COMP202 SPRING 2017

PROGRAMMING PROJECT 1 *23.02.2017*

Submission Location: S Drive

***Your in-lab submissions will be checked for plagiarism and the rules in the course syllabus will be effective.*

***Read the description carefully and understand it well. Then start implementing it!*

***Do not modify the Test.java file*

You **are not** allowed to use any java library that substitutes the parts you are asked to implement and solve. Otherwise, your grade will be 0.

1 DESCRIPTION

In this lab project, you are required to solve a given problem using $Doubly\ Linked\ Lists$. We want to store student information in a database which is implemented using a doubly linked list. Each node in the linked list represents a *student*. Every student (which is represented by a node) in the linked list has a $\boldsymbol{name}$, a $\boldsymbol{surname}$, an $\boldsymbol{age}$, a *unique student* $\boldsymbol{ID}$, and a $\boldsymbol{major}$ (which is the department s/he studies). The database is sorted in ***ascending*** order according to student IDs. You are given three java files, one text file, and one pdf file in project folder as $\boldsymbol{Test.java}$, $\boldsymbol{StdNode.java}$, $\boldsymbol{StdDoubleLL.java}$ (which represents the implementation of the database), $\boldsymbol{Output.pdf}$ and $\boldsymbol{students.txt}$. Your task is to implement missing methods, and constructors (if necessary) in $StdDoubleLL$ class.

1.1 STDNODE.JAVA FILE

$StdNode$ class is just a node class representing a student. A $StdNode$ class has following variables and method(s);

Variables& Methods	***Description***
$name$	The name of the student
$surname$	The surname of the student
age	The age of the student
$major$	The major/department of the student
ID	Unique identifier of the student
$next$	Pointer to the next node
$prev$	Pointer to the preceding node
$toString()$	Returns information of the student in a specific format.

Table 1: Description of StdNode.java file.

1.2 TEST.JAVA FILE & STUDENTS.TXT FILE

The $Test$ class is where the database is created, updated, and printed. In the Test class, the student.txt file is used to store the operations to be applied on the database ($StdDoubleLL$). Each line in the student.txt follows the given a format:

$$< \boldsymbol{command} >< \boldsymbol{ID_{student}} >< \boldsymbol{name_{student}} >< \boldsymbol{surname_{student}} >< \boldsymbol{age_{student}} > < \boldsymbol{major_{student}} >$$

The command in the beginning of the line represents the operation to be applied on the database. The legitimate commands are shown in Table 2. These are already handled for you.

Command	***Description***
$insert$	Inserts the student, whose information is given, into the $StdDoubleLL$
$print$	Prints out the desired students in $StdDoubleLL$. The prints commands are *printIDInterval*, *printMajor* and *printAll*. Details are in Section 1.3
$search$	Searches for the given student in $StdDoubleLL$
$delete$	Deletes the student whose ID is given.

Table 2: Legitimate commands for the database operation.

1.3 STDDOUBLELL.JAVA FILE

The $StdDoubleLL$ class is a modified doubly linked list. The $StdDoubleLL$ has a $median$ node, which points to the node in the middle of the linked list, in addition to $head$ and tail nodes. The general structure of $StdDoubleLL$ is shown in *Figure 1*.

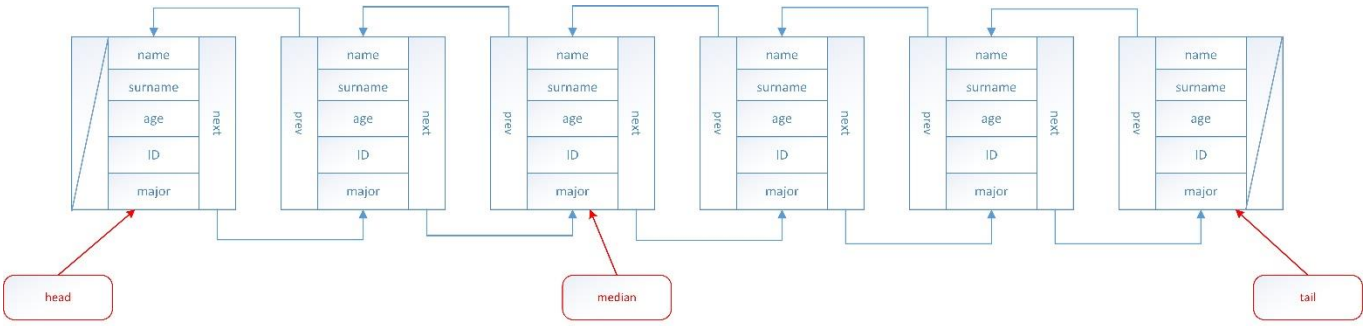

Figure 1: General structure of $StdDoubleLL$

When you open the $StdDoubleLL.java$ file, you will see the missing parts. **Your task is to fill these missing parts**. Description of the $StdDoubeLL$ class is shown in Table 3.

Methods & Variables	***Description***
$head$	A StdNode pointer, pointing to the head of the list
$tail$	A StdNode pointer, pointing to the tail of the list
$median$	A StdNode pointer, pointing to the middle of the list. If the number of students is odd, then it should point to (size+1)/2 and if it is even (size/2)
$insert(StdNode)$	Inserts a new student to the list. The list should be sorted in ascending order according to the ID. You need to find the right position to insert the new student by starting from the median student node. You should look at the median student's ID, and compare it with the new student's ID, and then; • If the median student's ID is greater, you should search for the position between the head and the median nodes • Otherwise, search for the position between the median and the tail nodes. After finding the right position, insert the new student. *You are not allowed to insert first then sort all the list. Remember that the IDs are unique. If a student with an already existing ID is attempted to be inserted, you need to throw a DuplicateIDException which is provided to you. You just need to know when and how to throw.
$search(ID)$	Searches for a student by given ID in the list and returns the student if it is in the list. When the ID is not in the list, returns null since it is handled in the $Test.java$ file. You can check it how it is handled. *(You should not search the list from head to tail. Use the median node for searching the student.)*
$delete(ID)$	Deletes the student with given ID. It returns $true$ if deletion is successful, returns $false$, otherwise (i.e. if the ID is non-existent).
$print(ID1, ID2)$	Prints all of the students' records between ID1 and ID2 in the list. The command for this method is $printIDInterval$ We are only going to test with existing IDs so you do not need to check whether they are in the list or not.
$print(major1)$	Prints all students' records who are enrolled in major1. The command for this method is $printMajor$
$print()$	Prints all of the students in the list. It prints out "$There\ is\ no\ one$" if the list is empty. The command for this method is $printAll$

Table 3: Description of the methods, variables of $StdDoubleLL$ class.

2 INPUT & OUTPUT

You can check the expected output from $\boldsymbol{Output.pdf}$ file. In order to get student information (unsorted) and decide which operation to apply, you are provided $\boldsymbol{students.txt}$ file. For testing you can play with it and see if you are handling all the conditions.

3 RULES

1) You **are not** allowed to use any java library that substitutes the parts you are asked to implement and solve.
2) Your program **must** check against error conditions and throw an exception if any occurs.
3) Your code will be tested with different input files and checked against plagiarism and the rules in the course syllabus will be effective.
4) Do not log off or shut down your computer without checking your submission.
5) It is also part of your task to check if your code is uploaded after submission. There is no way to back-up your code if you shut down/log off from your computer without saving you code to S drive.

Good Luck!
Have fun implementing your task ☺