CSC 220 Data Structures

Program #4

Stacks, Queues and Generics

Description:

Your objective is to implement a stack and a queue using the list implementation you did in assignment #3. You must use the linked list implementation of a list. You are to use your list operations to accomplish the operations of a stack and queue.

Getting started:

- Grab your ListAsLinkedList class and your Node class from assignment #3
- In the List.java file given to you, where it says "public class List", put your ListAsLinkedList class's fields and methods. To be clear, this creates a new class called List (that does NOT implement IList) that is the exact same as your ListAsLinkedList class. DON'T FORGET to rename the constructor to "List".
- Where it says "class Node", put your Node class's fields and methods.
- You will not need the IList interface nor the ListAsArray class, so do NOT include them.
- The Stack.java and Queue.java files given to you are where you will implement your stack and queue. They have been "stubbed out" meaning method "stubs", or blank methods, have been created for you. These need to be given an implementation that will carry out the operations.

Running the code:

The Main.java file contains the entry point, so compiling and running that class will run the application. The undo command in the game relies on a correct stack implementation and the replay command relies on a correct queue implementation. Testing these two commands will test your stack and queue implementations and ensure they are functioning correctly.

Next steps (adding generics):

Make sure everything is working correctly before moving on to this step. After things are working, you need to turn the Stack and Queue classes into generic classes. This requires several changes including a change to the List and Node classes to turn them into generic classes as well. Refer to the notes for help. After making these changes, you must modify the Main class to use your generic Stack and Queue classes. This means giving both Stack and Queue a value for the generic type parameter both at declaration and at instantiation. As an example, if you had the following code:

```
List list;  // declaration
list = new List();  // instantiation
```

you would change it into:

```
List<Datatype> list;  // declaration
list = new List<Datatype>();  // instantiation
```

where "Datatype" is the data type you want to use, either String, Integer, Character, or something else.

You only need to specify the datatype during declaration and instantiation, not every time you use the object.

Help with generics:

You may have noticed this warning when compiling your code:

```
Note: List.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

This indicates that you have missed a generic parameter somewhere in your code. As the warning says, you can use -Xlint to find these issues. To do this, type the following in the terminal:

```
javac -Xlint Main.java
```

By default, compiling Main will compile other classes that are referenced from Main. So, this should show all of your warnings and help point out any places that you forgot to put generics on. If you don't get any warnings, you might want to remove the .class files manually, then run javac again. If you still don't see any warnings, you should be good.

Submitting your assignment:

Since you have multiple files this time, you will need to zip them up before submitting. So, make sure to grab all four files (List.java, Stack.java, Queue.java, and Main.java), zip them up into a zip folder and submit that folder. Do not submit any .class files.

Rubric:

#	ITEM	POINTS
1	Correct stack implementation	13
2	Correct queue implementation	13
3	Uses generics fully and correctly	10
4	Game functions as intended	4
	TOTAL	**40**

#	PENALTIES	POINTS
1	Doesn’t compile	-50%
2	Doesn’t execute once compiled (i.e. it crashes)	-25%
3	Late up to 1 day	-25%
4	Late up to 2 days	-50%
5	Late after 2 days	-100%