CSYM026: Software Engineering (Draft version)			
Date of Issue	**26/02/2024**	**Last Date for Submission:**	**24/05/2024 By 23:59:59**
Weight 100%		Module Tutor:	Dr Suraj Ajit and Dr Ali Sadiq

Coursework Overview

This coursework contributes 100% of your mark for the module. **This is a group assignment**. You will need to appoint a group leader who will create a private GitHub account and invite group members to collaborate on the assignment. There should be evidence of each member's contribution to the work regularly. This evidence will essentially be via git logs that will need to be submitted as part of your final submission. Please ensure to use your official names in your git commits.

You are required to develop a simulation of customers collecting parcels from a depot. You should work in the groups assigned to you by your tutor, with the group collaborating on the design, implementation and testing of the application.

The coursework has 2 sets of requirements:

- functional requirements, which describe WHAT the application should do
- software engineering requirements, which describe HOW you should develop the application.

The basic simulation is developed in 2 stages, the first concentrating on planning and testing, the second adding in threads and patterns. You will also write an individual report which gives technical details of some sections of the application, summarizes the contribution of each team member, and discusses the development of the whole application.

More precise details are outlined in the following sections.

Your mark for the group will be the group mark scaled by your contribution to the group. This contribution will be assessed by the lecturer according to your report and the reports of the other members of your team. The key here is that there should be evidence of collaboration among group members and each member contributing to the development.

For any decisions that your group makes, these decisions and the resulting code should not be identical to those of other groups. You are welcome to discuss the work with other students, but not to copy.

Please read the complete document before starting. In your report at the end of stage 2, you are expected to discuss the methodologies and resources used for both stages.

The First Stage of the Application.

Functional Requirements Stage 1

This application has been designed to be complex enough to enable you to try out various software engineering features, whilst small enough to fit in the time available. You are required to develop a simulation of customers collecting parcels from a depot. Details of the customer (apart from their name) are not needed for the purposes of this program.

This first stage concentrates on providing the basic functionality, planning and testing. You are to adopt a Test-Driven Development (TDD) approach.

Your depot only contains parcels waiting to be collected. Parcels have a unique ID, dimension (length, width, height), weight, and the number of days that they have been in the depot for. For your program, you will need at least 20 parcels. You can choose whether to use integers or real numbers for the size and weight.

Each customer in the queue has a queue sequence number, a name and the id of the parcel that they want to collect. There will be a collection fee for the parcel, which is determined when the parcel is collected.

Seq no	Name	Parcel ID
1	John Brown	C101
2	Mary Smith	C200
3	Sue Jones	X59
4	Tim Baker	C44
5	Brian Hunter	C105

PARCELS			
Parcel ID	Days in Depot	Weight	Dimensions
C45	1	24	4 x 1 x 2
C101	2	10	2 x 5 x 2
C105	10	32	3 x 3 x 3
C200	5	8	6 x 2 x 2
X59	1	5	4 x 1 x 2

You should decide what format IDs have and how the collection fee for the parcel is calculated. You can choose whether to use integers or real numbers for the size and weight.

- The ID format should be something that you can write code to check (e.g. “3 digits in the range 400 – 999”, or “a letter which is C or X followed by 2 or 3 digits” etc etc).

- The collection fee for the parcel should be based on some of: dimension; weight; number of days that the parcel has been in the depot for; discounts for parcels with a particular type of ID. Part of the assignment is to test your calculation thoroughly, so it should not be too simple or too complicated!

Input for the program could be a text file containing details of the parcels in the depot, and another text file containing details of the customers. These text files do not need to be altered by the program.

Output for the first stage is a text file consisting of: a list of collected parcels, including the fee; a list of uncollected parcels; some counts and totals of your choice.

You should NOT use a Graphical User Interface(GUI) in Stage 1. It is better to get the whole program working without a GUI first.

To summarise, your program should:

- Initialise the list of parcels and the queue of customers by reading from text/csv files

- Process each customer one by one, update the list of parcels in the warehouse in memory, and write a report to a text file.

And you must follow the instructions in the 'Software Engineering Requirements Stage 1' below.

Software Engineering Requirements Stage 1

Your initial program should contain at least the following classes: Parcel class; a class for a collection of parcels; Customer class: a class for a collection of customers; some sort of manager class. To make your initial program easily adaptable for stage 2, the code to process a customer (getting their parcel id, finding the parcel, calculating the cost) should be in the manager class or a class to represent a depot worker – not inside any customer or parcel classes.

Remember that you should not produce any graphical user interface now. Your software engineering tasks in this stage are to:

1. Choose the most appropriate data structures (chosen from lists, maps, sets). When making your decision, imagine that you have a large number of parcels. You are expected to alter some or all of the data structures in the provided code.

2. Decide on all the classes, their instance variables and methods. Make a plan to divide the work into iterations, and divide this work between you. Decide when and how often you need to meet or be in contact. Members must do some work independently and then merge their work. How will you integrate your work?

3. ***One person in the group should submit a document giving details of your workplan to GitHub BEFORE you start. Git logs will be used so the lecturer can check whether you have submitted it in advance or not. **

4. Develop your program using iterative development and GitHub for version control.

5. Use validation and exception handling in the constructor of at least one of the 'base' classes, to ensure that the objects of that class that you create are valid. For example, format of ID, and/or length or values of other parameters. This means that you don't check the values for correctness when you read the text file – the checking is done in the constructor.

6. Provide suitable data to check that your program is working correctly. Additionally, use JUnit to test some of your constructors and/or methods, particularly ones involving calculations and the formats of IDs. The calculation of the fee must be in a special 'calculateFee' method so that you can test it easily. You should use test-driven development for these methods.

In this stage you should do ALL the design before writing the code. You should decide on all your classes and their instance variables and their methods, then divide up the coding work in a suitable way, using incremental iterative development.

The group report stage 1

The report will consist of 5 sections:

1. Names of group members and a summary (no more than one page) explaining who did which parts of the application and which parts of the report.
2. Does your program meet the specification, or are there some bits missing or bugs outstanding? Either provide a single sentence “This program meets the specification” or provide details of problems that you know about.
3. A summary of your decisions on formats of IDs and fees.
4. Suitable UML class diagram(s) showing the associations between the classes, and the contents of each class. No need to show setters and getters.
5. A technical report explaining some aspects of this application (see below).
6. A development report describing issues arising during development (see below)

Your technical report should be written for someone who has studied the material on this course but is not familiar with the coursework. You should include diagrams and snippets of code where relevant. It should consist of 2 sections:

- Explain which data structures you used, which classes they are used in, and why you chose them.
- Explain how you have used exceptions in a constructor

Your development report should focus on the development of the project and contain details of the following:

- Your development schedule (as submitted to NILE at the start), showing the breakdown into sections, allocation per person and group meetings. How successful was this approach, and what problems did you encounter? How much did your schedule vary from the original development plan? Did using your iterations work? It’s ok to admit that your original plan was imperfect!
- Testing report. Make a list of tests that you did (e.g., cost of fee correct, parcel not in depot, etc.) and show the corresponding input and output texts to prove that this worked. (annotate these, don’t just produce a list).
- What did you test using JUnit? (just name the classes, don’t paste the code).

Submission Stage 1

Your report should be submitted by the deadline shown on the front page of this document.

Additionally, create

a) A **zip** file of your source code and input files. This source code will be inspected during marking. Please supply only .java files.
b) Provide details about how your work can be cloned from your GitHub account

The second stage of the application

In this part of the coursework, you will extend your application using threads and patterns, functional programming features, and include a graphical user interface.

Functional requirements stage two

An output text file should be produced, as in stage one.

A worker is processing customers in the queue, and at the same time, customers may be joining the queue. You need one thread for the worker and one thread for the queue. You will need to include a delay in each thread, so that processing happens slowly enough for you to see what is happening in your displays. However, you are recommended to get the processing working first, just showing console output, then add the displays as specified below.
A GUI shows the list of parcels still to be processed, the current queue of customers, and the details of the parcel currently being processed by the worker. Each panel should be updated whenever the data which is displayed in that panel is altered. One very basic example is shown below.

CUSTOMERS		
Seq no	Name	Parcel ID
3	Sue Jones	X59
4	Tim Baker	C44
5	Brian Hood	C105

PARCELS			
Parcel ID	Days in Depot	Weigh t	Dimensio ns
C45	1	24	4 x 1 x 2
C105	10	32	3 x 3 x 3
X59	1	5	4 x 1 x 2

Processing: Parcel 200	Collected by Mary Smith	Fee : £10:60

There should be a GUI component which allows the worker to stop work, simulating the end of the day. At this point, the worker continues until they have finished processing the parcel that they were working on. Then details of parcels which have not been processed are written to a text file, other text files are written, and the application closes.

Extend the system in two ways of your choice. For example:

- More than one worker
- A delivery of more parcels arrives in the middle of the day
- Include GUI controls to increase or decrease the number of workers during the session
- Show worker(s) taking a break
- Adjust the speed of processing with runtime controls

And you must follow the instructions in the ‘Software Engineering Requirements Stage 2’ below.

Software Engineering requirements stage two

Continue to use version control.
Use threads as described in the functional requirements section above.

Your design should include patterns, in particular:

1. Use the Singleton pattern in a Log class which is used to record every event (i.e. customer joins queue, customer is removed from queue, processing details are displayed etc.) The Log file is finally written to a text file and should be easy to read.
2. Use the Observer or MVC pattern in your GUI components.

As you develop your system for Stage 2, you should experiment with agile development. Don't make an initial overall plan. Divide the features that you need to implement into several increments, and don't plan beyond the current increment. Do, however, plan each increment! Choose features to incorporate in an increment then plan, design and develop each increment without considering features in the future increments. You may need to refactor your code at the end of each increment, before continuing. Spend some of the time trying out Pair Programming. Consider other agile ideas such as stand-up meetings and time-boxing, and decide whether they are practical for you to use in your project or not.

Group report stage two

The report will consist of a report consisting of several sections:

a) A brief description (< 1 page) of the functionality that your system provides and what it does not do. i.e. does it meet the specification, are there some bits missing or bugs outstanding. How did you extend the application?
b) UML class diagram(s) showing the associations between the classes, and the contents of each class – possibly not both in the one diagram
c) Explain how threads are used in your application.
d) Explain how patterns are used in your application.
e) Explain your extensions to the application.
f) Include Git commit history.
g) A development report. Include details about how you developed your program using agile processes. What features did you include in each iteration? Was this approach successful?
h) Sample screen shots of your application working

Your technical sections of the report should be written for someone who has studied the material on this module but is not familiar with the coursework. You should include diagrams and snippets of code where relevant.

The Individual Report Stage 1 & 2

Your individual report should focus on the development of the project and contain details of the following:

- A comparison of the methodologies used in stage 1 and stage 2. How successful were these 2 approaches, and what problems did you encounter? Which did you prefer?
- What resources did you refer to when developing the project (lecture notes, books, websites, etc).
- A one page summary of the contribution which each person made to the group and which sections of the system they worked on. You should include a weighting for each person in the group. E.g. Ann 25% Mary 35% Tim 40%

Presentation Video

Produce a 10 -15 minute video demonstration of your application. You need to provide a concise explanation of your code and include a walkthrough of all the features of your system.

Submission stage two

You are required to submit:

Your group report as described above.

Each person should submit an individual report as described above.

Additionally

a) The Group Leader for the group should submit your source code and input files to GitHub and NILE. Please either submit a zipped eclipse project or a zip file containing just the source and input files in a format which can be imported into eclipse easily (not rar or nested zips). This source code will be inspected during marking.
b) Each member should show regular Git commits as evidence of their contribution.
c) Each member should contribute to the video group presentation

Marking Scheme

Please refer rubric on NILE and discuss it with your tutor.