Year 1

Name: Mohammed Shabbir Nalwala

Candidate Number: 229054413

C01105 Introduction to Object Oriented Programming
CW2

University Of Leicester

Project Report

Project Evaluation:

The Gradebook coursework scenario was a challenging but engaging project that allowed me to put my object-oriented programming skills to the test. Overall, I found the project to be a valuable learning experience, as it provided an opportunity to apply important programming concepts such as encapsulation, inheritance, and polymorphism to a real-world scenario.

- ### What went well? and what have you learnt?

→

1) What went well:

Overall, I'm quite happy with how my project turned out. I think I did a good job of designing and implementing a system for managing student modules and marks, and I feel like I accomplished most of what I set out to do. I felt that I was able to create a well-organized and modular code structure that made it easy to modify and extend the functionality of the program. Additionally, I was able to successfully implement the class hierarchy and inheritance relationships between the Module and Student classes. Here are a few things that I think went particularly well:

- Design: I spent a lot of time planning and designing my project before I started coding, and I think that paid off. By taking the time to think through the requirements and design a clear system architecture, I was able to avoid a lot of confusion and wasted effort later.
- Object-oriented programming: This project was a great opportunity for me to practice my OOP skills, and I feel like I learned a lot about creating effective class hierarchies, designing interfaces, and implementing polymorphism. I also got to work with some more advanced OOP concepts like abstract classes and interfaces, which was really rewarding.
- Testing: I made a conscious effort to write good unit tests for my code, and I think that helped me catch a lot of bugs early on. By running my tests frequently, I was able to catch problems before they got too serious and avoid a lot of debugging headaches down the line.

2) What I learnt:

Throughout the course of this project, I learned a lot about software development and programming in general. Here are a few things that I think were particularly valuable:

- Designing for flexibility: One of the biggest challenges I faced in this project was creating a system that could handle different types of modules with different assessment patterns. By designing my code to be flexible and extensible, I was able to create a system that could handle a variety of use cases without becoming overly complicated or hard to maintain.
- Debugging: Despite my best efforts to write good code and tests, I still ran into a fair number of bugs and issues along the way. By learning how to use debugging tools effectively and systematically working through problems, I was able to get my code working reliably and efficiently.

- Code organization: As my codebase grew larger, I had to become more thoughtful about how I organized my classes and methods. By adopting a consistent and logical approach to code organization, I was able to keep my codebase manageable and easy to work with.
- Time management: This project was a big undertaking, and I had to be careful about how I managed my time and prioritized tasks. By breaking down the work into manageable chunks and staying focused on my goals, I was able to complete the project on time and to a high standard.

• What difficulties you faced when completing the project? And how you dealt with those difficulties?

→

One of the biggest challenges I faced during the project was formatting the output in the specified tabular form. However, by breaking down the problem into smaller tasks and using appropriate formatting tools, such as printf statements and table libraries, I was able to achieve the desired output. I found this exercise to be very helpful in improving my debugging skills, as I had to test the formatting as I went along to catch any errors or inconsistencies.

Another difficulty I encountered was understanding some of the more complex concepts related to object-oriented programming. This included concepts like inheritance, polymorphism, and abstract classes, which were new to me. To overcome this challenge, I spent extra time reading textbooks, watching online tutorials, and asking for help from more experienced programmers. This allowed me to gain a better understanding of these concepts and how to apply them effectively in my project.

Lastly, another challenge I faced was debugging errors in my code. I found that some of my classes were not interacting properly with each other, causing unexpected behaviour in the output. To address this, I used debugging tools like print statements and breakpoints to isolate the source of the error and make corrections. I also found it helpful to take breaks from the project and come back to it with a fresh perspective. This allowed me to identify errors more easily and come up with new solutions to problems I was encountering.

A short description of all *.class files

1) Student.class:

The Student class represents a student and has the following fields:

- id (String): the ID of the student
- firstname (String): the first name of the student
- lastname (String): the last name of the student
- cw001 (CW001): an object representing the CW001 module for this student
- ex002 (EX002): an object representing the EX002 module for this student
- ce003 (CE003): an object representing the CE003 module for this student

The class also has a constructor method to create a new student object, which takes in the ID, first name, and last name of the student as arguments. It initializes the CW001, EX002, and CE003 module objects for the student and has getter and setter methods for each field.

This class also has a method printmodulemarks that prints out the student's marks for that module. It calls the calculateFinalMark method for the appropriate module object, and then prints out the student's name, surname, and final mark for that module. The printmodulemarks method also calculates and prints out additional information based on the module type, such as the average homework mark for CW001, and the average homework marks for CE003.

2) Module.class:

Module: This is an abstract class that serves as the base class for all the modules in the system.
It has three fields/attributes: "name" (the name of the module), "status" (the assessment pattern for the module: 0 – coursework only, 1 – exam only, 2 – coursework and exam), and "finalMark" (the final mark for the module).
The class has a constructor method that takes the name and status of the module as arguments and sets the corresponding fields. It also has getter and setter methods for all three fields.
Additionally, the class has an abstract method called "calculateFinalMark()" which does not have an implementation in the abstract class. Any class that extends the "Module" class must implement this method and define its own implementation.

3) CW001.class:

This class is a subclass and a concrete class that extends the Module and represents a module where the final mark is determined by a weighted average of homework marks and a project mark. It has private fields homeworkMarks (an array to store the homework marks) and projectMark (to store the project mark). It overrides the abstract method calculateFinalMark in Module to calculate the final mark as the average of the homework marks (weighted 50%) and the project mark (weighted 50%). It also has accessor and mutator methods for homeworkMarks and projectMark as well as for finalMark.

4) EX002.class:

This class is also a subclass and a concrete class that extends the Module and represents and represents an exam-only module. It contains a variable to store the exam mark and includes methods to get and set the exam mark. It overrides the abstract method for calculating the final mark to simply set the final mark to the exam mark.

5) CE003.class:

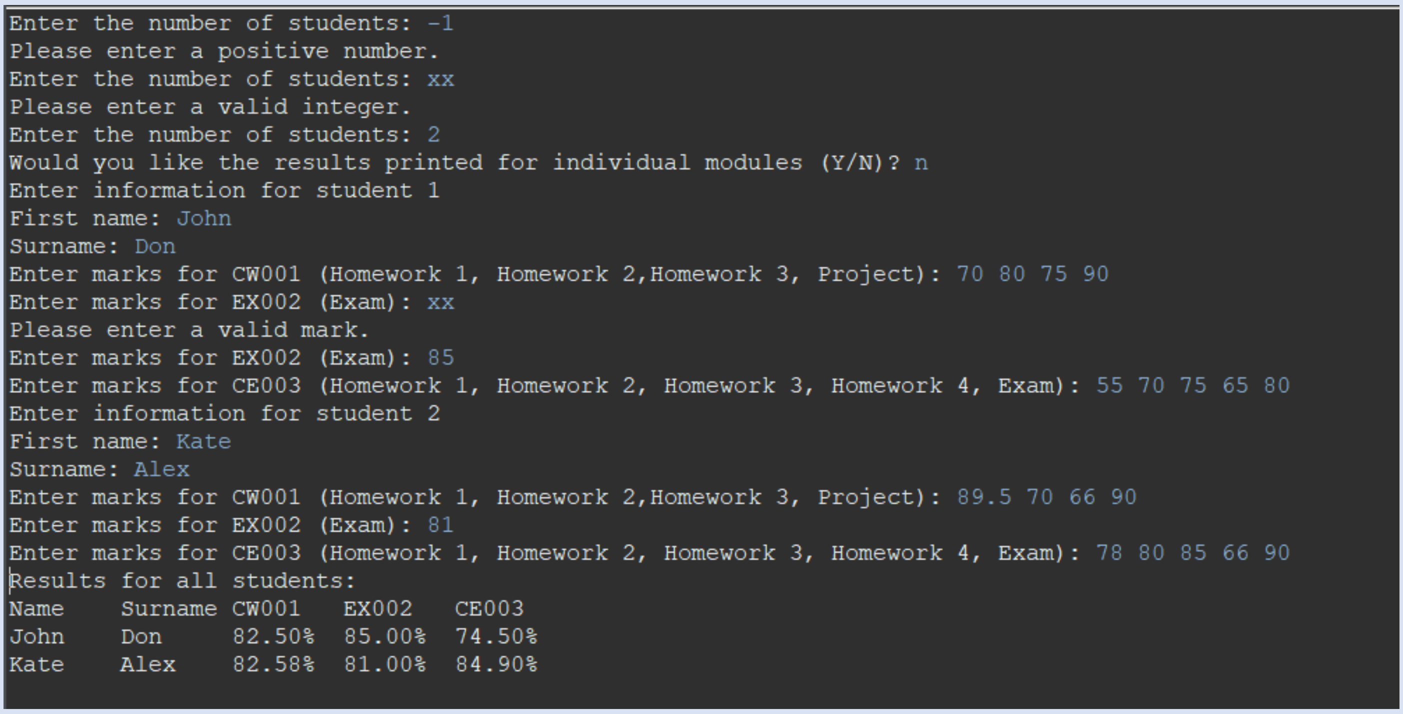
This is a concrete class that extends the Module class and represents a coursework, project, and exam module. It has private fields homeworkMarks (an array to store the homework marks) and examMark (to store the exam mark). It overrides the abstract method calculateFinalMark in Module to calculate the final mark as a weighted average of the homework marks (weighted 40%) and the exam mark (weighted 60%). It also has accessor and mutator methods for homeworkMarks and examMark as well as an additional getter and setter for finalMark.

6) Main.class:

The Main class is the primary class of the program, which is responsible for handling user input and controlling the flow of the program. It consists of a main() method, which is the entry point of the program, and it uses a Scanner object to read user input from the console.
The Main class begins by prompting the user to enter the number of students and verifying that the input is a positive integer. Then, it creates an array of Student objects with a size based on the user's input.
Next, the user is asked whether they want to print individual module results for each student or just the final results. The program then loops through each student in the array, prompting the user for information such as their first and last name, and module marks for each of the three modules - CW001, EX002, and CE003.
Each student's module marks are stored in their respective objects, and the final mark for each module is calculated using the calculateFinalMark() method. If the user chooses to print individual module results, the program will display the results for each student. Otherwise, the program will print the final results for all students in a tabular format. The Main class serves as the driver for the program and provides the user interface for input and output.

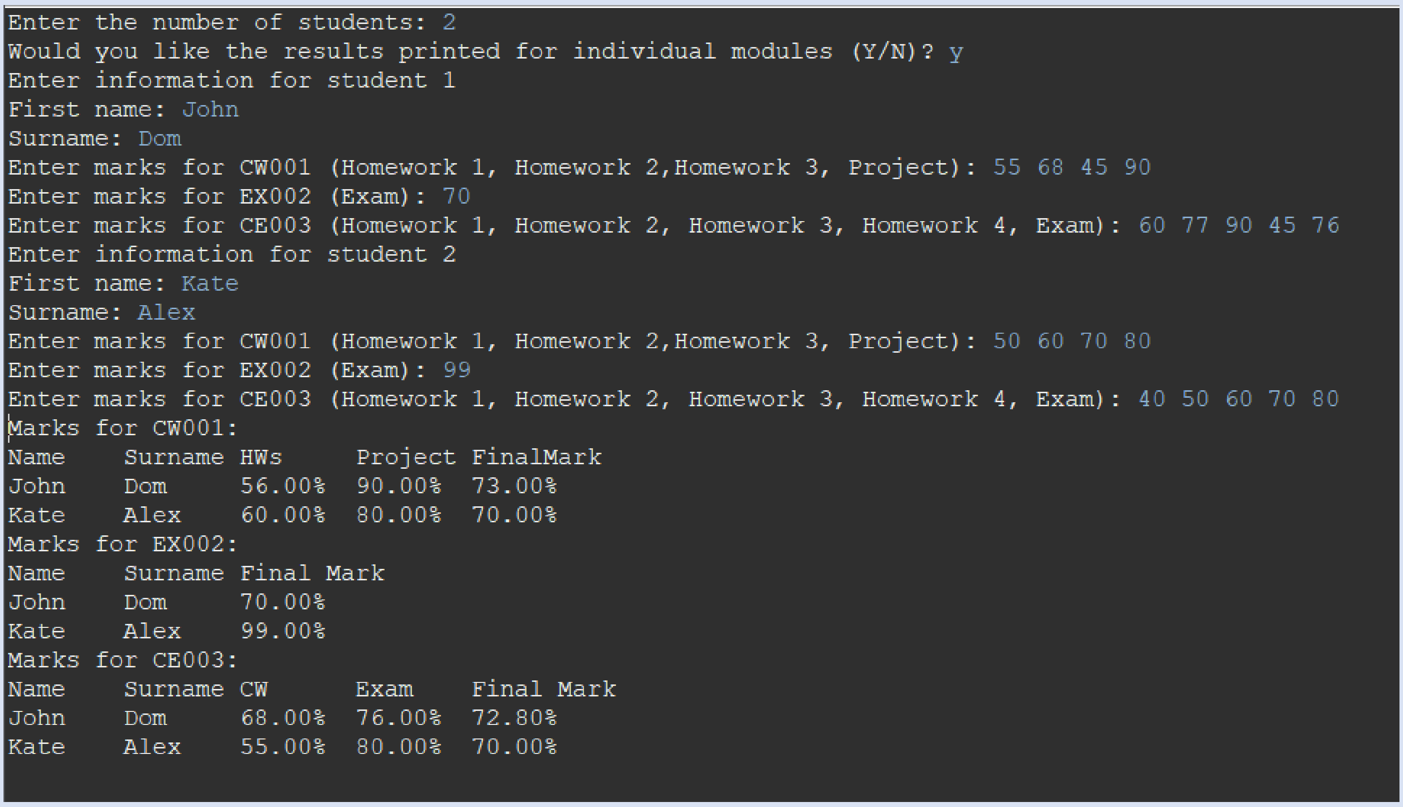
Examples of execution of my code:

Case-1) When marks for all module wants to be printed by the user, plus some try-catch instances:



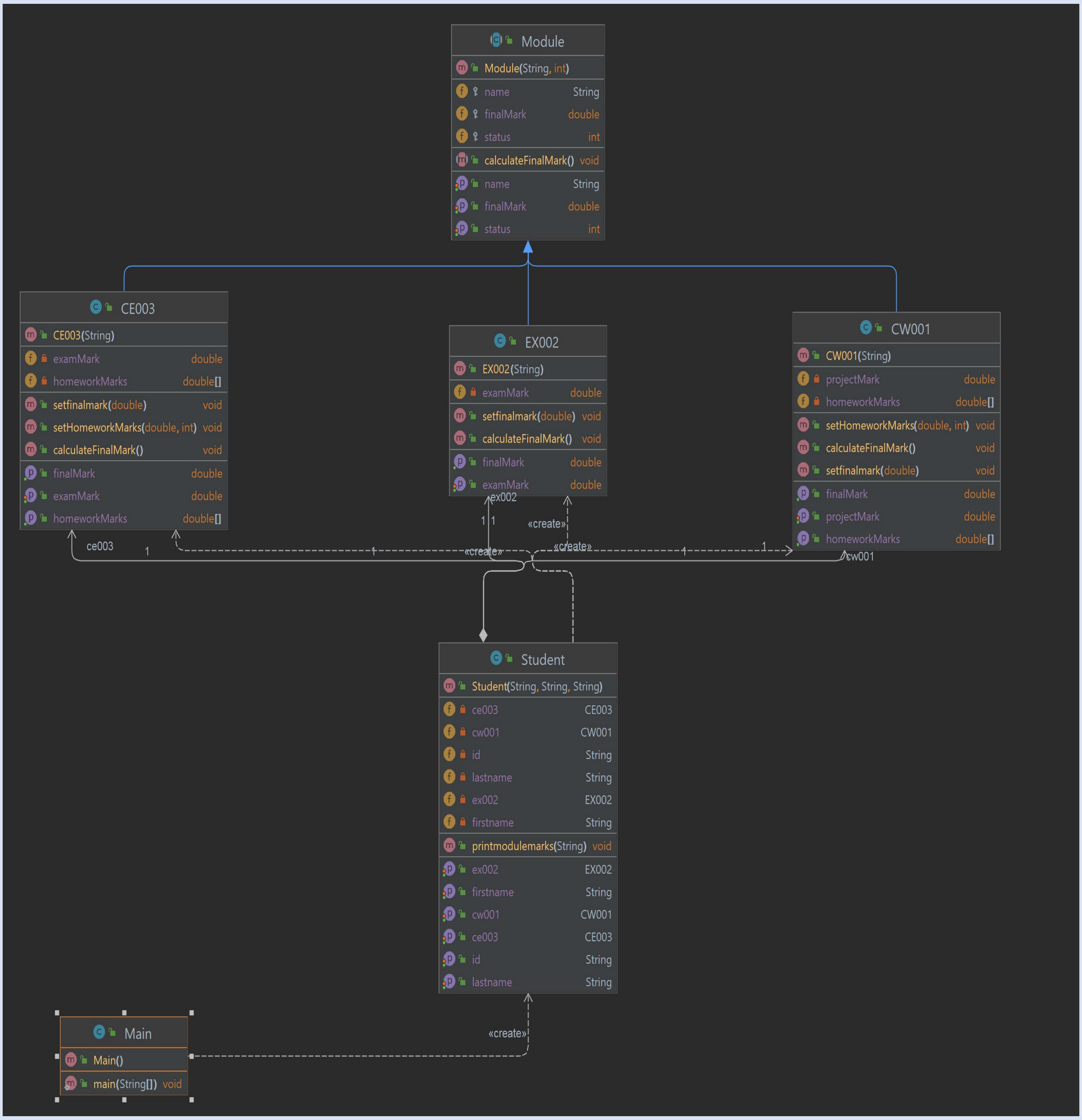
```
Enter the number of students: -1
Please enter a positive number.
Enter the number of students: xx
Please enter a valid integer.
Enter the number of students: 2
Would you like the results printed for individual modules (Y/N)? n
Enter information for student 1
First name: John
Surname: Don
Enter marks for CW001 (Homework 1, Homework 2,Homework 3, Project): 70 80 75 90
Enter marks for EX002 (Exam): xx
Please enter a valid mark.
Enter marks for EX002 (Exam): 85
Enter marks for CE003 (Homework 1, Homework 2, Homework 3, Homework 4, Exam): 55 70 75 65 80
Enter information for student 2
First name: Kate
Surname: Alex
Enter marks for CW001 (Homework 1, Homework 2,Homework 3, Project): 89.5 70 66 90
Enter marks for EX002 (Exam): 81
Enter marks for CE003 (Homework 1, Homework 2, Homework 3, Homework 4, Exam): 78 80 85 66 90
Results for all students:
Name    Surname CW001   EX002   CE003
John    Don     82.50%  85.00%  74.50%
Kate    Alex    82.58%  81.00%  84.90%
```

Case-2) When marks for every module individually wants to be printed by the user:



```
Enter the number of students: 2
Would you like the results printed for individual modules (Y/N)? y
Enter information for student 1
First name: John
Surname: Dom
Enter marks for CW001 (Homework 1, Homework 2,Homework 3, Project): 55 68 45 90
Enter marks for EX002 (Exam): 70
Enter marks for CE003 (Homework 1, Homework 2, Homework 3, Homework 4, Exam): 60 77 90 45 76
Enter information for student 2
First name: Kate
Surname: Alex
Enter marks for CW001 (Homework 1, Homework 2,Homework 3, Project): 50 60 70 80
Enter marks for EX002 (Exam): 99
Enter marks for CE003 (Homework 1, Homework 2, Homework 3, Homework 4, Exam): 40 50 60 70 80
Marks for CW001:
Name    Surname HWs     Project FinalMark
John    Dom     56.00%  90.00%  73.00%
Kate    Alex    60.00%  80.00%  70.00%
Marks for EX002:
Name    Surname Final Mark
John    Dom     70.00%
Kate    Alex    99.00%
Marks for CE003:
Name    Surname CW      Exam    Final Mark
John    Dom     68.00%  76.00%  72.80%
Kate    Alex    55.00%  80.00%  70.00%
```

UML Class Diagram:


Module
Module(String, int)
name String
finalMark double
status int
calculateFinalMark() void
name String
finalMark double
status int
CE003
CE003(String)
examMark double
homeworkMarks double[]
setfinalmark(double) void
setHomeworkMarks(double, int) void
calculateFinalMark() void
finalMark double
examMark double
homeworkMarks double[]
EX002
EX002(String)
examMark double
setfinalmark(double) void
calculateFinalMark() void
finalMark double
examMark double
CW001
CW001(String)
projectMark double
homeworkMarks double[]
setHomeworkMarks(double, int) void
calculateFinalMark() void
setfinalmark(double) void
finalMark double
projectMark double
homeworkMarks double[]
ce003
1
ex002
1 1
«create»
«create»
«create»
1
1
1
cw001
Student
Student(String, String, String)
ce003 CE003
cw001 CW001
id String
lastname String
ex002 EX002
firstname String
printmodulemarks(String) void
ex002 EX002
firstname String
cw001 CW001
ce003 CE003
id String
lastname String
Main
Main()
main(String[]) void
«create»