**KV4000 Week 8: TextUI for LabClass and main method**

**Purpose of this tutorial:** 

Reinforce previous work on Classes, Objects and ArrayLists 

Provide experience of building a text based user interface.

**Introduction**

In this tutorial we are going to add a TextUI to the LabClass application.

Previously you should have implemented a Student class and the LabClass class. Revised copies of these classes are available on Blackboard. Please read and understand the code contained in the LabClass class.

*Adding the Student and LabClass classes*

In BlueJ click the *edit* option on the menu bar. Select add files. Find the copies of Student.java and LabClass.java and add them to your project.

**TextUI Class**

*Brief Specification*

This class controls the I/O for your program. It should provide the following commands:

|  |  |  |
| --- | --- | --- |
| Function | Option | LabClass Methods? |
| Display full class details | 1 |  |
| Find student by name[[1]](#footnote-1) | 2 |  |
| Add student | 3 |  |
| Set Instructor | 4 |  |
| Set Room Number | 5 |  |
| Set Time and Day | 6 |  |
| Remove student | 7 |  |
| Get class size | 8 |  |
| Quit | 0 |  |

The I/O is menu driven. When the program is running the user will be presented with list of functions and asked to choose an option.

(Hint: You should look at the example covered in class in order to obtain an idea about the structure of this class).

*Building your TextUI class*

**Task 1**

Write out the responsibilities for the class LabClassTUI.

**Task 2**

Build the class. As noted above you will find it useful to model your class on the lecture example.

Start by locating the LabClass methods that perform the required functions of the interface.

Now create the basic class (remember you will need a Scanner object):

* import(s) }
* Javadoc comment explaining the purpose of the class, author, etc. } compile
* attribute declaration(s) - what do you need? }
* constructor } compile

Think of the menu() method on slide XXXX of the lectoral slides. What method(s) does it call? They will probably need to be (at least partially) written before you write menu(). Sometimes, in turn they will also require a method to be available.

A parameter of type int

public void menu()

calls displayMenu(), getCommand() and execute(int command)

* execute(int command) calls a method for each option on the menu.
* displayMenu() and getCommand() contain no method calls. Write these methods and test them. Do they behave as you expect? Remember they are probably helper methods and therefore private.

private void execute (command)

* calls a method for each function. You can either write all these methods first, or alternately provide just enough code for the method to compile and add the rest as you develop the functionality. As the method has a void return type we can start by writing the header and the opening/closing braces (and compiling).

private void execute(int command)

{

}

Add the code & compile.

With displayMenu(), getCommand() and execute(command) complied, we can now write and test menu().

private void execute(int command)

* The method should contain as series of if – else if statements and finally an else (if it’s not one of the others it must be this one!)
* Each should call a method representing one of the functions in the menu.

Each of these methods should either call a method in the supporting class (LabClass in this case) to perform the function or perform it directly.

* e.g.

A method to remove a student needs to call the equivalent method to carry out the function. The method should obtain the student’s name using the Scanner object and pass it as a parameter to the method in LabClass.

* + A method to close the interface could simply call System.exit(0);

Identify the first function you want to implement (presumably the first one in your menu); write the method and test it.

Write the last function you want to implement (presumably the last one in your menu); write the method and test it.

Add the first and last methods to the execute method. Test it.

private void execute(int command)

{

if(command == 1)

{

// call 1st method

}

// other ‘else if’s will go here

else

{

unknownCommand(command);

}

}

Add the remaining functionality, testing as you go.

**Task 3**

Build a driver class which contains the main method.

1. Note: for simplicity a student’s name is represented by a single string. [↑](#footnote-ref-1)