ToDoMan - Java To-Do List App Documentation

ToDoMan - Java To-Do List App Documentation

Project Overview

ToDoMan is a simple and effective desktop-based To-Do List application built using:

- Java
- Swing (for GUI)
- File I/O (for task persistence)

The user can:

- Add tasks
- Mark tasks as done/undone
- Delete tasks
- Save/load tasks automatically from a file (tasks.txt)

Project Structure

```
todoman/
 App.java                // Main entry point
 model/
    Task.java           // Task data class
 ui/
    ToDoFrame.java      // GUI frame with buttons, list
 storage/
    FileStorage.java    // File I/O for saving/loading tasks
```

Task.java (in model package)

This class defines each task's structure.

```
public class Task {
    private String description;
    private boolean done;

    public Task(String description) {
        this.description = description;
        this.done = false;
    }

    public String getDescription() { return description; }
    public boolean isDone() { return done; }
    public void toggleDone() { this.done = !done; }

    public String toString() {
```

```
        return (done ? "[] " : "[ ] ") + description;
    }
}
```

Explanation:

- toggleDone(): Switches done from true to false and vice versa.
- toString(): Returns a displayable version of the task (used by JList).

FileStorage.java (in storage package)

Handles saving/loading tasks to and from a text file tasks.txt

```
public class FileStorage {
    private static final String FILE_NAME = "tasks.txt";

    public static void saveTasks(ArrayList<Task> tasks) {
        try (PrintWriter writer = new FileWriter(FILE_NAME)) {
            for (Task t : tasks) {
                writer.println(t.getDescription() + "::" + t.isDone());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static ArrayList<Task> loadTasks() {
        ArrayList<Task> tasks = new ArrayList<>();
        try (BufferedReader reader = new FileReader(FILE_NAME)) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("::");
                if (parts.length == 2) {
                    Task t = new Task(parts[0]);
                    if (Boolean.parseBoolean(parts[1])) t.toggleDone();
                    tasks.add(t);
                }
            }
        } catch (IOException e) {
            // First-time run: file may not exist.
        }
        return tasks;
    }
```

```
}
```

Explanation:

- saveTasks(): Writes each task's description and status to the file.
- loadTasks(): Reads each line, splits by ::, and recreates task objects.

ToDoFrame.java (in ui package)

The main user interface using Swing.

Key components:

- JTextField taskInput: For entering task names.
- JList<Task> taskList: Displays all tasks.
- DefaultListModel<Task>: Binds tasks to JList.
- Buttons: Add, Delete, Mark Done

```
addButton.addActionListener(e -> {
    String text = taskInput.getText().trim();
    if (!text.isEmpty()) {
        Task newTask = new Task(text);
        taskListModel.addElement(newTask);
        tasks.add(newTask);
        taskInput.setText("");
        FileStorage.saveTasks(tasks);
    }
});
```

Important UI Functions:

- taskListModel.set(index, task): Forces UI to refresh modified task.
- JScrollPane: Enables scrolling for the list.
- BorderLayout, JPanel: Swing layouts to organize components.
- JOptionPane: Can be used for alerts/messages.

App.java (Main Class)

```
public class App {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ToDoFrame());
    }
}
```

Why use SwingUtilities.invokeLater()?

To ensure Swing components run on the Event Dispatch Thread (EDT), keeping UI responsive and error-free.

File Storage Format: tasks.txt

Each line in the file:

<description>::<true/false>

Example:

Buy groceries::true

Finish homework::false

How It Works (Flow Summary):

1. App loads -> reads tasks.txt -> displays tasks.
2. User adds/deletes/marks task -> task list updates.
3. Any change is instantly saved to tasks.txt.
4. On next launch, tasks are restored from the file.

Inbuilt Java APIs Used:

Function/Class	Package	Purpose
JFrame, JPanel	javax.swing	Window and layout UI elements
JList, JTextField	javax.swing	List of tasks and input field
DefaultListModel	javax.swing	Manage list data
FileWriter, BufferedReader	java.io	File save/load
ArrayList	java.util	Store dynamic list of tasks
split()	java.lang.String	Splitting data string by delimiter

Enhancements You Can Add:

- Category-wise tasks (Work, Personal, etc.)
- Due dates or reminders
- Statistics (completed %, total tasks)
- Dark mode UI
- Save data to DB instead of file

Final Note:

ToDoMan is a solid foundational mini-project for beginners learning Java + Swing + File I/O. It's extendable, cleanly structured, and teaches you real-world app flow.