Java programming: Coursework 1

For students enrolled on Principles and Applications of programming this coursework is worth 5% of your overall grade (30 credits). For students enrolled on Java for Industry this coursework is worth 20% of your final grade (15 credits).

The coursework is in 3 parts worth 20, 15 and and 10 marks respectively. Additional marks will be awarded for formatting, style and commenting (5 marks). The total marks available is 50.

Submit your work as a zip file. The zip should contain 3 subfolders, one for each part of the coursework. Each subfolder should only contain the source files. E.g.

myCoursework-
- part1-
  - Student.java
  - Grade.java
  - Main.java
- part2-
  - Password.Java
  - Main.java
- part3-
  - Node.java
  - Tree.java
  - Main.java

To complete the coursework the only imported libraries you should use are **java.util.ArrayList** and **java.util.Scanner**.

Part 1 Student records system

In this part you are going to build a small dummy records system for storing student information in a university. Please follow these instructions carefully.

i. Your programme should consist of 3 classes **Main** (IntelliJ will create this for you if you use the command line template when creating your project), **Student** and **Grade.**

ii. Add the following private instance variables to student, you will need to determine the correct type for each variable **[3 marks]**
   - Name
   - Department
   - age
   - userName
   - studentNumber
   - fullTime

III. Create getters and setters for each of the private instance variables. **[2 marks]**

IV. Write a constructor that sets each of these variables when a new student is created. **[2 marks]**

V. Add a public ArrayList to the student class called grades. grades should store objects of the type Grade. **[1 mark]**

VI. Add 2 private instance variables to Grade; subject and score. Create a constructor that sets both these values, and getter and setter methods for each. **[2 marks]**

VII. Add a static method to Grade, called getLetterGrade, that returns a lettered grade of type char for a parameter score of type double. So that **[3 marks]**:

- a score of >= 70% is an A
- a score of >= 60% is a B
- a score of >= 50% is a C
- a score of >= 40% is a D
- a score of <40% is an F
- An invalid score (either less than 0 or greater than 100 returns ‘E” for error.

VIII. In the main method of your Main class create the following students and grades using the code written above in an ArrayList. The arrayList should be declared as a private static variable of the Main class (You can just add grades one by one to each students grade list directly) **[2 marks]**

Name	Department	Age	UserName	studentNumber	fullTime	grades
Bert Smith	computing	21	bsmit001	12345	TRUE	programming 52, web dev 63, maths 76, algorithms 68
Olivia Green	computing	19	ogree001	12346	TRUE	programming 73, web dev 82, maths 72, algorithms 66
Eloise Jones	computing	18	ejone001	12347	TRUE	programming 65, web dev 63, maths 37, algorithms 40
Ben Bird	computing	42	bbird001	12348	FALSE	programming 55, web dev 29, maths 56, algorithms 38
Karen Brown	computing	25	kbrow002	12349	FALSE	programming 62, web dev 51, maths 43, algorithms 43

ix. Create a menu system in the Main class that allows for uses to interrogate the system. It should have the following options **[5 marks]**.

i. Print out a report of all students including lettered grades for each module score

ii. Print the name of all students with a failed module

iii. print out average grade for each student

iv. Quit the program.

Part 2: Password Generator and Validator

In this part you will build a simple password generator and validation class. Again this class will need a simple menu system for users to access. Obviously, this is just a toy example and not something to be used in the real world!

i. Create a program with two classes Password and Main.

ii. In Password write a static method generator that takes three arguments; length, symbols and digits. The method should return a secure jumbled up password of the given length with the correct number of symbols and digits. Remaining characters can be given by letters, randomised to be either upper or lower case. **[6 marks]** For example:

length 16, symbols 4 and digits 3 could produce the password ‘gD3Wd$^Skw2d%E\8’

iii. In numbers write a second static method to validate the quality of a password passed as a string parameter. The method should return one of the strings in the following table depending on whether the conditions have been met. All conditions must be met for the password to be in the category. **[6 marks]**

iv. Write a menu system that asks the user if they want to generate or verify a password. If they select generate, the system will ask them to provide a length, number of symbols and number of digits. If the user selects verify they will be asked to enter a password to verify. **[3 marks]**

You will need to use the scanner class for the last part of this question.

	length	**Symbols**	**digits**	**mix of letter cases**
Poor	<= 8	0	0	FALSE
ok	> 8	>1	>2	FALSE
Good	>12	>3	>2	TRUE
Excellent	>=16	>4	>4	TRUE

Part 3 Huffman Coding [Extension]

Write a Java program that builds Huffman codes for a short piece of text. Your program will likely consist of 3 classes Node, Tree and Main. Here are a few simplifications you should employ.

1. The text can be hard coded in you program. (Although loading from a file would be worthy of extra style credit)
2. You can store the resulting codes as Strings of 1s and 0s.

If you have forgotten Huffman codes here is a short video of Tom Scott explaining them. We will also do an example in class.

[10 marks]