point at the (`x`, `y`)-location (0, 0).
   - A constructor that takes two integer parameters and creates a point with the given `x` and `y` coordinates. **Note:** Order matters for these parameters. The `x`-value should be the first parameter and the `y`-value should be the second parameter.
   - Getter methods (these must be named properly!!) for the instance variables `x` and `y`.
   - A method named `setXY` that takes two integer parameters to set both `x` and `y`. The value for `x` should be the first parameter and the value for `y` should be the second parameter. The method should not return anything.
   - A method named `toString` that returns the values of the `x` and `y` in the following String format: (x, y)
   - A method named `distance` that takes two integer parameters (the first being a value for an `x`-coord and the second being a value for a `y`-coord). The method returns the distance (a `double` value) from the point specified in the instance variables to the location specified by the parameters.
   - An overloaded method named `distance` that takes a `MyPoint` object as a parameter. The method returns the distance (a `double` value) from the (`x`, `y`)-location of the `MyPoint` object that calls the method to the (`x`, `y`)-location of the `MyPoint` object passed in.
   - Distance can be calculated using the following equation:

     $$\sqrt{(\mathrm{x}_1 - \mathrm{x}_2)^2 + (\mathrm{y}_1 - \mathrm{y}_2)^2}$$

   - Use `Math.pow(`*`num`*`, 2)` to raise a value to the 2nd power.
   - Use `Math.sqrt(`*`num`*`)` to find the square root of a value.

2. Download the needed files from the course website and find the `MyPointTest.java` file. Compile both the `MyPoint` and `MyPointTest` classes. Run the `MyPointTest` class. If you created the `MyPoint` class correctly, you will see the following output.

```
The distance from (0, 0) to (3, 4) is: 5.0
x is: 10
y is: 10
The distance from (10, 10) to (4, 8) is: 6.324555320336759
The distance from (4, 8) to (7, 3) is: 5.830951894845301
```

Problem #3

1. Create a class named `PerformanceSeating` which models empty and booked seats at a performance. The class should have the following instance variables, constructor, and methods:

   - Private instance variable: A 2D character array named `seats`.

   - A constructor that takes two integer parameters representing the number of rows and columns for the grid of seats at the performance and then creates the `seats` array appropriately. Initially, all seats should be marked as available. Booked seats are represented by the character `'X'` and available seats are represented by the character `'A'`.
   - Create a method named `findBookedSeats` that does not take any parameters, but determines how many booked seats there are at the performance and returns that value.
   - Create a method named `findAvailableSeats` that does not take any parameters, but determines how many available seats there are at the performance and returns that value.
   - Create a method named `bookSeats` that takes an integer as a parameter and books the first `n` available seats (starting at the first row and checking from the first row to the last row). The method should not return anything.
   - Create a method named `cancelSeat` that takes two integers as parameters that represents a row index and column index (in that order) and cancels or "unbooks" the seat at that row and column index. The method should not return anything.
   - Create a method named `printSeats` that prints the characters of each row on its own line (see output below).

2. Download the needed files from the course website and find the `PerformanceSeatingTest.java` file. Compile both the `PerformanceSeating` and `PerformanceSeatingTest` classes. Run the `PerformanceSeatingTest` class. If you created the `PerformanceSeating` class correctly, you will see the following output.

```
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A

X X X X A X X X X X
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A
A A A A A A A A A A

X X X A X X X X X X
X X X X X X X X A X
X A X X X X X X X X
A A A A A A A A A A
A A A A A A A A A A

Available seats: 23
Booked seats: 27
```

A note on cheating/plagiarism:
A plagiarism detector is used on all submitted code (across all sections) for homework assignments. If the plagiarism detector determines that 25% or more of your code for a particular assignment is plagiarized, you will receive a zero (i.e. an F) for that homework assignment, regardless of whether you

cheated from someone or vice-versa. If you plagiarize half or more of the total homework assignments, you will receive a zero for the entire homework percentage.

Submitting your assignment to D2L

1. Make sure your name and assignment number are in the .java file(s) (as comments) and text file.
2. Place all your files in a folder and compress (i.e. .zip) the folder. Submit the .zip file to the Homework #4 folder on D2L. You should submit only one file - the .zip file. Do **NOT** upload multiple files.
3. Turn your homework in to D2L by the specified deadline (no late homework will be accepted - see syllabus for policies)