COURSEWORK 2(25%)
Due date – 20th March, 23:59

1 Submission files:

Please submit one ***YourITaccount*.zip** file.
The file should contian:

- All relevant ***.java** files;
- A report in ***. pdf** format. Please:
  - Provide a short description of all ***.class** files
  - Draw the **UML** diagram, showing the relation between the classes.
  - Provide an example of execution of your code (you can copy it from Console, or take a screenshot and insert it as an image).
  - Project evaluation:
    - What went well? and what you have learnt?
    - What difficulties you faced when completing the project? and how you dealt with those difficulties?

2 Penalties:

5% penalty will be applied to any work submitted late. No submissions made later than 24 hours will be accepted, unless you have applied for Mitigating Circumstances.

3 Coursework Scenario - Gradebook:

Students on a course take three modules **CW001, EX002, CE003**. The modules have different assessment patterns:

- The mark for **CW001** is based only on the coursework, which consist of 3 homeworks and a final project; the marks are weighed as **50%** for homeworks (average mark) and **50%** for the project.
- The mark for **EX002** is based only on exam.
- - The mark for **CE003** is based on a coursework, the average of 4 homeworks, and an exam, weighed as **40%** for the coursework and **60%** for the exam.

All marks are entered as a percentage, integer values up to 100.

4 Task Description:

Your code should include the following classes:

1. **An abstract class Module**, which has attributes name, status and final mark.
   - **name** – to store the name of a module.
   - **status** – corresponds to the assessment pattern for the module: 0 – **coursework** only, 1 – exam only, 2 – coursework and exam.
   - **final mark** – to store the final mark for the module in a range from 0 to 100 (allows decimal values).

   The methods of the class should allow to set/get values of the attributes and to print the final mark.

2. **Three concrete classes**: one for each type of the modules (corresponding to status 0,1, and 2), subclasses of Module. In addition to attributes specified in the super class, these classes should have attributes for coursework marks, exams and the corresponding weights (where applicable).

3. **Class Student**: to store information about a student: ID, First name. Surname, and three modules as described in the scenario.
   From Student class you should be able to access (set/get) attributes of an object and all marks for the student; also it should have a printing method to print marks for all modules or for a particular module.

4. **Main class**: where you create instances of all Modules, and an **array** of Students instances, fill-in details of the students and their marks for every module, and print out marks in a tabular form (similar to CW1).
   - For printing marks you need to prompt a user whether to print marks for a single module (by name) or for all modules, in this case it is sufficient to print only the final marks (see the examples below).

5 Example of the printed results:

- Marks for all modules:

```
Name        Surname     CW001       EX002       CE003
John        Bradley     47.80%      67%         61.24%
Grace       Hightower   72.30%      62%         65.40%
Sophie      Turner      65.80%      79%         64.56%
```

- Marks for CW001:

Name	Surname	HWs	Project	Final Mark
John	Bradley	61.10%	34.50%	47.80%
Grace	Hightower	79.30%	65.00%	72.30%
Sophie	Turner	48.60%	83.00%	65.80%

- Marks for EX002:

Name	Surname	Final Mark
John	Bradley	67%
Grace	Hightower	62%
Sophie	Turner	79%

- Marks for CE003

Name	Surname	CW	Exam	Final Mark
John	Bradley	52.60%	67%	61.24%
Grace	Hightower	73.50%	60%	65.40%
Sophie	Turner	80.40%	54%	64.56%

6 Marking Criteria

Criteria	**Less than 40 (Fail)**	**40-49**	**50-69**	**Above 70**
Report project (30%)	1. Poor explanation of the code and outputs. 2. The write up lacks any understanding of the project. 3. No UML diagram presented	1. Satisfactory explanation of the code and outputs. 2. The write up demonstrates a basic understanding of the project. 3. Some awareness of key strengths and weaknesses of the project 4. An attempt made to present the UML diagram but some part are missing	1. Good explanation of the code and outputs. 2. The write up demonstrates a good understanding of the project. 3. Good evaluation of strengths and weaknesses of the project, with good justification. 4. UML presented in a good format	1. Excellent explanation of the code and outputs. 2. The write up demonstrates a complete and thorough understanding. 3. Excellent, reflective evaluation of the strengths and weaknesses of the project, including possible improvements. 4. Excellent design of the UML diagram, excellent presentation of the relationships between classes.
Code (70%)	1. No attempt made 2. Little or no use of object oriented principles 3. Submitted, failed to compile but looked like a real attempt	1. Some use of Object Orientation principles in JAVA. 2. Code is not easy to read. 3. Little comments in the code. 4. Output is good with minor errors.	1. Good use of Object Orientation principles with some advance features implemented. 2. Easy to understand the code flow	1. Nothing to fault. Well laid out, systematic comments, naming OK 2. Excellent use of Object Orientation principles. Code is easy to understand with proper comments.