ICT711 Programming and Algorithms T225 - August Intensive

Assessment 2

Java Application Group Project

Value: 30% (Group work 20%; Individual contribution 10%)

Due Date: Sunday Week 5

Overview

This assessment will allow students to demonstrate their understanding of the concepts covered in Weeks 1 to 4. This assessment contributes to learning outcomes a, c and d.

In this group project, students in a group of up to 5, will collaboratively design, implement, and test a text-based application using Java. The project requires applying object-oriented programming principles such as inheritance, polymorphism, abstraction, and encapsulation to develop a structured and maintainable application.

On successful completion of this project, you should have demonstrated that you are able to:

- Write a program consisting of multiple interacting objects with inheritance and use of polymorphism and abstraction
- Write a program that provides the requested functionality for the system
- Design class hierarchies to promote code reusability and extend base classes for specific functionality
- Utilise appropriate collection classes from Java's Collections API in appropriate ways/places
- Utilise advanced data structures like ArrayList, Linked Lists, Queues, etc. for efficient data manipulation and management
- Read data from and write data to text files
- Handle exceptions by constructing try … catch blocks for appropriate circumstances
- Design a text-based user interface

In addition to the group-based project, each member will be required to participate in a peer review process. After submission of the final group project report by the group leader on Moodle, each student will evaluate the contributions of their peers. Individual grades may be adjusted based on the results of the peer review.

Task: Employee Management System

Create a medium-sized Employee Management System (EMS) for a company that allows the company admin to manage employees, their personal details, their monthly performance, and their monthly salary details.

To create this EMS for the company, the assessment requirements can be broken down into the following key points:

Object-Oriented Programming (OOP) Concepts

UML diagrams: You should demonstrate your ability to create several objects that interact with each other. For instance, you may have classes for Employee, Manager, and Performance. Create UML diagrams illustrating the class structure and relationships.

Inheritance: You must design your system using inheritance, where a general class (e.g., Employee) can be extended by more specialized classes (e.g., Manager, Intern, Regular). You must highlight these concepts and demonstrate clearly in report.

Polymorphism: The system should allow for polymorphic behavior, where you can treat different types of employees (e.g., Managers, Regular) uniformly, but they will exhibit different behaviors (e.g., calculating salary). You must highlight these concepts and demonstrate clearly in report.

Abstraction: Some of the details of implementation should be hidden, with abstract classes used where appropriate. For example, an abstract Employee class could define common properties (e.g., name, id) and methods (e.g., calculateSalary()), while subclasses define specific behaviors. You must highlight these concepts and demonstrate clearly in report.

You are free to choose the number of attributes and necessary methods to implement the scenario, however you need to ensure all attributes and methods are essential and not irrelevant. For example, the Employee class can have attributes such as department, salary, performanceRating, etc. along with necessary personal details. Please go through all next steps first before finalizing the attributes and methods.

System Functionality

You need to implement all the requested features for the Employee Management System. These features should include functionalities like

- Adding employees to the system
- Viewing employee information via a given query such as query by ID or query by name or query by performanceRating.
- Updating or Deleting an employee record
- Managing monthly performance and salary details, i.e. to view details, issue warning letter, issue appreciation letter, award bonus, apply fine etc.

Utilizing Java Collections

You should make use of appropriate collections from Java's Collections API to manage employee data. Collections like ArrayList or LinkedList has to be used to store lists of employees. You need to justify your choice and then implement and demonstrate the benefit. Consider and explain if there would be a benefit of using more advanced data structures like Queue for managing employee requests or workflows (e.g., processing employee data in FIFO order) or LinkedList for more dynamic and efficient insertion/removal operations.

File Handling

The system should be able to read employee data from and write data to text (csv) files. The system should use a given file employee_data. This file can be a .txt file or .csv file as per your design. This file should have data about at least 10 employees before you load this file. You can create this file directly or write objects into it use Java file writing operation. However, these 10 records must include your group names as employee names. Then, once the file is loaded, the text based menu should ask the admin which operation to perform such as add record, delete record, query record or update record (such as mentioning in the record warning / appreciation / fine / bonus etc.).

You must highlight these concepts and demonstrate clearly in report.

Exception Handling

Try-Catch Blocks: Your program should handle exceptions gracefully. For example, catching exceptions when trying to read from a file that does not exist, or when the user inputs invalid data. Ensure that common exceptions related to File IO are properly handled using try-catch blocks.

User Interface

You should create a text-based user interface (UI) to interact with the system. This could be a command-line interface (CLI) where admin can:

Add, update, and delete employees or view employee information after a query or exit after performing the tasks. The UI should be clear, interactive, and intuitive, guiding the user through each functionality with prompts and feedback.

A sample menu is shown below:

```
===== Employee Management System =====

      1.  Load records in a new file to access latest records
      2.  Add new employees and save to a new file
      3.  Update Employee Information and save to a new file
      4.  Delete Employee and save to a new file
      5.  Load new file to View / query employee details
      6.  Exit

Please choose an option:
```

Implementation and Testing

Implement the EMS in Java, ensuring to comment your code and maintain readability. Include multiple test cases for edge scenarios (e.g., searching for an employee that doesn’t exist, updating record for an employee not existing etc.).

Submission Guidelines

Please remember - there two parts of submission:

Under **Group Assignment (30%) on Moodle:**

In the tab Part 1: Group Report with code

- **ONLY the group leader needs to submit a single PDF report. Other members of the group must not submit in this tab.** This project report must demonstrate the system design, implementation, and testing and include code snippets and output snippets. Please use a cover page showing all group details, and a table of content. Please use figure captions and numbers. Please use references. The report must also include a weekly activity log documenting contributions from each group member. **Please add all source code as appendix to the end of this PDF file or as a drive link having source code.**

In the tab Part 2: Individual tasks (Word file)

- Each group member needs to submit a **single PDF file containing a link to a cloud storage where the source code (working project with all necessary files) is located.** This PDF file must also include an individual report containing

o A peer review table, listing contributions of your other group members. In short, each individual will evaluate the rest of the group members.

o Attendance at weekly meetings – Elaborate how many meetings you have attended and what tasks were assigned to you

o Weekly activity log – List what activities were assigned to you and how you completed them

o Time management – Explain in a short paragraph, how you managed time to complete your tasks

o Tasks – Explain how you completed your tasks and your progress.

o Actual contribution to group project – Explain how much tasks were assigned to you and how many you completed. Mention if there was any change of plan.

In summary, you are only supposed to submit two PDF files. One as a group by the group leader and the other individually!