CMPS 261: Project 4

Due Date: Check on Moodle

Instructions

1) You are allowed to discuss the problem and solution design with others, but the code you submit must be your own. Your solution must include the certification of authenticity as given in the course syllabus.

2) Follow the instruction on Moodle under "Submitting to Moodle".

General Requirements:

- Project submissions not including a Certification of Authenticity will not be graded until such a written and signed submission is provided by the student.
- Code, even if substantial, that fails to compile will not receive a grade higher than 25%
- Code, even if substantial, that fails to run will not receive a grade higher than 50%
- A penalty of up to 10% may be assessed for lack of required documentation, lack of descriptive identifiers, lack of indention of blocks of code, failure to follow naming requirements of the project and associated tar archive.
- Failure to submit the complete NetBeans project may result in a penalty of up to 10%.

Problems

1) (50 Points) Modify ListAPI.

a) Download "ListAPI.zip" from Moodle and extract the NetBeans project folder "ListAPI" from the archive file. In NetBeans under the Project tab, right click on ListAPI and select Rename. In the Rename Project dialog box, rename the project "p4_1_your-clid", select the Also Rename Project Folder option and click Rename.

b) Define the following method in MyList and implement it in MyAbstractList.

```
// Adds the elements in otherList to this list.
// Returns true if
// this list changed as a result of the call.
public boolean addAll(MyList<E> otherlist)
```

c) Add code to "TestMyArrayList.java" to demonstrate that this method works correctly. Display the modified list using a java.util.Iterator<E> object and a while loop.

2) (50 points) Use ListAPI as a library.

a) Create a NetBeans project and name it "p4_2_your-clid".

b) Create a JAR file of "ListAPI".

i. Download "ListAPI.zip" from Moodle and extract the NetBeans project folder "ListAPI" from the archive file.

ii. In NetBeans, open "ListAPI". Under the Project tab, right click on ListAPI and select Clean. Under the Project tab, right click on ListAPI and select Build. The created jar file is in "dist" folder.

iii. Move the JAR file to the folder that contains your project folder. Navigate to the "ListAPI" NetBeans project folder and open it. Open the folder "dist". Copy the file "ListAPI.jar" and paste it in the same folder that contains the "p4_2_your-clid" project folder.

c) Attach the "ListAPI.jar" to your project.

i. Under the NetBeans Project tab, open the project created in step (a).

ii. Right click on Libraries and select "Add JAR/Folder...", and then navigate to "ListAPI.jar". In the Add JAR/Folder dialog box, select "ListAPI.jar", select Relative Path (which, if you have followed the preceding instructions, will be "../ListAPI.jar") and click OK.

iii. The classes in the JAR file are now available either by adding the following import statement to classes in your project

```
import listapi.*;
```

or by entering the full name of a class as in the following example.

```
listapi.TestGenericQueue tgq
= new listapi.TestGenericQueue();
```

Note: When distributing your code, libraries must be distributed as well. For example, if you wish to distribute your project "p4_2_your-clid", first clean this project, then build the project. In the "dist" directory in your project folder, there will be a JAR file named p4_2_your-clid.jar" and a "lib" folder that contains the libraries you have attached to your project. Both of these must be distributed in order for the recipient to be able to run your code. The README.TXT file gives instructions on how to run the project.

d) In your project "p4_2_your-clid", add code to enqueue 5 integer values in a listapi.GenericQueue, displaying the queue as each value is added. Then, dequeue each element in the queue, adding each value to a listapi.GenericStack, displaying the queue and stack as each value is moved from the stack to the queue. Finally, pop all values of the stack and add them to a listapi.MyArrayList, displaying the stack and the list as each value is moved from the stack to the list.

3) (50 points) Build and use your own minimal stack class using an array to hold the stack.

a) Create a NetBeans project and name it "p4_3_your-clid".

b) In your project, create a stack class based on an array of type double of fixed size. In the class,

i. The array should have the default size of 5.

ii. The array size should be settable via a constructor.

iii. The following methods must be implemented:

A. push – inserts a value a the top of the stack

B. pop – returns the top value, removing it from the stack

C. isEmpty – returns true if the stack is empty, false otherwise

D. isFull - – returns true if the stack is full, false otherwise

c) Add code to your project demonstrating the stack and all methods and constructors in the class work correctly.

4) (50 points) Build and use your own minimal queue class using an array of type String of fixed size. to hold the queue.

a) Create a NetBeans project and name it "p4_4_your-clid".

i. The array should have the default size of 5.

ii. The array size should be setable via a constructor.

iii. The following methods must be implemented:

A. enqueue – inserts a value at the rear of the queue

B. dequeue – returns the value at the front of the queue, removing the value from the queue

C. isEmpty – returns true if the queue is empty, false otherwise

D. isFull - – returns true if the queue is full, false otherwise

Tip: In a queue of 2 or more elements, the index of the rear is "larger" than the index of the front – EXCEPT, the queue will need to "wrap around" the array. For example: If the array is 20 elements. the queue front is at index 15 and the queue rear is at index 19, the next enqueue will be at index 20, which doesn't exist. The index must "wrap around" the array, i. e. the next enqueue will be at index 0. The solution is to modulus by the array size whenever increasing an index, either rear of front. For example, to change the index of the rear to the next index,use the formula

```
rear = (rear + 1) % arraySize;
```

b) Add code to your project demonstrating the queue and all methods and constructors in the class work correctly.

Additional Requirements

[1] Identifiers must be descriptive, i.e. must self-document. The only exception granted is in the case of a "for variable", that is a variable created as a simple counter as in the control

variable in a “for” loop statement.

[2] Indention of all code blocks (compound statements, anything in braces), including single statements following selection or while statements, is required. NetBeans will do this fairly automatically as you type if your syntax is correct. In NetBeans, ALT-SHIFT-F will re-format a whole file if your syntax is correct.

[3] The main “.java” file [the one with the method *public static void main(String[] args)*] of your project must contain this minimal documentation:

```
// Your Name
 // Your CLID
 // CMPS 261
 // Program Description: description of actions of code
 // Certificate of Authenticity:   (Choose one of the two
 following forms:)

     I certify that the code in the method functions
      including method function main of this project are
      entirely my own work.

 {or}

     I certify that the code in the method functions
      including method function main of this project are
      entirely my own work., but I received some assistance
      from {name}.  Follow this with a description of the
      type of assistance. (For example, if you consulted a
      book, and your solution incorporates ideas found in the
      book, give appropriate credit; that is, include a
      bibliographical reference.)  Note:  You do not have to
      list the text, the author of the text or the
      instructor's examples.
```