INTRODUCTION

Software development is filled with commonly occurring problems, which can be solved in many ways and some if these solutions are better than others. Implementing these common solutions requires a deep understanding of data modelling through abstract data types, classical inheritance, polymorphism, and design patterns. These concepts are the core of the Object Oriented Programming paradigm.

PROJECT OUTLINE

Create a desktop application with Java and JavaFX to aid the management of a banking system using Object Oriented Programming and Design Patterns.

Background and Context:

The application you will be creating will allow university employees manage the university's administration system. They will be able to view details of people and subjects, as well as register lectures and students for subjects.

What you will do:

You are in charge of creating the desktop application using Java and JavaFX. The university system must be carefully modelled using Object Oriented Programming techniques including Inheritance, Polymorphism, and Design Patterns.

Consider the following business rules when designing your objects:

- A university consists of a name, a collection of people and a collection of subjects.
- Each subject has a name, a code, a lecturer, credits, hours per week, a price per month, and a collection of students.
- Each person can be assigned subjects either to attend or to teach.
- The university has a pool of funds. Fees are paid into this pool and salaries are paid from it.

- Administrative staff members have fixed salaries, which they are paid every month. Academic staff members have salaries based on the average number of hours they teach each month.
- Students pay fees based on the subjects for which they are registered.
- At the end of each month, staff members are paid their salaries and students pay their fees.
- A student cannot register for a new subject if it would cause their maximum credits to be exceeded.
- There are four kinds of people, of which two are students and two are staff:
    - Diploma Students, who have 60 maximum credits
    - Degree Students, who have 180 maximum credits
    - Academic Staff (lecturers), who earn R400 per hour that they teach. Salaries are paid out at the end of each **month**. Assume 4 weeks in a month
    - Administrative Staff, who each earn R15000 per month

The JavaFX interface should allow for the functionality:

- Searching for a person by name or ID number
- Viewing that person's details and subjects.
- Viewing subject details.
- Registering people for subjects.
- Adding new people.
- Creating new subjects.
- Closing off the month. (Staff members are paid their salaries and students pay their fees)

A JUnit test suite should be created to test the implementation of business rules.
Changes should be persisted to a file on disk.

What you will be given:

No additional resources. You must create your own additional assets, such as logos and icons.

PROCESS

This section of the brief contains helpful guidance and information concerning your process. You will be technically guided through each phase of the project.

Phase 1: Planning

Screen planning

Design and plan the layout of your application's screens and navigation. Consider the layout and navigation system so that users can easily use the application and its functionality. Record all assumptions, decisions, and reasoning in your rationale.

Phase 2: Implementation Phase

Object Implementation

Create some classes, which represent the entities that make up a University (people, subjects, etc.) The people should leverage inheritance and polymorphism for their functionality. Create three abstract classes called Person, Staff, and Student as well as four concrete classes (one for each kind of person.) Include a class diagram detailing this inheritance hierarchy in your rationale.

Interface Implementation

Create a desktop interface following your designs from phase 1 using JavaFX. Ensure that it allows for all the functionality specified in the project outline.

Phase 3: Testing phase

Unit Testing

Write a test suite in JUnit which tests the business logic specified in the project outline and ensure all tests are green.

Manual Testing

Test your application on a desktop machine.

Validation and Feedback

Make sure that the screens communicate information to the user effectively. Any input errors must be considered and communicated, and users should be guided through the functionality of the application. Use Java's Exception Handling to handle invalid registrations (too many credits).

Quality Control

Make sure the screens of your application are consistent, follow your designs, display all the information that they should, and support all the functionality that they should.

Phase 4: Final Phase

Documentation

Go back to your rationale document and polish it. This document should include all your assumptions, decisions, and reasoning. Include a class diagram of your project in the appendix.

Presentation

You must present your deliverables to your lecturer and the class.

DELIVERABLES

Project files

The project folder including all of your Java files and resources as well as your `pom.xml` manifest file and a self-executing JAR.

Rationale Document

This document should be delivered as a PDF. It must include all your assumptions, design decisions, planning and reasoning. The appendix must include a class diagram.
In addition to the soft copy you submit to the Google Classroom, you must also submit a printed hard copy before presenting your project.

Plagiarism Document

You must hand in a signed plagiarism document in both hard copy and soft copy.

SUBMISSION

- Late submissions will be accepted for a maximum of 48 hours after the allocated time of the deadline, as outlined in the brief. Any late work submitted during this 48 hour grace period will be marked by the lecturer as usual, however the work submitted will be awarded a maximum of 50% as a late-submission penalty
- After 48 hours, 0% will be awarded. OW will take a zero-tolerance stance on work submitted later than 48 hours after the deadline
- Extensions may be applied for BEFORE the deadline only, and should be done via a formal application by the student, which will be recorded on an institutional list where all lecturers can see which students are applying for extensions, and why
- Submission and presentation will take place in Week 8
- Projects should be digitally submitted on Google Classroom
- All deliverables must be placed in a single zipped archive
- The zipped folder must follow the naming convention **`IDV303_xxxxxx_yyyyyy.zip`** where **`xxxxxx`** is your first name and **`yyyyyy`** is your student number

OUTCOME

By completing this project, learners will have learned to:

- Create a desktop application in Java
- Utilise Object Orientation through inheritance, polymorphism, and design patterns
- Write tests in JUnit
- Create a desktop interface with JavaFX and FXML
- Throw, catch, and handle exceptions
- Serialize data to a file
- Manage dependencies with maven
- Document an application implementation
- Professionally present projects
- Manage their time

ASSESSMENT

Learners will be assessed on their:

- Data analysis abilities
- Planning
- Problem solving skills
- Decision making abilities
- Tool use
- Technical skill
- Technical understanding
- Correctness in implementation
- Usage considerations
- Device and hardware considerations
- Initiative and brief expansion
- Layout, communication and aesthetics
- Documentation
- Professionalism

**Please see the attached mark sheet for a detailed breakdown of the assessment weightings.*

SCHEDULE

WEEK	DATE	CLASS WORK	MILESTONES
1	2018-07-16 to 2018-07-20	• Project Brief • IDE Setup	• Planning Phase
2	2018-07-23 to 2018-07-27	• Data Types • Classes and Objects • Singleton Pattern	• Planning Phase • **Assignment 1 Due**
3	2018-07-30 to 2018-08-03	• Object Orientation • Inheritance and Polymorphism • Factory Pattern	• Implementation Phase • **Assignment 2 Due**
4	2018-08-06 to 2018-08-10	• JavaFX • FXML	• Implementation Phase • **Assignment 3 Due**
5	2018-08-13 to 2018-08-17	• Exceptions • Serialization	• Implementation Phase • **Progress Milestone 1**
6	2018-08-20 to 2018-08-24	• Strategy Pattern • JUnit	• Implementation Phase • **Progress Milestone 2**
7	2018-08-27 to 2018-08-31	• Documentation	• Testing Phase • **Progress Milestone 3**
8	2018-09-03 to 2018-09-07	• Submission, Presentation, and Assessment	• Final Phase