problem: hash maps – additional

brief

This work builds on previous work using HashMaps. It is intended to reinforce exposure to the HashMap class & its methods.

The task is to develop a small program to record module results for students.

Assuming our previous work (with the phone book) used a HashMap called contacts to map person names to their phone numbers (both represented as Strings) then we can describe our use of HashMaps using the following notation...

```
contacts HashMap{String name → String number}
```

For each student we will map module titles to result as follows (note we use Integer not int because maps cannot use primitive data types)...

```
HashMap{String module → Integer result}
```

Assuming (for simplicity) that student names are unique we can then hold all our data using a Hashmap of these HashMaps...

```
results HashMap{String studentName
                        → HashMap{String module → Integer result}}
```

Java outcomes (see other notes for explanation)

- HashMap
- GUI components & event handling
- control structures (conditional, foreach)
- declaration of classes & methods

stage 1

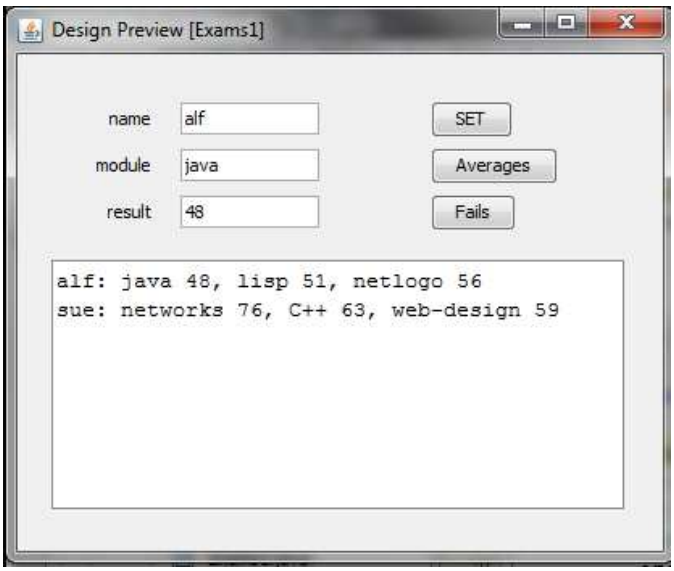
- use the "new file" option in NetBeans to set up a JPanel Swing GUI Form;
- use the NetBeans design feature to create a skeleton JPanel similar to that shown in the screen-shot below (generate event handlers for the buttons but do not write any java code yet) and name the large TextArea "outputArea";

stage 2

For simplicity we will assume that all data is provided and is in the correct format.

- complete the code for the actionPerformed method for the "SET" button as follows...
  1. get the student name & module name from the relevant text fields & store them in local variables;
  2. make a new Integer from the text provided in the result field;
  3. check whether the student already has an entry in the data – if not create a new entry;
  4. get a `HashMap{String module → Integer result}` for the named student and add/amend the result supplied for the named module;

sample GUI



stage 3

- write a new "displayData" method which causes the contents of the results map to be displayed in the outputArea (the large TextArea of the GUI);
- if you used a for-each loop to display the details for each student on a new line you have finished stage 3 (well done!) if not... rewrite the displayData method so it uses a for-each loop to display the details for each student on a new line.

stage 4

- the "Averages" button causes [student name, average of all results] to be displayed in the outputArea – write the code necessary to achieve this.
  1. write a new method "calcAverage" which takes one set of student results as its argument (a `HashMap{String module → Integer result}`) and returns an int (the average mark);
  2. write a for-each loop to go through each of the students in the results, call "calcAverage" and display appropriate information in the outputArea;

stage 5

- the "Fails" button causes [student name, module name & module result] to be displayed in the outputArea – write the code necessary to achieve this.
  notes...
  1. you will need probably to write some kind of loop
  2. you will probably also need a conditional (if statement)