Task Management Application

Sabina Tleugabyl

Key Features

- Add new tasks with title, description, and priority.
- Change task status (In Progress, Completed, On Hold).
- View all tasks in one interface.
- Real-time updates of the task status displayed on the interface.

Project Objective:

- The goal of this application is to create a task management system that allows users to add, track, and update tasks with different priorities and statuses in an easy-to-use interface.

Chosen Architecture and Design Patterns with Justifications

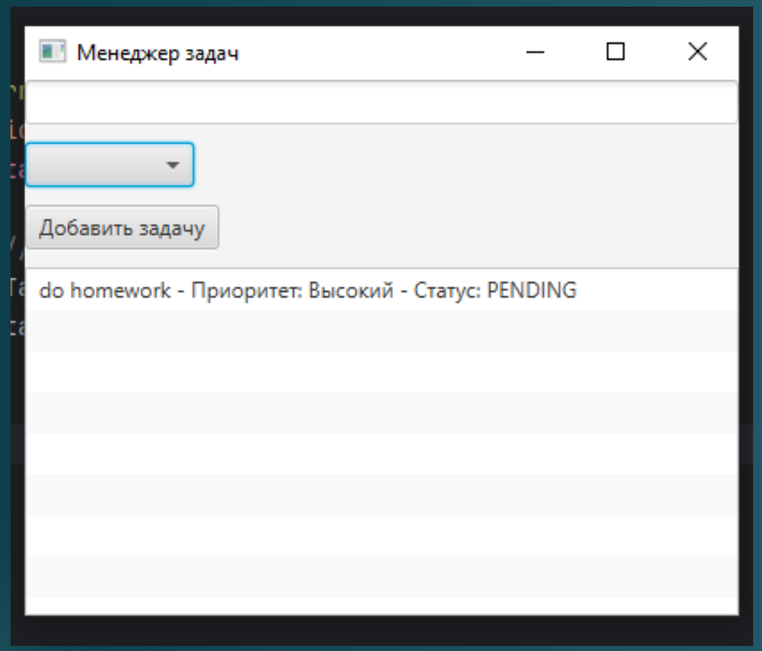
Architecture: Model-View-Controller (MVC)

-
  - Model: Responsible for the data and business logic. Example: Task, TaskManager.
  - View: Responsible for displaying the data and interacting with the user through the GUI.
  - Controller: Handles user inputs, updates the Model, and updates the View accordingly.

Design Patterns)

Singleton Pattern
Factory Pattern
Strategy Pattern
Observer Pattern
Facade Pattern
Adapter Pattern

Key Features of the Application



- Adding Tasks: Users can input the task title, description, and select its priority (e.g., High Priority) through a simple form.
- Priority Assignment: Each task is assigned a priority level during creation. This allows users to categorize tasks based on urgency and importance, which helps in organizing the task list effectively.
- Viewing Tasks: All tasks are displayed in a list format, showing each task's title, description, and priority level. The application also allows for filtering or sorting tasks by priority to help users focus on higher-priority tasks.
- User Interface: The interface updates dynamically whenever a task is added, deleted, or modified, ensuring the task list always reflects the current state of tasks.
-

Challenges Faced and Solutions

- Problem 1: Managing task states and transitions.
  - Solution: Used the Strategy pattern to handle different task statuses and transitions dynamically.
- Problem 2: Updating the UI when task statuses change.
  - Solution: Used the Observer pattern to automatically update the task list in the UI whenever a task's status changes.
- Problem 3: Creating various types of tasks with different priorities.
  - Solution: Used the Factory pattern to abstract the creation of different task types and simplify adding new types of tasks in the future.
- Problem 4: Adapting internal task data to be displayed in UI components.
  - Solution: Used the Adapter pattern to convert task objects into a format that can be displayed in JavaFX components.

Conclusion

In summary, this Task Management Application provides a streamlined approach to organizing tasks by prioritizing and tracking their progress in a user-friendly interface. The use of well-chosen design patterns, including Singleton, Factory, Strategy, Observer, and Adapter, enhances the maintainability, flexibility, and scalability of the system. These patterns not only simplify task management and allow for easy expansion of features in the future, but they also make the codebase easier to understand and extend.