East West University

Smart Task Manager

Submitted By:
Student Name: MD.Tawsif Islam
Student ID: 2025-1-60-194
Course Code & Title: CSE110 – Object Oriented Programming

Submitted To:
Dr. MD. Hasanul Ferdaus
Department of Computer Science and Engineering

Introduction

The Smart Task Manager is a console-based Java project designed to help users manage daily tasks efficiently. The program allows users to add tasks, mark tasks as completed, and view tasks based on their completion status. The application follows a simple menu-driven approach so that users can easily interact with the system through the command line.

The main purpose of developing this project is to give an idea of core object-oriented programming (OOP) concepts in Java. This project focuses on writing clean, structured, and reusable code rather than building a complex user interface. By implementing this system, the fundamentals of OOP and exception handling can be clearly understood.

The main features of the Smart Task Manager include adding new tasks, displaying all tasks, filtering completed and pending tasks, and handling invalid user input properly. The project uses several object-oriented concepts such as classes and objects, Encapsulation, Inheritance, Polymorphism, and Exception Handling.

System Design

Overview of the System Structure

The Smart Task Manager follows an modular object-oriented design. The program starts execution from the ManageTask class, which contains the main() method. This class controls the menu system and user interaction. Based on user input, different methods are called to perform specific actions such as adding tasks or displaying tasks.

All tasks are stored in a single ArrayList<Task>, where Task is the base class. Depending on the task type, objects of WorkTask or PersonalTask are created and stored in the list. The program continues running until the user chooses to exit.

Class Descriptions

Main features include:

- Add new tasks (Work or Personal)
- View all, completed, or pending tasks
- Mark a task as completed
- Delete tasks
- Handles input errors using exception handling

OOP concepts used:
Class, Object, Encapsulation (private fields + getters/setters), Inheritance (subclasses extending super class), Polymorphism (method overriding), and Exception Handling (try-catch for invalid input and index errors).

2. System Design

2.1 Overview of System Structure

The system consists of a main `TaskManager` class that controls user input and menu flow. Tasks are stored using `ArrayList<Task>`. The base class `Task` holds common

properties. Two subclasses (`WorkTask` and `PersonalTask`) inherit from `Task` and override the `getDetails()` method to show different behavior based on task type.

Program Flow

1. Program starts and shows menu options
2. User enters a number
3. Based on choice, the matching method runs
4. If input is invalid, error is caught and user can try again
5. Loop continues until exit is chosen

2.2 Class Descriptions

Class Name: Task
Parent Class: None
Purpose: Stores common task data (title, description, status)
Key Fields: `private String title`, `private String description`, `private boolean isCompleted`
Key Methods: `getDetails()`, `setCompleted()`, getters

Class Name: WorkTask
Parent Class: Task
Purpose: Represents work-related tasks
Key Fields: Inherited from `Task`
Key Methods: `getDetails()` (Overridden)

Class Name: PersonalTask
Parent Class: Task
Purpose: Represents personal daily tasks
Key Fields: Inherited from `Task`
Key Methods: `getDetails()` (Overridden)

Class Name: TaskNotFoundException

Parent Class: Exception
Purpose: Handles invalid task index errors
Key Fields: message
Key Methods: Constructor

Class Name: TaskManager
Parent Class: None
Purpose: Controls menu, user input, task storage, and program execution
Key Fields: `private static ArrayList<Task> tasks`
Key Methods: `addTask()`, `markCompleted()`, `showTasks()`, `deleteTask()`, `main()`

2.3 Inheritance and Polymorphism

`WorkTask` and `PersonalTask` inherit from `Task`. They override the `getDetails()` method. The base method prints task info normally, while overridden versions add a header like `[Work Task]` or `[Personal Task]` before showing details. This allows one method call (`getDetails()`) to behave differently depending on object type.

2.4 Encapsulation

All fields inside `Task` are private and cannot be accessed directly from outside the class. The program interacts with task data only through getter and setter methods like `getTitle()` and `setCompleted()`. This protects the data and prevents accidental modification.

2.5 Exception Handling Strategy

Exceptions handled:

- `InputMismatchException` → when user enters non-number in numeric input
- `IndexOutOfBounds / Invalid index check` → prevents selecting task index that does not exist.
- These are necessary to prevent crashes and make the program reliable.

3. Implementation Details

Important Code Snippets

Snippet 1: Part of Constructor

```
public Task(String title, String description) {
        this.title = title;
        this.description = description;
        this.isCompleted = false;
    }
```

This initializes a new task object and sets default status to Pending.

Snippet 2: Overridden Method (Polymorphism)

```
    @Override
    public String getDetails() {
        return "[Work Task]\n" + super.getDetails();
    }
}
```

This changes how WorkTask prints details compared to super class.

Snippet 3: Try–Catch Block

```
catch (InputMismatchException e) {
    System.out.println("Error: Enter a valid number!");
    sc.nextLine();
}
```

This handles invalid numeric input safely.

4.Sample Outputs:

4.1 Adding Task

```
--- Smart Task Manager ---
1. Add Task
2. Mark Task Completed
3. Show All Tasks
4. Show Completed Tasks
5. Show Pending Tasks
6. Delete Task
7. Exit
Choose option: 1
Enter title: CSE
Enter description: 110
Type 1 for Work, 2 for Personal: 1
Task added successfully!
```

4.2 Invalid Input:

```
Choose option: 1
Enter title: cse
Enter description: 110
Type 1 for Work, 2 for Personal: 3
Invalid type! Task not added.
```

4.2 Mark Completed:

```
--- Smart Task Manager ---
1. Add Task
2. Mark Task Completed
3. Show All Tasks
4. Show Completed Tasks
5. Show Pending Tasks
6. Delete Task
7. Exit
Choose option: 5

Index: 0
[Work Task]
Title: CSE
Description: 110
Status: Pending

--- Smart Task Manager ---
1. Add Task
2. Mark Task Completed
3. Show All Tasks
4. Show Completed Tasks
5. Show Pending Tasks
6. Delete Task
7. Exit
Choose option: 2
Enter task index: 1
Error: Invalid index!
```

5. Conclusion

The project successfully implements a working task management system using core OOP principles. Through this project, I learned how inheritance reduces repeated code, encapsulation protects data, and polymorphism allows methods to behave differently based on object type. Exception handling made the application crash-proof and more user-friendly.

Possible improvements:

- **Add priority or deadline**
- **Save tasks to file**
- **Improve search feature**

6. References (Optional)

1.Java OOP Concepts from course materials

2.Course lecture materials