CS146 Data Structures and Algorithms

Fall 2018, Instructor: K. Potika SJSU CS Department

Programming Project 1

Important: Do individually (each student alone is NOT a group assignment), each student has to turn in a java program. For this project we will have Code review with the grader or the Instructor.

A. Working at MusicBest

You are planning a road trip and want to create a playlist of your favorite songs. Assume that the song titles are in an array of strings. Create a shuffle of your songs (permutation of your original songs).
Use the Fisher–Yates shuffle algorithm that works in O(n) running time. We will use a method that creates pseudo-random numbers (see end for help) in O(1) running time.
The basic idea is to start from the last element, swap it with a randomly selected element from the whole array (including last). In the next step you will consider the array from 0 to n-2 (size reduced by 1), and repeat the process until you reach the first element. Write a program that uses the provided Playlist.txt as input and outputs the shuffled array in a file called LastNameFirstNamePlaylist.txt.
Follow the next pseudocode:

```
To shuffle an array a of n elements (indices 0..n-1):
for i from n - 1 downto 1
    j = random integer with 0 <= j <= i
    exchange a[j] and a[i]
```

Create appropriate JUnits to test your program.

B. Circular linked list game

In an ancient land, a King had many prisoners. He decided on the following procedure to determine which prisoner to grand freedom. First, all of the prisoners would be lined up one after the other and assigned numbers. The first prisoner would be number 1, the second number 2, and so on up to the last prisoner, number *n* . Starting at the prisoner in the first position, he would then count k prisoners down the line, and the kth prisoner would be eliminated from winning her/his freedom removed from the line. The King would then continue, counting k more prisoners, and eliminate every kth prisoner. When he reached the end of the line, he would continue counting from the beginning.
For example, if there were six prisoners, the elimination process would proceed as follows (with step k=2):

1->2->3->4->5->6 Initial list of prisoners; start counting from 1.
1->2->4->5->6 Prisoner 3 eliminated; continue counting from 4.
1->2->4->5 Prisoner 6 eliminated; continue counting from 1.
1->2->5 Prisoner 4 eliminated; continue counting from 5.

1->5 Prisoner 2 eliminated; continue counting from 5.
1 Prisoner 5 eliminated; 1 is the lucky winner.
Write a program that creates a circular linked list of nodes to determine which position you should stand in to win your freedom if there are *n* prisoners. Your program should simulate the elimination process by deleting the node that corresponds to the prisoner that is eliminated for each step in the process.

Create appropriate JUnits to test your program.

For both programs:

Programming Standards:

- Your header comment must describe what your program does.
- You must include a comment explaining the purpose of every variable or named constant you use in your program.
- You must use meaningful identifier names that suggest the meaning or purpose of the constant, variable, function, etc.
- Precede every major block of your code with a comment explaining its purpose. You don't have to describe how it works unless you do something tricky.
- You must use indentation and blank lines to make control structures more readable.

Deliverables:

- ✓ Your main grade will be based on (a) how well your tests cover your own code, (b) how well your code does on your tests (create for all non-trivial methods), and (c) how well your code does on my tests (which you have to add to your test file). For JUnit tests check canvas.
- ✓ Use sjsu.<lastname>.cs146.project1 as your package, and Test classes should be your main java file, along with your JUnit java tests.
- ✓ What to submit to canvas (Steps):
    - Export your project on eclipse with your entire project (source code).
    - Create a zip file named **LastnameFirstProjectName.zip** that includes the exported zip file + pdf report
    - upload the last zip file to canvas.
- ✓ All projects need to compile. If your program does not compile you will receive 0 points on this project.
- ✓ Do not use any fancy libraries. We should be able to compile it under standard installs. Include a readme file on how to you compile the project.

Extra Help for JUnit tests: Create the 'same' pseudo-random numbers by using a specific seed (e.g. 0):

Random generator (once)

in the class

```
public class MYCLASS{
        …
        private Random myRandGen;
        …
        double myrandom() {
                return myRandGen.nextDouble(); //random in 0-1
        }
```

And in the constructor

```
public MYCLASS(int dimension_in) {
    myRandGen=new java.util.Random(0); //seed is 0
}
```

Usage

```
(int)(myrandom() * size of array)
```