Assignment Details

University Centre at Blackburn College

Name of School	School of Health, Science & Technology
Course Title	FdSc Computing
Module No & Title	ST25167 Advanced Programing Concepts
Tutor	Stuart Gregory
Assignment Number	ST25167-1
Assignment Title	Design and implement a class room attendance system using OOP principles
Weighting	50%
Due Dates	26th November 2019 by 17:00
Word Count or Equivalent	700 words plus implementation and demonstration.

Learning Outcomes assessed in this assignment

1. Utilise a range of established modelling techniques in the design, development and documentation of a software system.
2. Apply the object-orientated paradigm to the development of complex systems.
3. Critically evaluate and apply appropriate design patterns to solve system level problems.
4. ~~Demonstrate critical understanding of subject terminology and concepts giving appropriate comparative examples in specific contexts where appropriate.~~

Task(s)

Design, implement and demonstrate a program given the scenario in appendix A. This is a group assignment with all group members working on aspects of the design documentation and code. Marks will be given for each element that an individual student produces and demonstrates however the design documentation will be 1 mark per group.

Design documentation (700 words plus designs):

All design documentation should be in a formal report format, with a clear introduction. Any research undertaken or resources consulted should be discussed and/or presented in a bibliography. The designs should be presented in the appendices and clearly referred to from the body of the text. The following should be included:

- Identify any assumptions you have had to make (this is NOT information given in the brief)
- Discuss any research or resources you have used throughout the development, with appropriate citations.
- Given the brief evaluate the program clearly identify any areas of the brief not achieved or issues yet to be resolved
- A bibliography (all resources consulted)
- Formal UML diagrams depicting such as use cases, class diagrams and state diagrams where applicable to the scenario, (in the appendices). This should be as complete as possible and reflect the final design. They main contain a number of elements that are not implemented.

Implementation:

The implemented program should demonstrate an appropriate use of the OOP programming concepts, as detailed in the marking criteria, given the scenario in appendices A. Credit will not be given for extending the brief or producing something unrelated.

All program files should include a clear header that includes at least the authors name. The (main) program name should be as specified in the brief. The code should be well structured and commented and follow best practice. The majority of classes should demonstrate some form of effective testing (commented out, output statements, main methods in each class use of Java

It is not expected that all designed elements are implemented, there should be a focus on demonstrating the elements given in the marking criteria and the overall functionality of the application. However, the implementation should NOT be a collection of unrelated or unconnected classes nor should it be significantly different from the design documentation (which should be updated as work progresses). It is expected that the implementation may contain a number of supporting classes to demonstrate the requested functionality and OOP concepts.

Demonstration:

The program will be demonstrated as a group to the module tutor, and in some instances a second marker. Where applicable test data will be provided by the tutor, unless otherwise stated in the brief. Students will be expected to explain specific aspects of their code and should do so using appropriate technical language.

Submission Requirements

Your report is targeted at a word count of 700 words (excluding bibliography and appendices). You must keep to this target and you must include the actual word count at the end of your submission. Submissions that are more than 10% above or below the target will be rejected unmarked. Appropriate appendices and references are not included in this word count.

Your written work must be submitted to UCBC via Turnitin on Moodle by the due date. Your work must be word-processed. All references should be formatted using the UCBC Referencing Guide (Harvard style). You MUST keep a copy of your work in case of loss or damage to the original. Work submitted late will only be accepted if your course tutor has granted permission in advance, and an extension form has been completed.

It is your responsibility to ensure that Moodle and the module tutor have received your submission. You are advised to keep copies of email receipts received and make no edits to your documents until you have confirmation your submission is successful. If you have any doubt as to your work being successfully submitted, you should present your work to the module tutor by **9am on the next working day**. **DO NOT** email your work to your tutor unless specifically asked to.

Marking Criteria

LO	Assessed Element	Weight
1	Design Document 700 words: ☐ Research and resources ☐ Evaluation of implemented solution ☐ Bibliography (excluded from word count)	15%
1	Applicable and appropriate range of UML diagrams: ☐ Use cases ☐ Class diagrams (high and low level) accurately depicting requested OOP concepts (see list in implementation criterion).	25%
2	Implementation: Clear demonstration of the following OOP concepts relevant to the given context: ☐ Composition and/or Inheritance (basic OOP) ☐ Enumerated Types ☐ Interface class ☐ Abstract class and methods ☐ Method overriding and/or polymorphism ☐ Application of some aspect of the collections API ☐ Effective application of design patterns	35%
2	Demonstration: ☐ Program compiles and runs ☐ Functionality as requested and accurate results where applicable ☐ Clear rationale and understanding of what has been implemented ☐ Effective development and testing process applied	25%

Total:	**100%**

See supplementary sheet for marking criteria descriptions at each grade band.

Late Submissions

Students who do not submit their coursework by the published deadline, without an agreed extension, will have up to **72 hours (3 days)** to make their submission. Where late submission deadline falls on a Saturday or Sunday students should submit their work no later than 9am on the following Monday. Submissions received during this three-day period will result in a reduced grade as shown in the table below. After this period no submissions will be accepted and will be recorded as an F4 zero grade.

Quality of Work	Grade Awarded	Quality of Work	Grade Awarded
A+	B+	C-	D-
A	B	D+	F1
A-	B-	D	
B+	C+	D-	
B	C	F1	F2
B-	C-	F2	F3
C+	D+	F3	F4
C	D	F4	

Appendix A:

Design and implement a simple class register system to keep track of student attendance.

Main class name: "**StudentAttendenceTracking**"

Requirements:

Functional specification

- Each course has a number of modules and each module can have one to many sessions
- Sessions have a specific date, start time and duration (in whole hours)
- Where a session is repeated on a specific day at a specific time, the day is recorded and a new register is created for each instance.
- Sessions can be of type: Lecture, Lab or Seminar
- A single tutor is associated with each taught session
- A student may attend more than one session
- A student's attendance is recorded as:
    - Present
    - Absent with notification
    - Absent without notification
    - Sick
    - Cancelled (this would apply to all students attending a specific session)
- A text based interface should be design that enables:
    - A register to be marked
    - A record of attendance to be viewed for each module's session
    - A record of attendance to be viewed for each student, including associated course and module
    - The session's details to be viewed including a list of attached students.

None-functional specification

- The system should enable effective error handling with feedback and support
- The system should provide a minimal demonstration interface only
- The system should be designed with extensibility in mind.

Technical Specification

- The system should be developed using only the core Java 8 libraries using OOP principles
- The system should be able to be complied and run form the Eclipse Oxygen IDE
- The system can be stored and re-instated from a flat-file system.