COMP1815 (2020/21)	**JVM Programming Languages**	**Contribution: 100% of course**
Course Leader: **Dr Markus Wolf**	**Practical Coursework (Hackathon)**	**Deadline Date:** **Thursday 10/12/2020**

Coursework Submission Requirements

- **An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of** Thursday 10/12/2020 **using the link on the coursework Moodle page for COMP1815.**
- **For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). An exception to this is hand written mathematical notation, but when scanning do ensure the file size is not excessive.**
- **For this coursework you must also upload a single** ZIP **file containing supporting evidence.**
- **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.
- Your work will not be printed in colour. Please ensure that any pages with colour are acceptable when printed in Black and White.
- **You must NOT submit a paper copy of this coursework.**
- **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 http://www2.gre.ac.uk/current-students/regs

Coursework Specification

This assignment consists of two parts:

- **Part A (implementation) will be completed in a group**
- **Part B (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
4. Communication and progress tracking is done through:
   a. face-to-face meetings with the local teams
   b. email/telephone/video conferencing with the geographically distant teams

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 and team leaders 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
- Set up teams
- Divide projects into tasks
  - Duration and sequence of tasks can be defined (a task can have none, one or more successor tasks)
  - Each task is assigned to particular team
- Register task progress. Project progress can be tracked

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 (70%)

Organise yourselves into groups consisting of 2 or 3 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.

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

   Use Java to create a desktop application with a graphical user interface which enables project managers to set up projects and manage teams and record progress on tasks.

   The application should look pleasant and be easy to use.

b) **Object-Oriented Design - Kotlin (10%)**

   Create domain and entity classes that 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

c) **Persistence and Lambda – Kotlin (10%)**

   Implement persistence for the project, team and task data, which makes it possible to save this data. It is up to you to decide how you wish to save the information (e.g. save it to file or to database). You should use Lambda expressions to manage the collections of data. This should be implemented in Kotlin and integrated into Java GUI.

d) **Critical Path - Kotlin (10%)**

   Implement an object-oriented component in Kotlin that can calculate the duration and finishing date of a project (based on its critical path) when provided with a project object. In project management, the critical path is the longest path of planned activities to the end of the project such that a delay to any of the tasks on the critical path will result in a delay of the overall project. You can find a description of the critical path method at https://en.wikipedia.org/wiki/Critical_path_method

e) **Integration – Kotlin/Java (10%)**

   Integrate the critical path component into the project management process, so that when a project is edited, the critical path is automatically calculated and the duration and finishing date are updated. A user should also be notified when progress on a task is recorded which would affect the critical path (e.g. a task took longer to complete than planned and this affects the critical path).

f) **Critical Path - Scala (10%)**

You implemented the critical path algorithm using an object-oriented approach in Kotlin. Now implement a critical path algorithm using a functional approach in Scala.

g) **Integration – Scala/Kotlin/Java (10%)**

Integrate the Scala implementation of the critical path algorithm into the Java/Kotlin application you developed. Ideally, it should be possible to choose whether the object-oriented or functional implementation should be used at run-time.

Part B – Report (30%)

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

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

- **Section 1. (10%)** – (approx. 700-1,000 words) On this module you have been taught object-oriented and functional programming paradigms, using three different languages (i.e. Kotlin, Scala and Clojure). Critically compare your experience with these different paradigms and languages. You should include discussion of their suitability to different problems and what you perceive their strengths and weaknesses to be.
- **Section 2. (10%)**– (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.
- **Section 3. (5%)** Screenshots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.
- **Section 4. (5%)** 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.

Demonstration (0% but see below)

You must demonstrate your implementation to the company which is represented by your tutor who will sign off the system. 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
Hackathon	05/11/2020	Online (MS Teams)	Group
Demonstration	26/11/2020	Online (MS Teams)	Group
Report and Code	10/12/2020	Submission on Moodle	Individual

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

Due to current social distancing restrictions, it will not be possible to hold a hackathon in the traditional sense. The hackathon will take the following format. On the 5th of November 2020, students are encouraged to engage in an intensive online group programming session to progress the implementation of the coursework. The tutors will set aside time blocks on that day when they are able to drop in and advise groups. Details on this will be published on the module's Moodle page.

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).

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. **If you fail to demonstrate your work you will automatically fail the coursework**, irrespective of the quality of the submission. If you have a legitimate reason for missing your demonstration, then you should wherever possible contact your tutor in advance and arrange an alternative time slot. It is entirely your responsibility to contact your tutor and arrange a new demo if you miss your slot for a legitimate reason.

You are expected to talk knowledgeably and self-critically about your code. The demonstration will take place online, but it is your responsibility to make sure that you can demonstrate the code running on the computer you use to participate in the online demonstration. Please allow yourself plenty of time to get your system working on the computer you will use to demonstrate the work before your demonstration time.

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

A schedule for the online coursework demonstrations will be made available on the module's Moodle page closer to the submission deadline.

Grading Criteria

The implementation which consists of the group work accounts for 70% and the individual report accounts for 30%. 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 four 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.

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

To achieve a 2:1 mark (above 60%) you must have made a serious attempt to implement at least six functionality sections. They must address most assessment criteria given in this document. A very good report must also be submitted.

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.

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.

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?

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.