University of Wolverhampton
Faculty of Science and Engineering
Department of Mathematics and Computer Science

Module Assessment

Module	5CS019 Object Oriented Design and Programming
Module Leader	Dr Adeel Rafiq
Semester	1
Year	2024-25
Assessment	Resit assignment **What is covered:** classes, objects, inheritance, polymorphism, encapsulation and data hiding; selection, repetition and boolean expressions; numeric calculations, collections and basic array handling including finding counts, totals and maximum values. DB CRUD. Testing. Class diagram.
% of module mark	100%
Hand-in – what?	The requirements for each deadline are covered below
Hand-in- where?	To be submitted via Canvas by 12:00 on the given dates. Submit your source code, reports and diagram on Canvas. Place your files in a folder and zip it and submit one zip file. Make sure your diagram is either a pdf or an image.
Pass mark	40% is required overall to pass the module. Each portfolio task has an indicative weighting. However, the final grade will not be calculated entirely mechanically. A degree of academic judgement will be used to assess how well you have met the learning outcomes overall. You must attempt and submit all portfolio tasks.
Method of retrieval	Resit portfolio tasks in the winter resit period. These will not necessarily be the same as the original tasks, but variants of them.
Feedback	During scheduled sessions

Introduction

This is an individual coursework.

The work is divided in 3 parts that progress from one to the other.

Working in stages, you can develop your program using **incremental development** (bit by bit, not all at once). Develop your program a little bit at a time and keep testing it. Save a working version before making the next set of alterations.

All code should be well-designed, well-laid out and commented.

Read this document several times before starting work, to ensure that you understand what is involved.

Overview

In this semester's assessment, you will develop an application to manage competitors in various types of competitions. Competitions could range from archery to eSports. Each competitor will receive a fixed number of scores, and an overall score will be calculated based on these scores to determine their performance.

Your task is to create a Java application that stores competitor details in a MySQL database, retrieves this data, and produces a comprehensive report including details of all competitors, the top performer, and various summary statistics.

Example Competitor Data

Imagine the following competitors in a shooting competition:

Competitor ID	Name	Level	Score1	Score2	Score3	Score4	Score5
200	Alice Green	Beginner	4	3	5	2	4
201	Bob Brown	Intermediate	3	4	4	5	4
202	Carol White	Advanced	5	5	4	4	5
203	David Black	Advanced	4	4	5	5	4

Example Report
The generated report should include:

Competitor Table:

```
Competitor ID     Name              Level         Scores      Overall
200               Alice Green       Beginner      4 3 5 2 4   3.6
201               Bob Brown         Intermediate  3 4 4 5 4   4.0
202               Carol White       Advanced      5 5 4 4 5   4.6
203               David Black       Advanced      4 4 5 5 4   4.4
```

Full Details for **CompetitorID** 200:

CompetitorID 200, name Alice Green.

Alice is a Beginner and received these scores: 4, 3, 5, 2, 4.

This gives her an overall score of 3.6.

Short Details for ***CompetitorID*** 202:

CN 202 (CW) has an overall score of 4.6.

Statistical Summary:

- Total number of competitors: 4
- Competitor with the highest score: Carol White with an overall score of 4.6
- Frequency of individual scores:

Score:	2	3	4	5
Frequency:	1	6	10	7

Part One – Develop the Competitor Class

Don't use MySQL or any sort of graphical user interface in this stage. Just develop your competitor class, without the list of scores. Using a main method, create several objects of the class, and test all your methods, which you will use in the next stage.

Details for each competitor should be held in a class named either ***XXXCompetitor*** (where ***XXX*** is your initials) or some more specific name e.g., IceSkater. Develop this class, according to the requirements that are listed below. Some of these requirements are described fully, and in other cases you have to make some of your own decisions. You can base your competition on a known sport or game, or just invent values.

When you make your own decisions, don't make the same decisions as your friends. You will lose marks if you hand in work in which all design decisions and code structures are identical to those of other students.

Your competitor class - Instance variables

Create a ***Competitor*** class to hold the details of each competitor.

Competitor Class – Instance Variables

For each competitor, store:

- Competitor ID – integer
- Competitor name – use a Name class
- Competition level – fixed values (e.g., Beginner, Intermediate, Advanced)

- Extra attribute of your choice (e.g., country, age)

Competitor Class - Methods

The Competitor class should have (use **exact** method names please):

- A ***Constructor*** and get and set methods.
- A method which will be used later, to get the overall score. You will change it in the next stage, but for now it should look exactly like this:

  public double getOverallScore() { return 5; }

- A ***getFullDetails()*** method which returns a String containing all the details of the competitor (including your extra attribute) as text. It should be clear what all the values are. It doesn't have to look exactly like the example below.

  Competitor number 100, name Keith John Talbot, country UK.
  Keith is a Novice aged 21 and has an overall score of 5.

- A ***getShortDetails()*** method which returns a String containing a ONLY competitor number, initials and overall score. The format should look **exactly** like the line below, (with different data). 'CN' stands for 'competitor number'.

  e.g. CN 100 (KJT) has overall score 5.

Part Two – Working with MySQL and Arrays

Modify the ***Competitor*** class to include individual scores:

Changes:

- Add an instance variable for an array of integer scores (e.g., 5 scores).
- Provide a method ***getScoreArray()*** to return the array of integer scores.
- Update the ***getOverallScore()*** method to calculate the overall score based on the scores and the level. Consider:
  - Average of top ***n*** scores where ***n*** is the level number.
  - Weighted average of scores.
  - Average disregarding the highest and lowest scores.

Database Integration:

- Create a MySQL database named ***CompetitionDB*** with a table ***Competitors*** that matches the structure shown above.
- Implement methods in the ***Competitor*** class to read from and write to the MySQL database using JDBC.

Part Three – Using MySQL and Generating Reports

Introduce a ***Manager*** class and a ***CompetitorList*** class:

Tasks:

- Connect to the ***CompetitionDB*** database and retrieve competitor data.
- Produce a final report including:
  - A table of competitors with full details.
  - Details of the top performer.
  - Summary statistics, including frequency reports of individual scores.

User Interaction:

- Allow the user to enter a Competitor ID and display the short details if the ID is valid.

Error Handling:

- Implement basic error handling to check for invalid data and connection issues with the MySQL database.

Submission Requirements:

1. **Decisions:**
   - Detail the extra competitor attributes chosen.
   - Describe how the overall score is calculated.
2. **Status Report:**
   - Briefly state whether the application meets the specifications or is incomplete.
3. **Known Bugs and Limitations:**
   - List any known bugs and limitations of your application.
4. **Tests:**
   - Include a table of tests performed.
5. **Diagrams:**
   - Provide a class diagram showing all classes with their instance variables, methods, and associations.
6. **Javadoc Comments:**
   - Write Javadoc style comments for the ***CompetitorList*** class and include the generated HTML documentation.

How to submit

Submit your source code, reports, diagrams, and Javadoc HTML documentation as a single zip file on Canvas. Ensure your diagrams are in PDF or image format.

Marking scheme

See separate marksheet. If out of 5:

5 – Outstanding, perfect in every way
4 – Mostly excellent with one or two minor problems
3 – Good
2 – Adequate
1 – Poor

Your completed work for assignments must be handed in on or before the due date. ***You must keep a copy or backup of any assessed work that you submit. Failure to do so may result in your having to repeat that piece of work.***

Penalties for late submission of coursework
Standard Faculty of Science and Technology arrangements apply.
ANY late submission (without valid cause) will result in 0 marks being allocated to the coursework.

Procedure for requesting extensions
If you have a valid reason for requiring an extension you must request an extension using e:vision. **Requests for extension to assignment deadlines should normally be submitted at least one week before the submission deadline and may be granted for a maximum of seven days (one calendar week).**

Retrieval of Failure
A pass of 40% or above must be obtained overall for the module (but not necessarily in each assessment task). **Where a student fails a module they have the right to attempt the failed assessment(s) once, at the next resit opportunity (normally July resit period). If a student fails assessment for a second time they have a right to repeat (i.e. RETAKE) the module.**

NOTE: STUDENTS WHO DO NOT TAKE THEIR RESIT AT THE NEXT AVAILABLE RESIT OPPORTUNITY WILL BE REQUIRED TO REPEAT THE MODULE.

Mitigating Circumstances (also called Extenuating Circumstances).
If you are unable to meet a deadline or attend an examination, and you have a valid reason, then you will need to request via e:vision **Extenuating Circumstances.**

Feedback of assignments
You will be given feedback when you demonstrate your work.

You normally have **two working weeks** from the date you receive your grade and feedback to contact and discuss the matter with your lecturer. See the Student's Union advice page http://www.wolvesunion.org/adviceandsupport/ for more details.

Registration

Please ensure that you are registered on the module. You can check your module registrations via e:Vision You should see your personal tutor or the Student Support Officer if you are unsure about your programme of study. The fact that you are attending module classes does not mean that you are necessarily registered. A grade may not be given if you are not registered.

Cheating
Cheating is any attempt to gain unfair advantage by dishonest means and includes **plagiarism** and **collusion.** Cheating is a serious offence. You are advised to check the nature of each assessment. You must work individually unless it is a group assessment.

Cheating is defined as any attempt by a candidate to gain unfair advantage in an assessment by dishonest means, and includes e.g. all breaches of examination room rules, impersonating another candidate, falsifying data, and obtaining an examination paper in advance of its authorised release.

Plagiarism is defined as incorporating a significant amount of un-attributed direct quotation from, or un-attributed substantial paraphrasing of, the work of another.

Collusion occurs when two or more students collaborate to produce a piece of work to be submitted (in whole or part) for assessment and the work is presented as the work of one student alone.