Name: - T.G.D.S. Chathuranga

Index: - AS2021367

Introduction

The To-Do List Application is a command-line program that allows users to manage a list of tasks. Users can add, remove, complete, and edit tasks, as well as display task information.

Usage

- The program provides a menu-based interface for managing tasks.

- Users can interact with the program to add, remove, complete, edit, sort, and display tasks.

Main Components

1. `**ToDo\_List\_App**` Class

- This is the main class representing the To-Do List Application.

- It contains the `main` method for starting the application.

- It provides methods for displaying menus, adding tasks, and interacting with tasks.

1. `**TaskList**` Class

- This class manages the list of tasks and provides methods for task operations.

- It includes methods for adding, removing, sorting, and displaying tasks.

1. `**Task**` Class

- The `Task` class represents individual tasks.

- It contains fields for title, description, due date, and priority.

- This class is used to create and manage tasks.

1. `**Node**` Class

- The `Node` class represents a node in a linked list.

- It holds a reference to a `Task` object.

- This class is used for creating linked lists of tasks.

Features

1. Adding a Task

- Users can add tasks by providing a title, description, due date, and priority.

- Tasks can be added one by one, and users have the option to add more tasks.

1. Removing All Tasks

- Users can remove all tasks in the list, effectively clearing the list.

1. Sorting the Task List

- Users can sort the task list based on various criteria, including due date, title, description, added order, and priority.

1. Displaying Completed Tasks

- The program can display a list of completed tasks.

1. Displaying Tasks Due Today

- Users can view tasks that are due on the current day.

1. Task Menu

- When interacting with a specific task, a task menu is provided with options for marking as completed, removing the task, and editing the task.

1. Task Edit Menu

- The task edit menu allows users to edit specific task details, including the title, description, due date, and priority.

TaskList Class(Linked List)

Rationale and Design Decisions:

1. Linked List Data Structure: The class uses a linked list to store tasks. Linked lists are efficient for frequent insertions and deletions, which are common operations when managing a to-do list.

2. Add and Remove Methods: The “**addNode**” and “**remove**” methods allow for adding and removing tasks from the list. When removing a task, it searches for the task in the list and removes it.

3. Display Methods: The “**displayList**” method allows for displaying tasks based on different criteria, such as displaying all tasks or only those due today. The method also organizes tasks by their due date for better readability.

4. Sorting: The “**selectionSort**” method provides sorting options based on different criteria, such as title, due date, description, task added order, and priority.

5. Marking Tasks as Complete: The “**markAsComplete**” method marks a task as completed, removes it from the list, and stores it in a separate completed tasks array.

6. Completed Tasks: The class keeps track of completed tasks in a separate array and provides a method to display them.

7. The “**Node”** class serves as a container for individual tasks within the linked list. It stores task data and maintains references to the previous and next nodes, enabling efficient traversal and manipulation of the linked list.

Trade-offs:

1. Memory Usage: The class uses an array to store completed tasks, which has a maximum size of 100. This design choice limits the number of completed tasks that can be stored.

2. Selection Sort: The selection sort algorithm is simple to implement but may not be the most efficient sorting algorithm for larger lists.

Task Class(Abstract data type)

Rationale and Design Decisions:

1. Task Properties: The “**Task**” class defines key properties of a task, including its title, description, due date, status (ongoing or completed), priority level, and a unique task ID.

2. Constructors: The class provides two constructors to allow the creation of tasks with different levels of detail (with or without a description). The first constructor accepts title, description, due date, and priority, while the second constructor sets a default description of "No Description".

3. Marking as Complete: The “markAsComplete” method allows for changing the status of a task to "completed".

4. Status Display: The “**getStatus**” method returns the status of a task as a string as “ongoing” or “completed”.

5. Priority Levels: The “**priority**” field is used to indicate the priority level of a task (e.g., high, medium, or low). The “**setPriority**” method allows for updating the priority level.

6. toString Method: The “**toString**” method generates a string representation of the task, including its Title, Description, Due Date, Priority Level, and Status.