Simple Task List Application Documentation

Table of Contents

Introduction

The Simple Task List Application is a basic Java program that allows users to manage a list of tasks. The primary functionalities include adding new tasks, removing existing tasks, and listing all tasks. The application is designed to be used via a simple text-based user interface, making it accessible and easy to use.

Design Choices

1. Simplicity

The application is designed to be straightforward and easy to use, with a simple command-line interface. This choice ensures that users can quickly interact with the program without needing a complex setup or GUI.

2. Object-Oriented Design

An object-oriented approach was chosen to encapsulate the data and behavior related to tasks. This design choice makes the code more modular, maintainable, and scalable.

3. User-Friendly Interface

The text-based user interface provides clear instructions and feedback to the user, ensuring an intuitive interaction experience. This interface is suitable for a basic task list application without requiring additional libraries or frameworks.

Implementation Details

The application consists of three main components: the main class (`SimpleTaskListApp`), the `Task` class, and the `TaskManager` class.

Main Class

The `SimpleTaskListApp` class contains the main method, which acts as the entry point of the application. It handles user input and calls the appropriate methods in the `TaskManager` class.

```
import java.util.Scanner;


public class SimpleTaskListApp {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        TaskManager taskManager = new TaskManager();


        while (true) {

            System.out.println("Task List Application");

            System.out.println("1. Add Task");

            System.out.println("2. Remove Task");

            System.out.println("3. List Tasks");

            System.out.println("4. Exit");
```

```
System.out.print("Choose an option: ");

int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

switch (choice) {
    case 1:
        System.out.print("Enter task description: ");
        String description = scanner.nextLine();
        taskManager.addTask(new Task(description));
        break;
    case 2:
        System.out.print("Enter task ID to remove: ");
        int id = scanner.nextInt();
        taskManager.removeTask(id);
        break;
    case 3:
        taskManager.listTasks();
        break;
    case 4:
        System.out.println("Exiting application.");
```

```
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
}
}
```

Task Class

The `Task` class represents a single task. It contains an ID and a description.

```
public class Task {
    private static int idCounter = 1;
    private int id;
    private String description;

    public Task(String description) {
        this.id = idCounter++;
        this.description = description;
    }

    public int getId() {
```

```
        return id;
    }

    public String getDescription() {
        return description;
    }
}
```

TaskManager Class

The `TaskManager` class handles the logic for adding, removing, and listing tasks. It uses an `ArrayList` to store the tasks.

```
import java.util.ArrayList;

public class TaskManager {
    private ArrayList<Task> tasks = new ArrayList<>();

    public void addTask(Task task) {
        tasks.add(task);
        System.out.println("Task added: " + task.getDescription());
    }

    public void removeTask(int id) {
        for (int i = 0; i < tasks.size(); i++) {
            if (tasks.get(i).getId() == id) {
```

```
                tasks.remove(i);
                System.out.println("Task removed.");
                return;
            }
        }
        System.out.println("Task not found.");
    }

    public void listTasks() {
        if (tasks.isEmpty()) {
            System.out.println("No tasks available.");
        } else {
            System.out.println("Task List:");
            for (Task task : tasks) {
                System.out.println(task.getId() + ": " + task.getDescription());
            }
        }
    }
}
```

User Interface

The user interface is text-based and interacts with the user via the console. The user is presented with a menu of options to add, remove,

or list tasks. Each option is selected by entering a corresponding number.

OUTPUT:

```
Task List Application
1. Add Task
2. Remove Task
3. List Tasks
4. Exit
Choose an option: 1
Enter task description: Buy groceries
Task added: Buy groceries

Task List Application
1. Add Task
2. Remove Task
3. List Tasks
4. Exit
Choose an option: 3
Task List:
1: Buy groceries

Task List Application
1. Add Task
2. Remove Task
3. List Tasks
4. Exit
```

Challenges Faced

1. Unique Task Identification

Ensuring each task has a unique ID was a key challenge. This was solved by using a static counter in the `Task` class that increments with each new task creation.

2. User Input Handling

Handling user input and providing meaningful feedback for invalid inputs required careful consideration. This was addressed by using a `Scanner` object to read inputs and a switch statement to manage different user choices.

3. List Management

Managing the dynamic list of tasks involved implementing add, remove, and list functionalities. Ensuring the correct task is removed based on its ID was particularly important, requiring iteration over the task list to find the matching task.

Conclusion

The Simple Task List Application provides a basic, yet functional, solution for managing tasks. It demonstrates the principles of object-oriented programming and basic user interface design. The application can be further enhanced by adding features such as task editing, task prioritization, and persistence through file storage or a database. The challenges faced during implementation provided valuable learning experiences in Java programming and application design.