1. **Problem statement**

Write a console application for astronauts that allows them to manage their daily schedule, add tasks, remove tasks, and view the tasks. A task is going to be composed of:

**Description**

Start time, End time

Priority: High, Medium, Low

**The software must do the following:**

NOT show overlapping

Sort tasks by start time  
Suitably give error messages

All of your codes should be efficient as well as best practice.

Use design patterns: Singleton, Factory, and Observer.

1. **Solution Strategy**

**Requirements Understanding:** Offer a straightforward solution for performing CRUD-Create, Read, Update, Delete functionality for managing tasks and not having duplicate times of tasks.

**Design Patterns:**

**Singleton:** There should be only one instance of ScheduleManager.

**Factory:** Use the factory to generate task objects for beginning the tasks uniformly.

**Observer:** An observer will notify the user in case of duplication of the task times.

**Input and Validations Monitor:** the input of task descriptions, start and end times and verify for no overlapping of the schedules.

**Task Management:** It should define the functions adding, deleting, and displaying tasks. Tasks by time has to be sorted which is highly essential for management.

1. **Code Explanation**

**Classes and Responsibilities:**

**Task.java:**

represents a single task. It has attributes such as description, start time, end time, priority. We define the method overlapsWith(Task other) to check for overlapping times with another given task. Also, we override toString() to get an easy print-ready string representation of a task.

**TaskFactory.java:**

a factory class which creates tasks. It simplifies creation of tasks and makes centralized how the task objects are created.

**TaskObserver.java:**

Observer class checking whether there is some conflict of the given new task as argument with the tasks that are already executed. The class will alert a user if a task conflicts with an already scheduled task.

**ScheduleManager.java**: (Entity is a set of tasks)

It is the main application's class and implements the Singleton pattern, therefore, there is only one instance of the manager.

**Methods**

**addTask()**: Adds the task to the list. Meanwhile, it uses observer to check up some conflict with the previously added task.

**removeTask():**Removes a task from the list based on its description.

**viewTasks()**:Displays all tasks sorted by start time.

**taskExists():**Checks whether a task exists based on the description.

**Main.java:**

Is just the entry point of the program. It is nothing but a simple text-based menu for the user:

**Add Task**:

takes a description, start and end times of a task, and priority, and calls ScheduleManager to add the task.

**Remove Task:**

This function removes a task based on the user's input.

View Tasks: All tasks will be sorted by time and can be seen on the screen.

**Exit**: Exit the program.

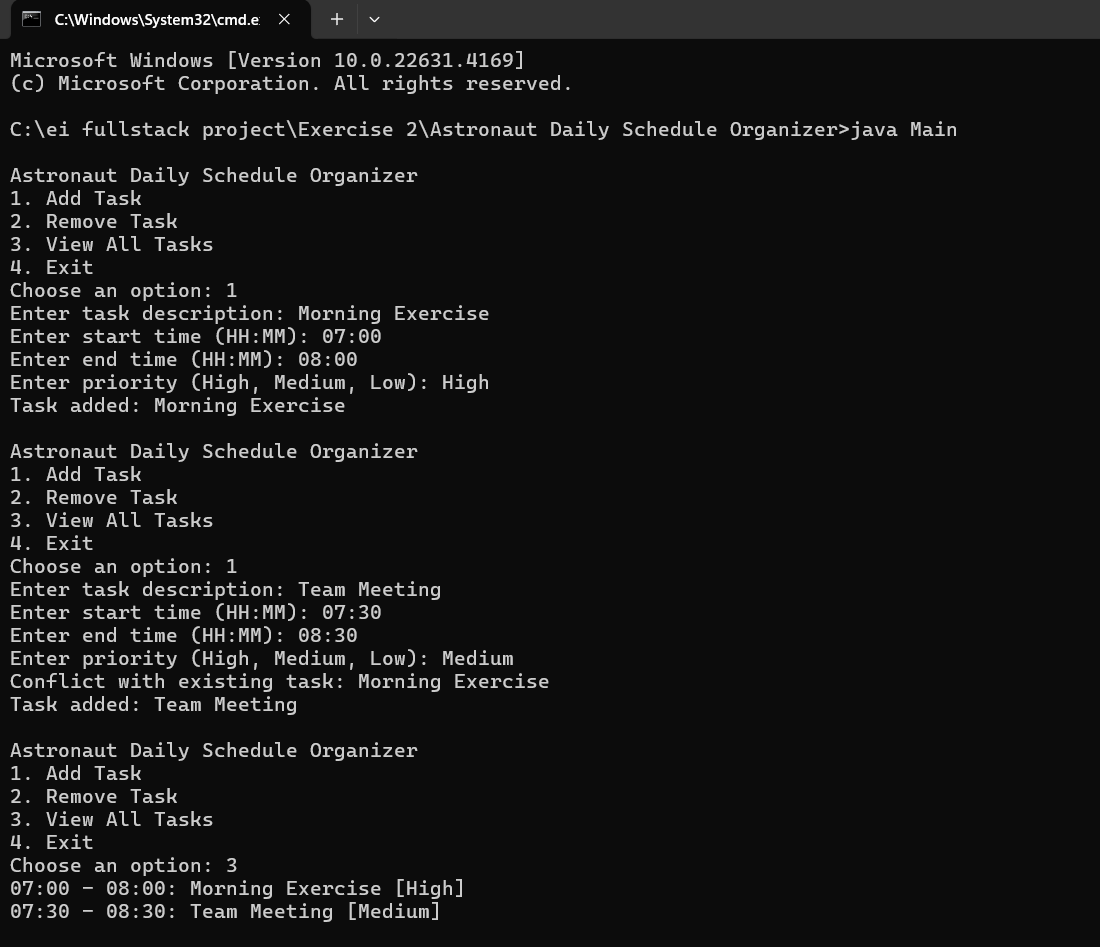
1. **Conclusion**

The Astronaut Daily Schedule Organizer application is a console-based program aimed at helping users be better at scheduling their daily life. Design patterns such as Singleton, Factory, and Observer have been applied in order to ensure that the solution indeed turns out to be modular, scalable, and maintainable. Major functionalities include adding, deleting, and viewing tasks with validation for potential scheduling conflicts.

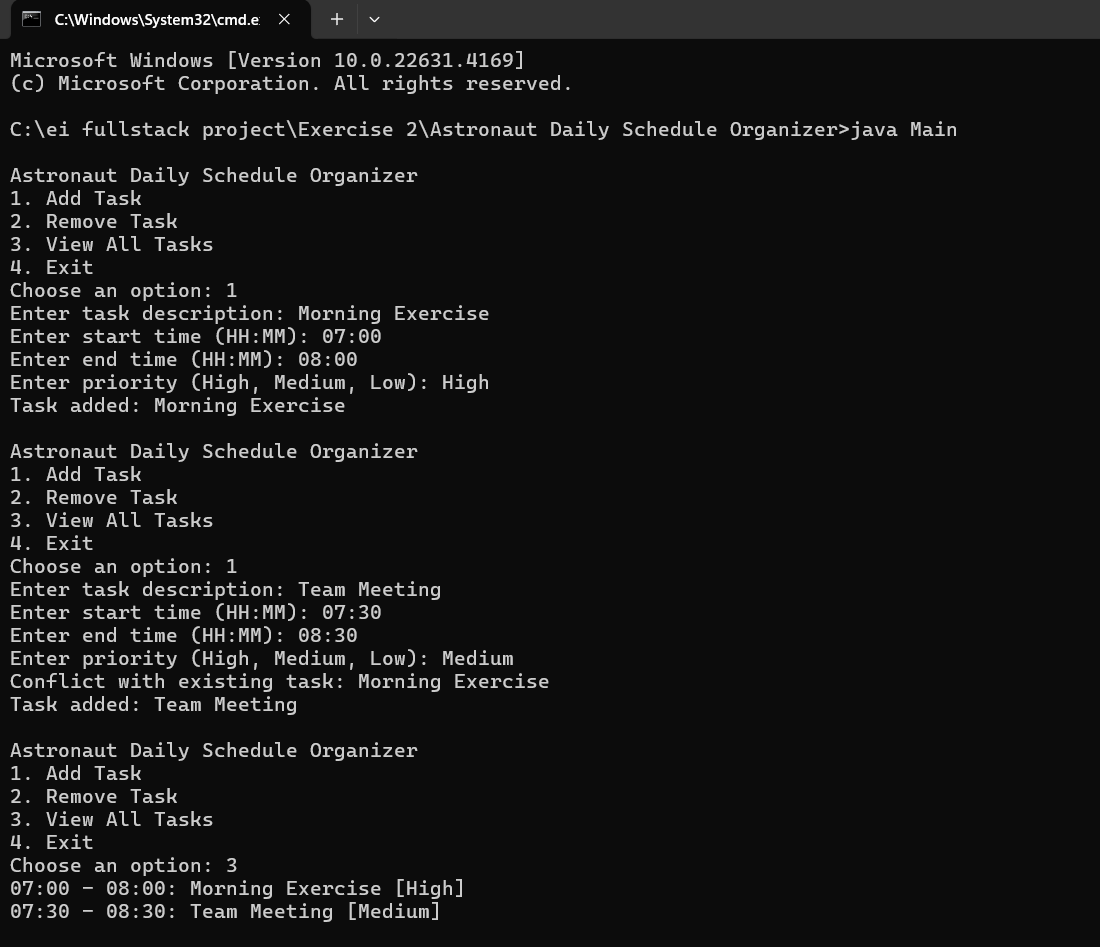
This project demonstrates how to apply object-oriented principles and design patterns to construct a simple, efficient application that is easy to use for an astronaut or the user. Provided you do not overlap tasks and maintain good coding practice, this program is a reliable means of organising daily tasks for astronauts.

1. **Output**

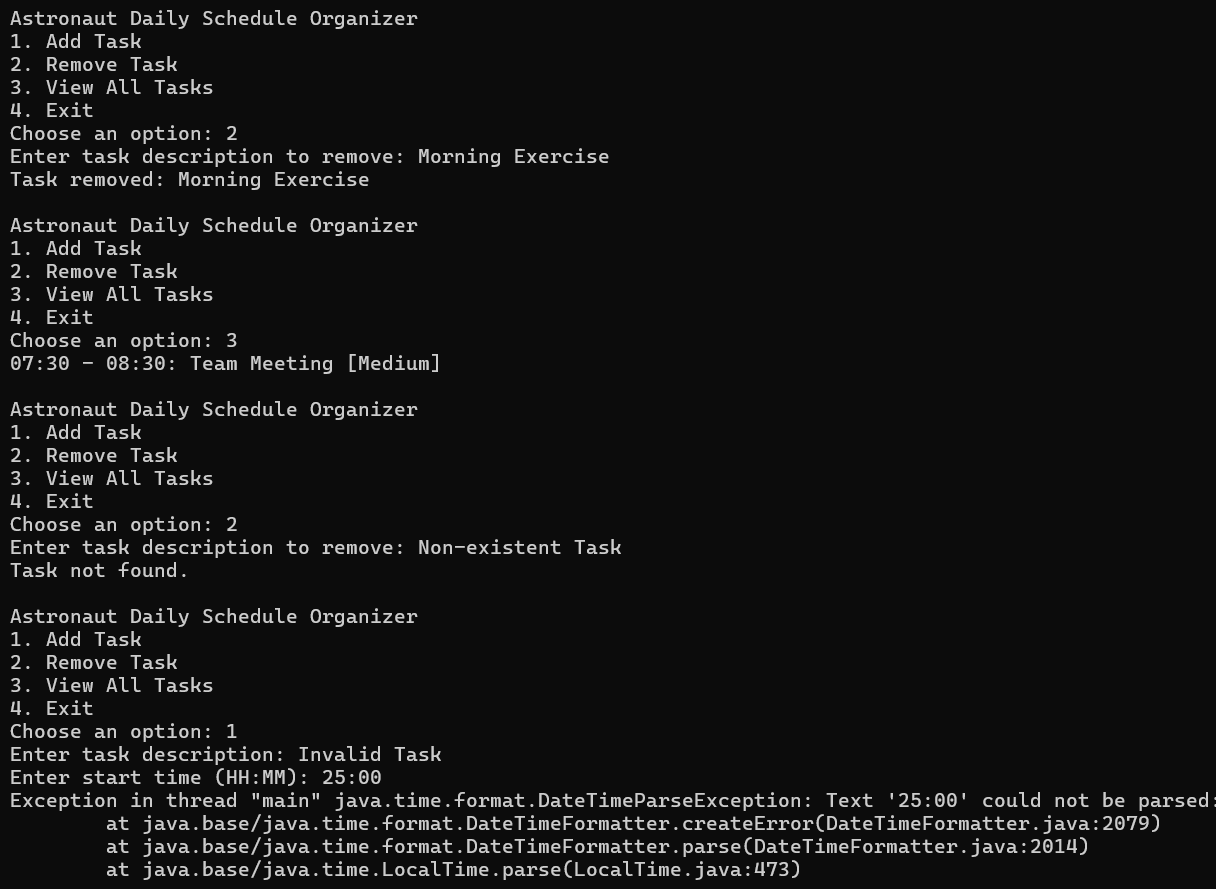
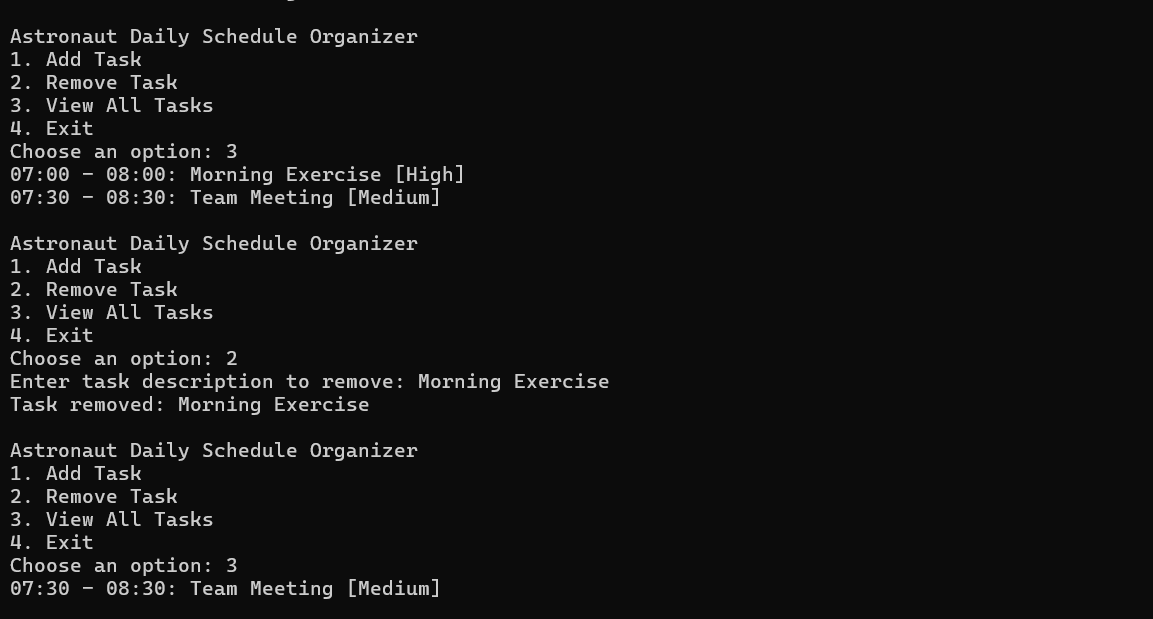
**1. Add Task**

****

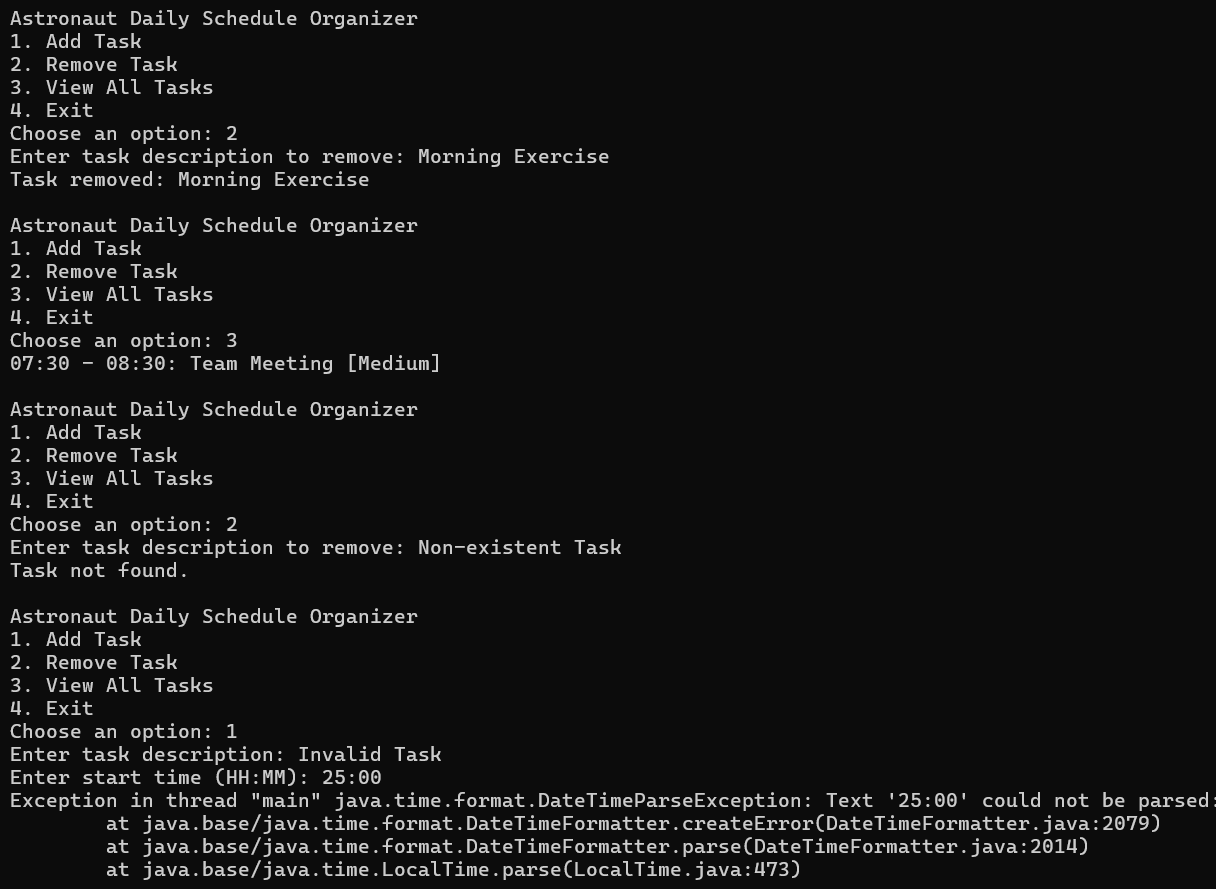
**2.View All Task**

****

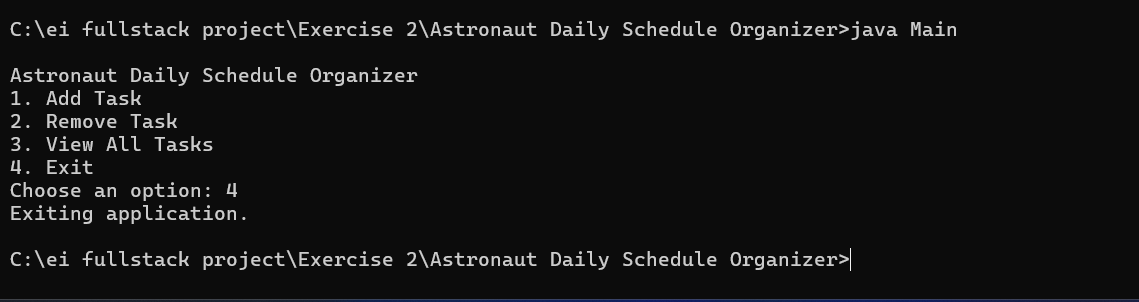
**3.Remove Task**



**4.Exception Handling**



**5.End**

****