COMP1815 (2022/23)	**JVM Programming Languages**	**Contribution: 100% of course**
Course Leader: **Dr Markus Wolf**	**Practical Coursework**	**Deadline Date:** **Thursday 14/12/2023**

Plagiarism is presenting somebody else's work as your own. It includes: copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing coursework from another student and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with according to the procedures set down by the University. Please see your student handbook for further details of what is / isn't plagiarism.

All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

Your work will be submitted for plagiarism checking. Any attempt to bypass our plagiarism detection systems will be treated as a severe Assessment Offence.

Coursework Submission Requirements

- An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of **Thursday 14/12/2023 23:30 UK Time** using the link on the coursework Moodle page for COMP1815.
- There are limits on the file size (see the relevant course Moodle page).
- Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.
- **All courseworks must be submitted as above. Under no circumstances can they be accepted by academic staff**

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences.
See https://www.gre.ac.uk/policies/undergraduate-and-postgraduate-taught

Coursework Specification

This assignment consists of three parts:

- **Part A (implementation) will be completed in a group**
- **Part B (group report) must be completed in a group**
- **Part C (hackathon) must be completed in a group**
- **Part D (individual report) must be completed individually**

Please read the entire coursework specification before starting your work.

Project Management System

You have been contacted by a company which specialises in managing software development projects. Projects are commissioned by customers (who could be individuals but are companies in most cases). The company has several teams of project managers and developers who are experts in a variety of technologies (every team is managed by a team leader). The company's main office is in London and some teams are based there, but others are at geographically dispersed locations (e.g. Germany, China, Brazil).

The current provision for managing the projects is as follows:

1. A project manager is assigned a project
2. The project manager uses Pert and Gannt charts to outline the tasks, responsibilities, timeframes, etc. of the project
3. The project manager instructs the team leaders involved in the project on their tasks and timeframes

While the current provisions work, they are not very effective, and it is often difficult for a project manager to know what the exact state of the project is.

To aid the project managers in the project coordination, the company has asked you to develop a software system which will facilitate this task.

The system will need to provide the following functionality:

- Set up projects
- Divide projects into tasks
    - Duration and sequence of tasks can be defined (a task can have none, one or more successor tasks)

Currently, the only type of user of the system will be a project manager (administrator).

There are strict requirements about the technologies and architecture to be implemented. These are detailed in the deliverables section.

Deliverables

Part A – Implementation (40%)

Organise yourselves into groups consisting of 3 or 4 students. Exceptionally, you can work individually, but please approach the tutor if you are thinking of doing this. If you cannot find a group to join, your tutors will allocate you to a group. **All groups are final on the 26th of October 2023** and no changes will be made to groups after this date.

NOTE: Please note that each individual student's grade within the group will be adjusted based on the contribution made to the implementation and group report

a) **Java GUI for Project Management – Java (10%):**

Use Java to create a desktop application with a graphical user interface which enables a user to set up, edit and delete projects and their tasks.

The application should look pleasant and be easy to use.

b) **Domain Classes and Graph - Kotlin (10%)**

Create domain and entity classes to represent the tasks in a project as a graph. Create an object-oriented structure supporting the Java GUI application. You should apply separation of concern to ensure that the Java GUI application contains only the user-interface related functionality, and all other responsibility is assigned to the domain and entity classes. These classes should be implemented in Kotlin and integrated into the Java GUI.

If you wish, you can implement persistence for the project and team data, so you don't have to reenter this every time you run the application. If so, it is up to you to decide how you wish to save the information (e.g. save it to file or to database).

c) **Adjacency Matrix - Kotlin (10%)**

An adjacency matrix allows you to represent a graph (directed of undirected).
Assume that the graph has n vertices. You can use a two-dimensional $n \times n$ matrix to represent edges. Each element in the list is 0 or 1, where there is a 1 if there is an edge from one vertex to another or 0 if there is not.
In this assignment, a project is represented as a directed graph and you should write an algorithm which converts the object-oriented graph structure into an adjacency matrix.

This algorithm should be integrated into the Java GUI, so that it is possible for a user to view the adjacency matrix of a given project.

d) Version Control (10%)

You must use Git to submit the progress of your code **at least once per week**. On the course Moodle page you will find instructions on how to use Azure DevOps to set up a Git repository for your code and how to integrate this with Android Studio and Visual Studio.

You must include all group members to ensure that you can work collaboratively on the code and you must also include both tutors (Markus Wolf and Rafael Martinez-Torres) to have viewing permission.

Part B – Group Report (10%)

The group report is to be completed by the group**.**

NOTE: Please note that each individual student's grade within the group will be adjusted based on the contribution made to the implementation and group report

Write a report consisting of **all** the following sections:

- **Section 1. (2%)** A **concise table** containing a checklist of the features you have been able to implement. Please refer to the features list given above in the specification. **Include a link to your Git repository.**
- **Section 2. (3%)** Screenshots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.
- **Section 3. (3%)** Code listing of any code files you have written. You do not need to include generated code. Please clearly label the code, so it indicates the source file and programming language.

  Please note that you must include the actual code, not just file names.
- **Section 4. (2%)** Include a completed CW Contributions Form (found at the end of this specification).

Part C – Hackathon (30%)

This hackathon is to be completed by the group**.**

During the hackathon, you will be required to solve and submit several coding exercises. Details of the hackathon will be provided on Moodle closer to the time.

Part D – Individual Report (20%)

This report is to be completed **individually.**

Write a report consisting of **all** the following sections:

- **Section 1. (8%)** – (approx. 500-800 words) On this module you have been taught object-oriented and functional programming paradigms, using two different languages (i.e. Kotlin, Scala). Critically compare these different paradigms and languages based on your own experience, complemented by more indepth research on these topics. You should include discussion of their suitability to different problems and what you perceive their strengths and weaknesses to be.

- **Section 2. (9%)**– (approx. 700-1,000 words) An evaluation of the evolution of your application. You should discuss any problems you had during implementation. You should be critical (both positive and negative) of your implementation. Be prepared to suggest alternatives. You should also include a reflection on how it was to work in a group and of your role within the team. Discuss lessons learnt, what you think went well and what you think could have been improved and how.

- **English Proficiency (3%)** – marks will be allocated to the level of language used in the individual report, including correct spelling and use of grammar

Demonstration (0% but see below)

The demonstration is carried out by the group.

You are required to prepare a brief video showing your implementation (one video per group). The duration should be approximately 5 minutes. There will be a dedicated Panopto coursework submission area – more details on this will be available on Moodle.

Your group must also attend a brief Q&A session with your tutor where you will be asked questions about your implementation and the code.

Failure to demonstrate will result in a failed assessment.

You will be allocated a time slot closer to the submission deadline.

Schedule of Submission

Deliverable	Date	Type	Group/Individual
Weekly Code Update	At least once a week	Git commit	Group
Hackathon	07/12/2023	Group programming session to solve coding exercises	Group
Code Q&A	14/12/2023	MS Teams	Group
Report and Code	14/12/2023	Submission on Moodle	Individual
Video Demonstration	18/12/2023	Submission on Panopto	Group

Each group is encouraged to create a group on MS Teams to enable working as a group. Although you can use this to communicate, to share files of code you are required to use version control to support the work as a group.

Notes on the Implementation

You **MUST** upload a ZIP file containing all of your source code (i.e. the folders containing the IntelliJ projects). If the resulting file is too large, then you can delete compiled code and libraries, but do not remove any source code files.

You **MUST** clearly reference code borrowed from sources other than the lecture notes and tutorial examples (e.g. from a book, the web, a fellow student).

Please include a file called "Group List.txt", which lists all the members of the group, including the name and id of each member.

Notes on the Report

The document should be submitted separately as a PDF document.

Notes on the Demonstration

You will demonstrate the implemented product as a group to your tutor by preparing a 5-minute video. **If you fail to demonstrate your work you will automatically fail the coursework**, irrespective of the quality of the submission. The video must be uploaded via the module's Panopto assessment submission link. There should be only one video submission per group.

The video you upload on Panopto should follow the follow naming convention – "Surname1,Surname2, etc."

You should include the surnames of all group members in the video name, so we can easily identify the group.

During the Q&A session you are expected to talk knowledgeably and self-critically about your code.

If you are unable to answer questions about the product you have developed, you will be submitted for plagiarism.

A schedule for the Q&A sessions will be made available on the module's Moodle page closer to the submission deadline.

Formative Assessment

Groups are encouraged to show the progress made with the implementation in the labs on a regular basis to get feedback on how the work progresses and, on the design, and implementation decisions taken at every stage of the assignment.

There are also weekly lab exercises, which train you in the knowledge required to complete the assessment. Completed exercises do not need to be uploaded but shown to the tutor for feedback.

Rubric

A rubric for the assignment is available as a separate document on the module’s Moodle page.

Grading Criteria

The implementation which consists of the group work accounts for 80% and the individual report accounts for 20%. There are multiple functionalities which comprise the development of the system. Each functionality section contributes 10% to the overall grade. The mark for each is awarded taking into consideration the quality and completeness of the implementation, as well as the assessment criteria specified below. Just because you implemented a particular requirement does not mean that you automatically get the full marks. The full marks are only awarded if the requirement has been implemented to outstanding quality, including software design, code quality, user interface, error handling, validation, etc. A poorly structured but working implementation of a requirement would attract a pass mark.

To achieve a pass (40%) you must have made a serious attempt to implement at least two functionality sections. The implementation must show some signs of attempting to focus on the assessment criteria given in this document. A relevant report must also be submitted. You must also have submitted some correct solutions to the hackathon coding exercises.

To achieve a 2:2 mark (above 50%) you must have made a serious attempt to implement at least three functionality sections. They must attempt to focus on the assessment criteria given in this document. A good report must also be submitted. You must also have submitted a good number of correct solutions to the hackathon coding exercises.

To achieve a 2:1 mark (above 60%) you must have made a serious attempt to implement at all functionality sections. They must address most assessment criteria given in this document. A very good report must also be submitted. You must also have submitted many correct solutions to the hackathon coding exercises.

To achieve a first class (above 70%) you must implement all requirements to a very high standard, or most to an outstanding level, in accordance with the assessment criteria given in this document. Submit an excellent report. Successfully meet most assessment criteria outlined below. You must also have submitted some mostly correct solutions to the hackathon coding exercises.

To achieve a very high mark (80% and above) you must implement all implementation requirements to an outstanding standard in accordance with the assessment criteria given in this document. Submit an outstanding report. Successfully meet all assessment criteria outlined below. You must also have submitted correct solutions to almost all the hackathon coding exercises.

NOTE: Failure to do your Demonstration will normally result in you being awarded 0% for the coursework. Even if your implementation and report are excellent and would be awarded a mark of 80% but you don't do a Demonstration then you may score 0% for the coursework.

Assessment Criteria

The Implementation

The following assessment criteria are used to determine the quality of your implementation and should be addressed in the development process:

- If you have incorporated features that were not explicitly asked for, but which add value to the application, they may be considered if you draw our attention to them.
- The application should look pleasant and be easy to use.
- Code structure – does your code demonstrate low coupling and high cohesion? Have you avoided hard coding (i.e. is your code stateless)? Have you reused external components? Have you minimised code duplication? How much impact would a further change of persistence medium have on your application?
- Quality of Design – how flexible is your application? How easy would it be to add in new functionality, or alter the presentation layer, or change the data source?
- Robustness of the application. Have you properly handled errors and validated input? Is there evidence of testing?
- Quality of code –
    - Is the code clear to read, well laid out and easy to understand?
    - Is the code self-documenting? Have you used sensible naming standards?
    - Is your code structure logical?
    - Have you commented appropriately?

The Report

The document should be clear, accurate, complete, and concise. State any assumptions you have made.

- Are all the required sections included and completed properly?
- Does the report give an accurate reflection of what you have achieved?
- Is the report clear and easy read? Does it follow the structure specified?
- Is the evaluation realistic and does it show that you have really thought about your implementation and the specified issues as well as how they may be enhanced to be ready for live deployment. Do you show insight into the complexities of software development?

The Hackathon

You will be submitting solutions to coding exercises on DOMjudge and will be graded by this system.

Coding solutions will be graded by the DOMjudge system and the solutions submitted must compile and match the output of the coded solution.

Demonstration

- You should be able to demonstrate the implementation level achieved in a clear logical and unambiguous manner without encountering errors. You must be able to show knowledge of your code and design choices.

Summative Feedback

Feedback on the final submission will be provided in written format and made available on Moodle within 15 working days of submission.

1. Coursework Contribution Form (completed as a group)

In percentage, please indicate the work contribution of each member. This should be agreed by all group members**. The total of all members work must add up to 100%**

Team member name	Student ID	individual overall work contribution (%)	Note
Student:			
Student:			
Student:			
Student:			
		Total 100%	