TaskMaster Project - Detailed Instructions

Project Overview

Create a to-do list application using Java that includes:

- User and AdminUser classes (demonstrating inheritance)
- JavaFX GUI for user interaction
- Stack data structure for undo functionality
- File I/O for saving and loading tasks
- Background thread for autosaving

Detailed Implementation Steps

Step 1: Create the User and AdminUser classes

User Class (src/User.java)

- **Fields**:
  - username (String)
  - password (String)
  - tasks (ArrayList<Task>)
- **Methods**:
  - Constructor to initialize fields
  - addTask(Task task): adds a task to the list
  - removeTask(Task task): removes a task from the list
  - getTasks(): returns the list of tasks
  - authenticate(String password): verifies user password
  - getters and setters for fields

AdminUser Class (src/AdminUser.java)

- **Extends**: User class
- **Additional Methods**:
  - clearAllTasks(): removes all tasks
  - Override appropriate methods from User class
  - Additional admin-specific functionality as needed

Step 2: Define the Task Class (src/Task.java)

- **Fields**:
  - title (String)
  - priority (int or enum)
  - completed (boolean)
  - dateCreated (LocalDateTime)
- **Methods**:
  - Constructor(s)
  - getters and setters
  - toString(): for display purposes
  - equals() and hashCode(): for proper comparison
  - Implement Comparable to sort by priority

Step 3: Implement JavaFX GUI

Main Application (src/TaskMasterApp.java)

- Extend javafx.application.Application
- Override start() method to set up the primary stage

Login Screen

- Text fields for username and password
- Login button with event handler for authentication
- New user registration option (if applicable)

Main Task View

- ListView or TableView to display tasks
- Form elements:
  - TextField for task title
  - ComboBox or Slider for priority
  - Button to add new task
  - Button to mark task as completed
  - Button to remove task
  - Undo button
- Menu bar with save/load options

Admin View

- All features from User view

- Additional button for clearAllTasks
- Possibly user management features

Step 4: Implement Stack for Undo Functionality

UndoManager Class (src/UndoManager.java)

- **Fields**:
  - stack (Stack<Command> or Stack<List<Task>>)
- **Methods**:
  - saveState(List<Task>): copies current state to stack
  - undo(): restores previous state
  - canUndo(): checks if stack has elements

Command Pattern (Optional)

- Create Command interface with execute() and undo() methods
- Implement specific command classes (AddTaskCommand, RemoveTaskCommand, etc.)

Step 5: Implement File I/O for Saving Tasks

FileIO Class (src/FileIO.java)

- **Methods**:
  - saveTasks(List<Task> tasks, String filename): writes tasks to file
  - loadTasks(String filename): reads tasks from file and returns list
  - Implementation options:
    - Object serialization
    - CSV/Text format
    - JSON (with external library)
    - XML with JAXB

Integration with GUI

- Add save/load menu items or buttons
- Implement handlers that call FileIO methods
- Add confirmation dialogs

Step 6: Implement Background Autosave

AutoSave Class (src/AutoSave.java)

- Implement Runnable interface

- **Fields**:
  - taskList reference
  - fileIO reference
  - running (boolean)
- **Methods**:
  - constructor to initialize fields
  - run(): periodically calls save method
  - stop(): terminates the autosave thread

Integration with Main Application

- Create ScheduledExecutorService
- Schedule autosave task at fixed intervals (e.g., every 5 minutes)
- Ensure proper shutdown when application closes

Step 7: Implement Error Handling

- Add try-catch blocks for:
  - File operations (FileNotFoundException, IOException)
  - Invalid inputs (NumberFormatException, etc.)
- Add input validation for all user inputs
- Display meaningful error messages using Alert dialogs
- Consider creating custom exceptions for specific error cases

Step 8: Enhance AdminUser Functionality

- Implement clearAllTasks() method:
  - Save current state for undo functionality
  - Clear the task list
  - Update UI to reflect changes
- Add user management features (optional):
  - Create new users
  - Delete users
  - Reset passwords

Project Setup and Compilation

Prerequisites

- Java Development Kit (JDK) 11 or higher

- JavaFX SDK (if using Java 11+)
- IDE (Eclipse, IntelliJ IDEA, or NetBeans) or text editor

Project Structure

```
TaskMaster/
├── src/
│   ├── User.java
│   ├── AdminUser.java
│   ├── Task.java
│   ├── TaskMasterApp.java
│   ├── UndoManager.java
│   ├── FileIO.java
│   ├── AutoSave.java
│   └── [other classes as needed]
├── resources/
│   └── [style sheets, icons, etc.]
└── data/
    └── [saved task files]
```

Compilation

1. Set up JavaFX in your IDE or classpath
2. Compile all Java files: `javac src/*.java`
3. Run the application: `java -cp src TaskMasterApp`

Testing

1. Create test cases for each functionality
2. Test user authentication
3. Verify task operations (add, remove, edit)
4. Test undo functionality
5. Verify save/load operations
6. Test autosave functionality
7. Verify admin-specific features

Extra Credit Ideas

1. Add due dates for tasks
2. Implement task categories/tags

3. Add search functionality
4. Create data visualization for task completion
5. Implement user preferences
6. Add email notifications for overdue tasks

Good luck with your TaskMaster project!