Department of Computer Science

UET Peshawar

Assignment: Task Management System Overview:

In this assignment, you will design and implement a **Task Management System** using Java. The system will allow users to manage their daily tasks efficiently. You will use both `LinkedList` and `ArrayList` to demonstrate their differences in managing dynamic data. The primary objective is to develop a program where tasks can be added, viewed, marked as completed, removed, or reordered.

Requirements:

1. **Task Class:** Create a `Task` class with the following attributes:
   - `String title`: The title of the task.
   - `String description`: A brief description of the task.
   - `boolean isCompleted`: A flag to indicate if the task is completed.
   - `int priority`: A numeric value to represent the task's priority (1 for highest priority).

   Override the `toString()` method to provide a user-friendly display of task details.

2. **Operations:** Implement the following functionalities in your program: ○ **Add Task:** Allow the user to add a task by providing its title, description, and priority. Default `isCompleted` to `false`.
   - **Display All Tasks:** List all tasks with their details. Clearly indicate completed tasks.
   - **Mark Task as Completed:** Let the user mark a task as completed by specifying its title. If the title does not exist, display an appropriate message.
   - **Remove Task:** Allow the user to remove a task by specifying its title. If the title does not exist, display an appropriate message.
   - **Reorder Tasks:** Enable the user to move a task from one position to another in the list. Ensure the priority remains intact.
   - **Filter by Completion:** Provide an option to display only completed or incomplete tasks.
   - **Sort by Priority:** Sort the tasks in descending order of priority.
3. **Data Structures:**
   - Use both `LinkedList` and `ArrayList` to store tasks.
   - Allow the user to choose which structure to use at the start of the program. 4. **User Interface:** Create a menu-driven interface to perform all the above operations. The program

should run until the user chooses to exit.

Hints for Efficient Implementation:

- **Choosing a Data Structure:** Use `LinkedList` for operations involving frequent insertion or removal in the middle of the list. Use `ArrayList` for faster random access or when frequent modifications are not expected.
- **Searching for Tasks:** Use a loop to iterate through the list and find tasks by title. • **Sorting by Priority:** Use the `Collections.sort()` method with a custom comparator to sort tasks based on their priority.
- **Reordering Tasks:** Use the `add()` and `remove()` methods of the chosen list to implement task reordering.
- **Menu Implementation:** Use a `while` loop with a `switch` case for the menu-driven interface.

Expected Output:

The output should match the following scenarios:

1. **Adding Tasks:**

```
Task added successfully: [Title: "Complete Assignment", Description:
"Finish the Java assignment", Priority: 1, Completed: false]
```

2. **Displaying Tasks:**

```
1. [Title: "Complete Assignment", Description: "Finish the Java
assignment", Priority: 1, Completed: false]
2. [Title: "Prepare Presentation", Description: "Create slides for the
meeting", Priority: 2, Completed: true]
```

3. **Marking Task as Completed:**

```
Task marked as completed: "Complete Assignment".
```

4. **Removing a Task:**

```
Task removed successfully: "Prepare Presentation".
```

5. **Reordering Tasks:**

```
Task reordered: "Prepare Presentation" moved to position 1.
```

6. **Filtering Tasks:**

```
Completed Tasks:
```



```
1. [Title: "Prepare Presentation", Description: "Create slides for the
meeting", Priority: 2, Completed: true]
```

7. **Sorting by Priority:**

```
Tasks sorted by priority:
1. [Title: "Complete Assignment", Description: "Finish the Java
assignment", Priority: 1, Completed: false]
2. [Title: "Prepare Presentation", Description: "Create slides for the
meeting", Priority: 2, Completed: true]
```