CPSC 2151

Lab 7

Due: Tuesday Nov. 7th or Thursday Nov. 9th **Before Lab!**

In this lab, you will use the decorator pattern to extend the `List` interface. We want to add the ability to swap two positions in the list and to shuffle the list to the `List` interface. Unfortunately, we cannot edit the `List` interface, so we extend the `List` interface to do so. We will need to provide an implementation of the new interface, but we don't want to have to recode all the methods from `List`, nor do we want to create an implementation for each implementation of the `List` interface. Luckily, the decorator pattern allows us to do this easily.

Refer to the video and the accompanying slides on the decorator pattern for help.

Instructions

1. Create a new project in IntelliJ with a package named `cpsc2150.listDec`
2. Add the `ShuffleApp` and `ShuffleList` classes provided on Canvas to your project
3. Create an interface called `IShuffleList`, and have it extend the `List` interface
    a. Remember, this interface will inherit all the methods from `List`, so we don't need to list them all out.
    b. This interface must also be generic.
    c. Add two default methods to the interface. Write contracts for them as well.
        i. `default void shuffle(int swaps)`
            1. Randomly picks two positions in the list and swaps them
            2. Repeats this `swaps` times
        ii. `default void swap(int i, int j)`
            1. Exchanges the values at positions `i` and `j` in the list
4. Follow the decorator pattern to complete the missing methods in `ShuffleList`. Many methods are already completed illustrating the decorator pattern.
5. Create a make file with the following targets
    a. `default`: compiles the code
    b. `run`: runs `ShuffleApp`
    c. `clean`: remove all compiled `.class` files
6. Test your code on SoC Unix computers.

Random Numbers in Java

We will need to create random numbers in order to shuffle our `List`. To do so, you need to import `java.util.Random`. Then you can declare a variable of the datatype `Random`.

```
Random rand = new Random();
```

To get a random number between 0 and N (not including N), you just call the `nextInt(int)` method, passing N in as the parameter. For example, if you wanted to get a random number between 0 and 100 (but not including 100), you would call:

```
int rand_num = rand.nextInt(100);
```

Partners

You are required to work with 2-3 partners on this lab assignment. Make sure you include both partners' names on the submission. You only need to submit one copy. Remember that working with a partner means working *with* a partner, not dividing up the work.

Before Submitting

You need to make sure your code will run on SoC Unix machines and create a `makefile`.
Include both of your names as a comment at the top of each Java file.

Submitting to Gradescope

Upload your `Lab10.zip` to Gradescope. It should automatically check if you have submitted everything correctly and verify that it was able to compile and test your code. While not all test cases are visible, you should make sure that all visible test cases pass. The visible test cases should be a good starting point for checking whether your solution is correct or not.

Only one member per group needs to submit the assignment, but you need to make sure to select your partner's name when creating a submission on Gradescope.

NOTE: Make sure you zipped up your files correctly and didn't forget something! Always check your submissions on Gradescope to ensure you uploaded the correct zip file and it is in the proper format that it expects. Notify your course instructor immediately of any Gradescope issues.