Homework 2 (due by 4 pm on Nov 7)

Objectives:

- Understand how Queue/Deque can be used as an aid in an algorithm.
- Explore and compare different data structures (ArrayDeque and LinkedList) and their methods to solve the same problem.

Josephus Problem (from Wikipedia)
There are people standing in a circle waiting to be executed. A certain number of people are skipped and then one person is executed. This is repeated starting with the person after the person who was executed. The elimination proceeds around the circle (which is becoming smaller as the executed people are removed) until one person remains.

Named after Flavius Josephus, a Jewish historian living in the 1st century. According to Josephus' account of the siege of Yodfat, he and his comrade soldiers were trapped in a cave, the exit of which was blocked by Romans. They chose suicide over capture and decided that they would form a circle and start killing themselves using a step of three. Josephus and another man remained alive last and gave up to the Romans.

The goal is to determine where to stand in the circle so that you are the last person alive.

Now, the problem is that you are in the same situation just like Flavius Josephus. Oh no! You are in danger!

However, you are a programmer and smart enough to build an algorithm that tells you where to stand in the circle to survive.

As you can imagine, ***time is critical here***. You need to quickly implement and compare three different methods in Josephus Java class and decide which method is the fastest to use for you to survive.

You are to implement three different methods to solve the Josephus problem. The following is the skeleton class for you.

```
public class Josephus {
  /**
   * This method uses ArrayDeque data structure as Queue/Deque to find the survivor's position.
   * @param size Number of people in the circle that is bigger than 0.
   * @param rotation Elimination order in the circle. The value has to be greater than 0.
   * @return The position value of the survivor.
   */
  public int playWithAD(int size, int rotation) {
    // TODO implement this
  }

  /**
   * This method uses LinkedList data structure as Queue/Deque to find the survivor's position.
   * @param size Number of people in the circle that is bigger than 0.
   * @param rotation Elimination order in the circle. The value has to be greater than 0.
   * @return The position value of the survivor.
   */
  public int playWithLL(int size, int rotation) {
    // TODO implement this
  }

  /**
   * This method uses LinkedList data structure to find the survivor's position. However, this does not use the LinkedList as Queue/Deque. Instead, this method uses index to find a person to be executed in the circle.
   * @param size Number of people in the circle that is bigger than 0.
   * @param rotation Elimination order in the circle. The value has to be greater than 0.
   * @return The position value of the survivor.
   */
  public int playWithLLAt(int size, int rotation) {
    // TODO implement this
  }

}
```

Step-by-step guide:

- Understand how Josephus problem works.
  - For your understanding, the steps of execution are as follows using Queue data structure (size = 6, rotation = 3).

Initial Circle: 1 2 3 4 5 6 After the 1st execution: 4 5 6 1 2 After the 2nd execution: 1 2 4 5 After the 3rd execution: 5 1 2 After the 4th execution: 5 1 After the 5th execution: 1 (1 survives)

- ***Read carefully the Javadoc comments of each method in the skeleton class provided.***
  - Think carefully about what kind of methods of the two Java classes ought to be used in your algorithm.
  - ArrayDeque(http://docs.oracle.com/javase/6/docs/api/java/util/ArrayDeque.html)
  - LinkedList(http://docs.oracle.com/javase/6/docs/api/java/util/LinkedList.html)
- Write your code on paper first.
  - For this homework, you need to write the whole Josephus class.
  - You need to submit the paper in class.
  - Give enough space per line as you write.
  - *Try to finish within 30 minutes. We will NOT deduct any points because of mistakes made at this point.*
- Implement the methods on Eclipse or any other IDEs you prefer.
  - You also need to submit the source code file, Josephus.java, on the Blackboard.
- Test your code extensively!
  - Use the provided MainDriver file to test your program. When you run the program, you should see the following result.

size (number of people)	rotation	survivor
1	5	1
5	1	5
5	2	3
11	1	11
40	7	24
55	1	55

66	100	7
1000	1000	609
-1	2	exception
5	0	exception

  - ***Notice that rotation value can be bigger than size that is the number of the people in the circle.***
  - *Remember! Time is critical here. You cannot afford to be slow to find out where to stand. So, in EACH iteration, your algorithm should be able to* ***find the elimination position without rotating the circle more than its current size****.*
- Update your paper code with comments.
  - Focus on the areas that were not done correctly or inefficiently. (Use a pen of different color.)

Deliverables:

- A few sheets of paper that have your initial code as well as your comments (Submit this in class that is on the due date.)
  - In your comments, clearly write ***WHICH method(s) you would use*** out of three methods, assuming that there are many people in the circle, to find the survivor's position and **WHY**.
- Your source code file. (Submit this on Blackboard by the due date.)

Grading:

Your homework will be graded first by compiling and testing it. After that, we will read your code to determine appropriate methods/classes are used. In addition, we will judge the overall style and modularity of your code. ***Points will be deducted for poor design decisions, uncommented and unreadable code***. Your comments on your paper should be able to demonstrate your proper understanding of the code.

- Working code: 60 points
- Coding style (refer to the guideline): 20 points
- Your paper code's comments: 20 points

Late submission will not be allowed and, if you have multiple versions of your code file, make sure you do submit the correct version. Only the version submitted before the due will be graded.

You can use the following MainDriver class to test your methods.

```
public class MainDriver {
   public static void main(String[] args) {
      int size = 1000;
      int rotation = 1000;

      Josephus game = new Josephus();
      Stopwatch timer1 = new Stopwatch();
      // call playWithAD method
      System.out.println("Survivor's position: " + game.playWithAD(size,
rotation));
      System.out.println("Computing time with ArrayDeque used as
Queue: "+timer1.elapsedTime()+"millisec.");

      Stopwatch timer2 = new Stopwatch();
      // call playWithLL method
      System.out.println("Survivor's position: " + game.playWithLL(size,
rotation));
      System.out.println("Computing time with LinkedList used as
Queue: "+timer2.elapsedTime()+"millisec.");

      Stopwatch timer3 = new Stopwatch();
      // call playWithLLAt method
      System.out.println("Survivor's position: " +
game.playWithLLAt(size, rotation));
      System.out.println("Computing time with LinkedList (remove with
index) : "+timer3.elapsedTime()+"millisec.");
   }
}
```