ECE318: Programming Principles for Engineers

Coursework: Individual Project

Job Finder Application Description

Important deadlines:

1. Submission deadline: 22/11/2024
    - Time: 23:59
    - Submission via Teams
    - What to submit: Code and UML diagrams (Class & Use case diagrams) in a Zip file.
2. Demo dates
    - Tutorial: 27/11/2024
    - Lecture: 28/11/2024

Project Rules

- You have to download the submitted code and diagrams through Teams and will be examine on them.
- Individual submission – the work should be developed solely by you and with no one else!
    1. This is an individual piece of work, hence you will have to design and implement it on your own. No group projects will be accepted.
- Late submission (Penalties):
    - Same day late submission (i.e., after 17:00 and before 23:59 on Nov 20th):
        - reduction of 10% from overall mark
    - Next day late submission:
        - Reduction of 20% from overall mark
    - 2+ day submission until 5 days:
        - Reduction of 50% from overall mark
    - 5+ day submission:
        - Rejected and total 100% reduction from overall mark
- No show policy:
    - Students that do not appear for the demo with no justification of their absence will be automatically marked with 0.
        - Need to inform the ECE admin office cc'ing Prof. Marnerides in your email and explaining why you will be absent that day.
- Plagiarism:
    - Students submitting copied diagrams or code that they cannot explain will be investigated for plagiarism and will be automatically assigned with a 0 mark. If the case is severe a further case in the departmental council will be raised.
- Notes:
    1. students requesting an extension with the appropriate justification need to ask at least 1 week in advance of the submission deadline.
    2. students with last minute requests for extension without justification will be automatically rejected
        1. unless an unexpected critical circumstance has emerged (e.g., unforeseen family tragedy, illness etc.) that will be evaluated based on the justification provided by the student.

Goal

Design and implement an application for a leading job finder agency, allowing administrators to access and manage the agency's job dataset through an intuitive GUI. The application should enable the admin to view, add, update, and delete job listings, as well as filter, sort, and generate reports. This project provides hands-on experience in database design and management, including setting up tables and relationships, and implementing CRUD (Create, Read, Update, Delete) operations for direct database interaction. Emphasizing core OOP principles, you'll design flexible classes to represent elements within the dataset, incorporating features like UML diagrams, polymorphism, inheritance, and encapsulation to build a well-structured, modular codebase. Building a GUI is essential for user interaction, enabling data filtering, sorting, and visualization through charts to display trends and insights. Although using a database is recommended, you may opt for built-in collections, though this may affect performance with larger datasets.

Datasets

Job Dataset - A Comprehensive Job Dataset for Data Science, Research, and Analysis
* From Kaggle * you can download and see the description for each column in the link.

Requirements - what you will be examined on

The exam will be split in two parts, Explanation and Demonstration on two specific scenarios

Explanation - OOP Principles & Java (3 minutes)

- Class diagram
  - Create Class Diagram
  - Relationships between Classes: Describe the associations, dependencies, or generalization relationships between classes.
  - Methods & Attributes: Briefly explain the key methods and attributes in each class. Focus on relevant attributes and methods.
- Use Case diagram
  - Create Use Case Diagram
  - Explanation of Use Case Diagram: Discuss the main actor (Admin) and use cases (e.g., Create Job, Search Job, Export Report) shown in the diagram. Explain how the actor completes specific tasks through system interactions.
- Answer to 4 questions related to your code.

Demonstration - Implementing details for GUI & Database (4 minutes)

- You are the administrator of this application and you must to:
- Through the GUI demonstrate database functionality:
  - **Filter** the database to retrieve only the rows with these three characteristics:
  1. job title e.g. all Digital Marketing Specialist
  2. working type e.g. Full-time
  3. the Country e.g. Cyprus
  - **Sort** the filtered results based on the low-end years of experience (e.g. 5 to 15 use the 5 years as the low-end)
  - **Export** the sorted results to a .csv file, keeping the same format as the dataset.
  - **Present** in the GUI these calculated details:
    1. the average low-end and high-end salary (e.g. $59K-$99K low end : $59K high-end: $99K)
    2. the average low-end and high-end needed years of experience (e.g. 5 to 15 Years low-end: 5 years , high-end : 15 years)
    3. The number of jobs founded with these details (job title , working type , country)
- Through the GUI demonstrate database functionality:

    - **Create** a new job description (including all 23 characteristics in the dataset)
    - **Update** and **Delete** the new job description you just added
    - **Search** for the profile you just added using the job ID, and present (**Read**) the job profile with all 23 characteristics from the job description.
- For GUI:
    - **Functionality Check: Ensure all:** functions work as expected.
    - **Page Organization & Navigation:** Split functionality across well-organized pages with smooth navigation between them.
    - **Error Handling: Implement error:** messages and guidance for adding/updating elements to assist users.

Marking System

C. = Creation
D. = Explanation
E. = Demonstration

%	Marking System	Requirements
(20)	Class diagram	- **(5)** (C&E) : Class Diagram
		- **(5)** (E) : Relationships
		- **(5)** (E) : Methods & Attributes
	Use Case diagram	- **(5)** (C&E) : Use Case Diagram
(20)	OOP Principles	- **(5)** (E) : Encapsulation:
		- **(5)** (E) : Abstraction:
		- **(5)** (E) : Inheritance:
		- **(5)** (E) : Polymorphism:
(35)	Database or Built-in collections	- **(10)**(D) : CRUD
		- **(5)** (D) : Search
		- **(5)** (D) : Filter
		- **(5)** (D) : Sorting
		- **(5)** (D) : Presentation
		- **(5)** (D) : Export repots
(25)	GUI	- **(15)**(D) : Functionality
		- **(5)** (D) : Page Organization
		- **(5)** (D) : Error handling

Technical tips

- **GUI**:
  - **Swing or JavaFX**: Use Swing or JavaFX to create a responsive and interactive interface, ensuring smooth navigation and functionality for the user.
- **Database**:
  - **JDBC**: Recommend JDBC for connecting to the database, enabling CRUD operations and other data management tasks.
  - **Database Choice (MySQL)**: Use MySQL for a robust, server-based database if project scalability is important.
  - **The database is not mandatory** , you can use just built-in collection but probably will crash or the performance will be a detester
- **Class Diagram and Use Case Diagram**:
  - **draw.io**: Use draw.io to create clean and professional diagrams, including class and use case diagrams. Export diagrams as PDF or PNG for easy integration into documentation or reports.
- **Exception Handling**:
  - Use try-catch blocks to handle exceptions and provide user-friendly messages.
  - **GUI Error Handling**:
    - Use JOptionPane (Swing) or ControlsFX (JavaFX) for error dialogs and form validation.
  - **Database Error Handling:**
    - Use HikariCP for efficient JDBC connection pooling and error management.
    - Integrate Log4j or SLF4J for logging database and application errors.

Tips

- Begin by designing the GUI(not implement yet) and then continue with the class diagram.
- For OOP, implement the basic structure of all classes, define the relationships, then add details.
- Have a backup plan to demonstrate requirements if the GUI doesn’t work.
- As you implement features, consider optimizing the app for performance. The dataset is large, make sure the search and sorting functions are efficient to avoid slowdowns.
- Reminder that the database is not mandatory, you can just use built-in Java collections.