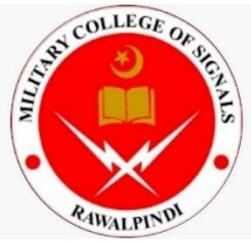


MILITARY COLLEGE OF SIGNALS

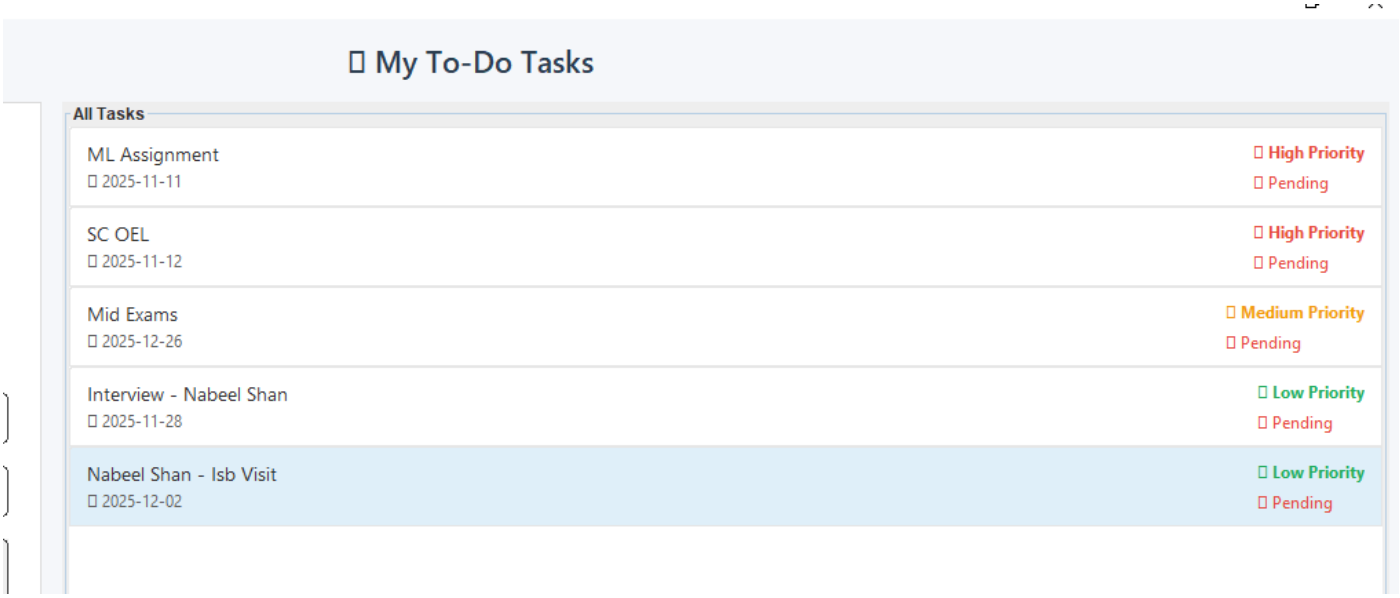
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

Open Ended Lab

Software Construction

Submitted By	**Nabeel Shan**
Date	**November 11, 2025**
Course	BESE 29 (C)
Semester	5th – Fall 2025

Developing a TODO List Application in Java



Abstract

This open-ended lab focuses on developing a Java-based desktop To-Do List Application that demonstrates object-oriented programming (OOP) principles such as abstraction, encapsulation, inheritance, and polymorphism. The project was implemented using Java Swing for the GUI, providing CRUD functionality (Create, Read, Update, Delete) and prioritization of tasks. The application enables users to add, modify, remove, and mark tasks as complete, offering a real-world implementation of OOP in a productivity tool context.

Objective

The primary objective of this lab is to **design, investigate, and implement** a functional To-Do List desktop application using the Java language. Specifically, this exercise aims to demonstrate:

1. **Effective application of object-oriented design principles** and modern software engineering practices, fulfilling the **Investigation (WA4/CLO2)** target by analyzing requirements

and designing an optimal architecture from multiple potential approaches.

2. **Proficient utilization of modern tools** such as the IntelliJ IDEA, Git for version control, and the Java Swing framework, satisfying the **Modern Tool Usage (WA5/CLO4)** target. The goal is to deliver a maintainable, user-friendly desktop application that manages daily tasks with features like CRUD and task prioritization.

Tools and Technologies

Category	Tool/Technology	Rationale/Application
Programming Language	**Java SE**	Core language for OOP implementation.
IDE	**IntelliJ IDEA**	Selected for its advanced debugging, refactoring, and project management capabilities
GUI Framework	**Java Swing**	Utilized for building the cross-platform desktop UI.
Version Control	**Git**	Essential for tracking changes, managing branching, and ensuring codebase integrity.
Operating System	**Windows 10**	Development and deployment environment.

System Design

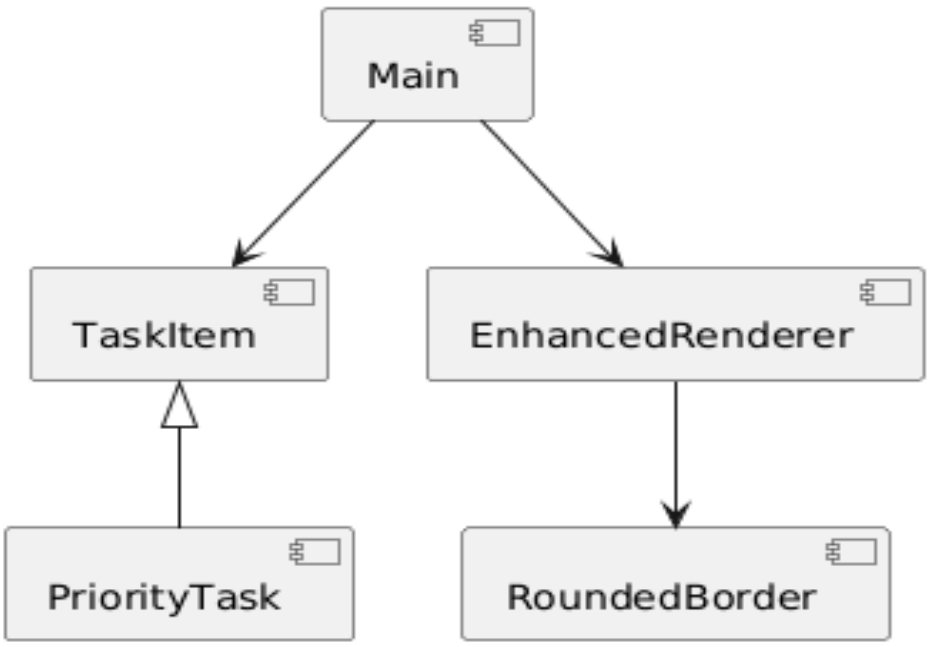
The system employs a **modular, layered OOP architecture** to ensure high cohesion and low coupling, which was determined to be the **best policy** after considering monolithic and Model-View-Controller (MVC) alternatives.

Key System Components:

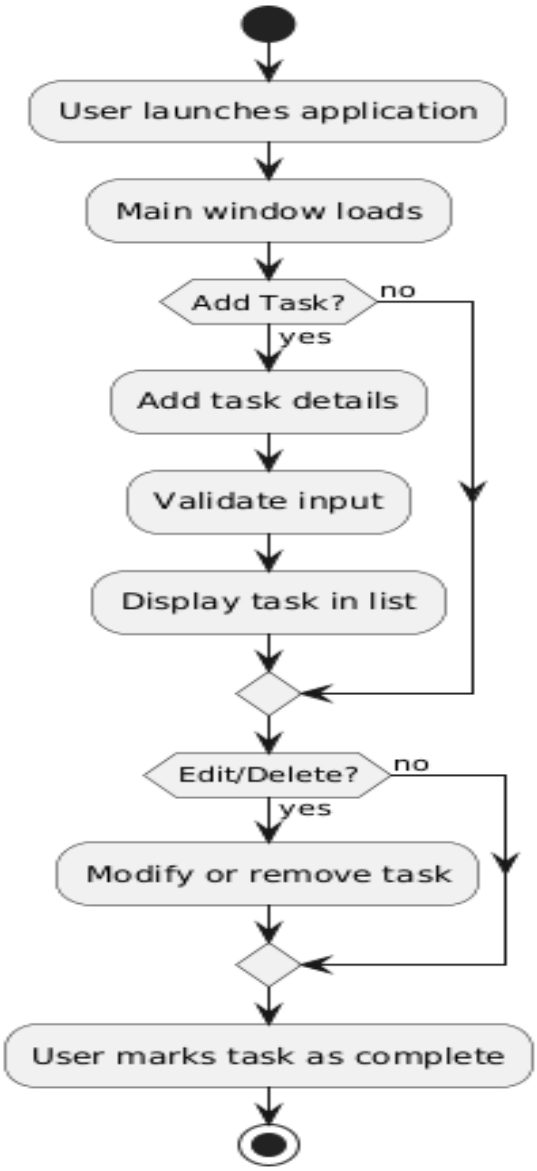
1. **TaskItem (Abstraction/Encapsulation):** The base class representing a generic task, encapsulating fundamental attributes like description, dueDate, and isCompleted status, managed via controlled **getter and setter methods**.

2. **PriorityTask (Inheritance):** Extends TaskItem and introduces the **polymorphic** priority attribute (an enum of Low, Medium, High). This demonstrates **inheritance** for feature extension.
3. **Main (Control Logic):** Manages the primary GUI logic, user interaction, and state changes (CRUD operations) through event handling.
4. **EnhancedRenderer (Polymorphism/Design):** A custom ListCellRenderer implementation that uses **polymorphism** to determine the rendering style (color, font, status icons) based on the task's status and priority, significantly enhancing user readability.
5. **RoundedBorder (Utility):** A dedicated utility class for applying non-standard, modern UI visuals, demonstrating separation of concerns.

Class Diagram



Flow Diagram



Implementation and Modern Tool Usage

The implementation utilized IntelliJ IDEA for streamlined development, leveraging its integrated debugger and dependency management.

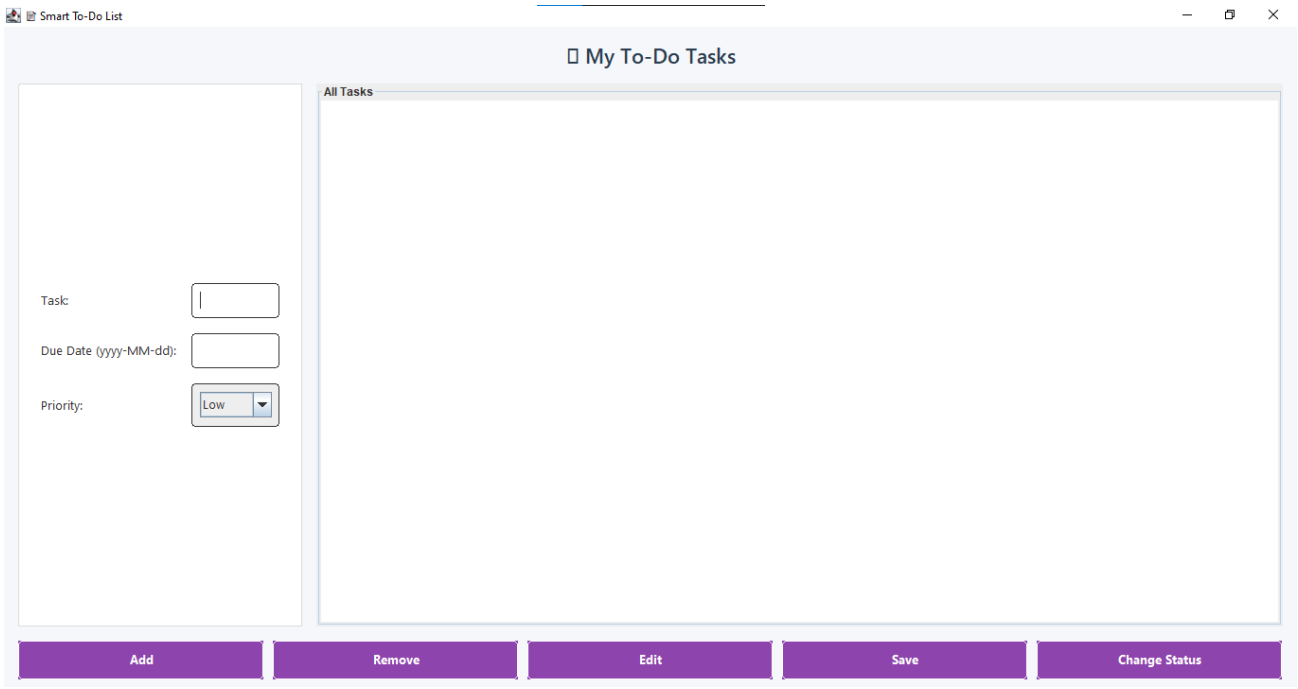
OOP Principles in Code:

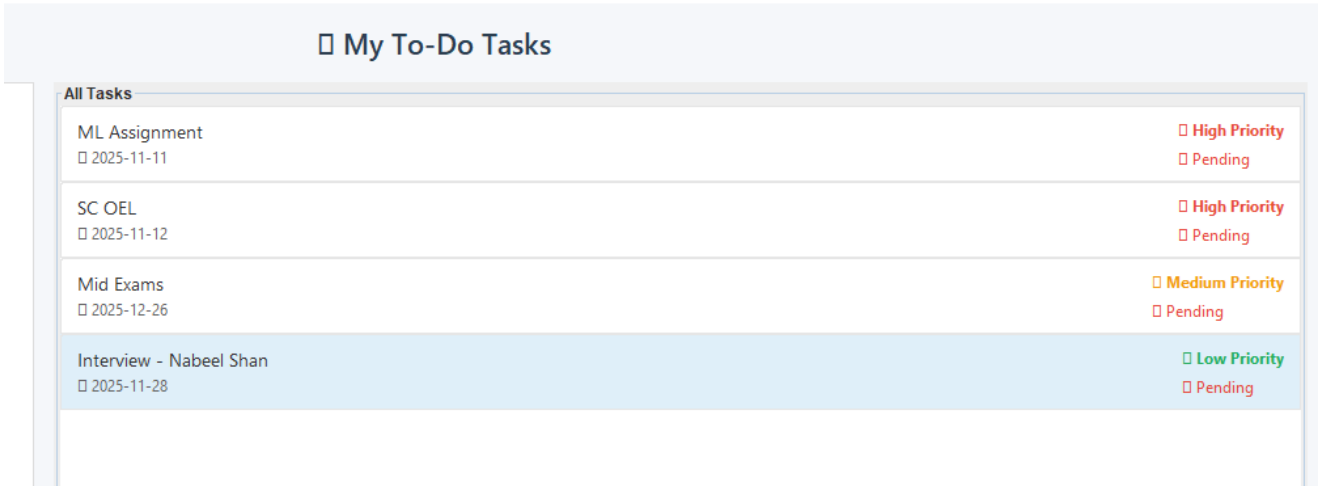
- Encapsulation: Task properties are protected and strictly accessed via public methods (e.g., setCompletionStatus(boolean)).
- Inheritance: PriorityTask inherits all base task logic from TaskItem, reducing code redundancy.
- Polymorphism: Utilized in the EnhancedRenderer where the display logic changes (is polymorphic) depending on the task's type and status without explicit type checking in the main logic loop.

Graphical User Interface (GUI) and Features

The UI was constructed with Java Swing components. This framework was selected over JavaFX for its simpler integration with basic desktop applications.

Feature Category	**Description**
CRUD Operations	Dedicated buttons for **Add**, **Edit**, **Delete**, and **Change Status** enable full task management.
Task Prioritization	A dedicated dropdown (JCombo Box) allows assigning Low, Medium, or High priority, visually reflected in the styled task list.
User Feedback	Utilized 'Toast-style' messages for non-intrusive feedback on successful operations (e.g., "Task Added Successfully").
Modern UI	Implementation of **RoundedBorder** and dynamic component colors to enhance user experience beyond default Swing aesthetics.





Version Control and Collaboration

Git was the mandatory tool for version control. All development steps were compartmentalized:

- Initial setup and core logic were committed to the main branch.
- Feature additions (e.g., Task Prioritization) and UI improvements were developed on separate, descriptive branches (e.g., feature/task-priority, fix/ui-styling).

- The commit history is **traceable and descriptive**, demonstrating professional-level version control practices.

Conclusion

This open-ended lab provided invaluable, hands-on experience in applying **Object-Oriented Programming (OOP) concepts** and utilizing **modern development tools (IntelliJ, Git)**. The development of the To-Do List application reinforced key concepts like abstraction, modularity, and GUI-based user interaction.