Task Manager App_ST10114043

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 */

package com.mycompany.taskmanagerapp;

/**
 *
 * @author Sizwe Majola
 */

import java.util.Scanner;

public class TaskManagerApp_ST10114043 {
    private TaskManager taskManager;
    private Scanner scanner;
    private String currentDay;

    // Predefined categories and days
    private final String[] CATEGORIES = {"Work", "Personal", "Health", "Learning", "Other"};
    private final String[] DAYS = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"};

    // Constructor
    public TaskManagerApp_ST10114043() {
```

```
    scanner = new Scanner(System.in);
    currentDay = "Monday";
}

// Initialize the application - get user name
public void initialize() {
    System.out.println("================================");
    System.out.println("   Welcome to Task Manager!");
    System.out.println("================================");
    System.out.print("Please enter your name: ");
    String userName = scanner.nextLine().trim();

    taskManager = new TaskManager(userName);
    System.out.println("Hello " + userName + "! Let's manage your tasks.");
    System.out.println();
}

// Display the main menu
public void showMenu() {
    System.out.println("\n=== TASK MANAGER - Current Day: " + currentDay + " ===");
    System.out.println("1. Select Day of Week");
    System.out.println("2. Add Task for " + currentDay);
    System.out.println("3. View All Tasks");
    System.out.println("4. View Tasks for " + currentDay);
    System.out.println("5. Update Task");
    System.out.println("6. Delete Task");
    System.out.println("7. Mark Task Complete/Incomplete");
    System.out.println("8. View Tasks by Category");
```

```
    System.out.println("9. Sort Tasks by Priority");
    System.out.println("10. Show Day Statistics");
    System.out.println("11. Exit");
    System.out.print("Choose an option (1-11): ");
}

// Select day of the week
public void selectDay() {
    System.out.println("\n=== SELECT DAY OF WEEK ===");
    for (int i = 0; i < DAYS.length; i++) {
        System.out.println((i + 1) + ". " + DAYS[i]);
    }
    System.out.print("Choose day (1-7): ");

    try {
        int dayChoice = scanner.nextInt();
        scanner.nextLine();

        if (dayChoice >= 1 && dayChoice <= 7) {
            currentDay = DAYS[dayChoice - 1];
            System.out.println("Current day set to: " + currentDay);
        } else {
            System.out.println("Invalid choice! Day unchanged.");
        }
    } catch (Exception e) {
        System.out.println("Invalid input! Please enter a number.");
        scanner.nextLine();
    }
```

```
}

// Add a new task with predefined categories
public void addTask() {
    System.out.print("Enter task description: ");
    String description = scanner.nextLine().trim();

    if (description.isEmpty()) {
        System.out.println("Description cannot be empty!");
        return;
    }

    // Priority selection
    System.out.print("Enter priority (1-5, where 5 is highest): ");
    int priority = 1;
    try {
        priority = scanner.nextInt();
        scanner.nextLine();
        if (priority < 1 || priority > 5) {
            System.out.println("Invalid priority! Using default (1).");
            priority = 1;
        }
    } catch (Exception e) {
        System.out.println("Invalid input! Using default priority (1).");
        scanner.nextLine();
    }

    // Category selection
```

```
    System.out.println("\nSelect Category:");
    for (int i = 0; i < CATEGORIES.length; i++) {
        System.out.println((i + 1) + ". " + CATEGORIES[i]);
    }
    System.out.print("Choose category (1-5): ");

    String category = CATEGORIES[4];
    try {
        int categoryChoice = scanner.nextInt();
        scanner.nextLine();
        if (categoryChoice >= 1 && categoryChoice <= 5) {
            category = CATEGORIES[categoryChoice - 1];
        } else {
            System.out.println("Invalid choice! Using 'Other' category.");
        }
    } catch (Exception e) {
        System.out.println("Invalid input! Using 'Other' category.");
        scanner.nextLine();
    }

    taskManager.addTask(description, priority, category, currentDay);
}

// Enhanced update method - updates all task properties
public void updateTask() {
    if (taskManager.getTotalTasks() == 0) {
        System.out.println("No tasks available to update.");
        return;
```

```
}

// Show all tasks first
taskManager.displayAllTasks();
System.out.print("Enter task number to update: ");

try {
   int taskNum = scanner.nextInt();
   scanner.nextLine();

   if (taskNum < 1 || taskNum > taskManager.getTotalTasks()) {
      System.out.println("Invalid task number!");
      return;
   }

   System.out.println("\n=== UPDATE TASK #" + taskNum + " ===");
   System.out.println("What would you like to update?");
   System.out.println("1. Description only");
   System.out.println("2. All details (description, priority, category)");
   System.out.print("Choose option (1-2): ");

   int updateChoice = scanner.nextInt();
   scanner.nextLine();

   if (updateChoice == 1) {

      // Update description only (original functionality)
      System.out.print("Enter new description: ");
```

```
            String newDescription = scanner.nextLine().trim();

            if (!newDescription.isEmpty()) {
                taskManager.updateTask(taskNum - 1, newDescription);
            } else {
                System.out.println("Description cannot be empty!");
            }

        } else if (updateChoice == 2) {

            // Update all details
            System.out.print("Enter new description: ");
            String newDescription = scanner.nextLine().trim();

            if (newDescription.isEmpty()) {
                System.out.println("Description cannot be empty!");
                return;
            }

            // Priority selection
            System.out.print("Enter new priority (1-5, where 5 is highest): ");
            int newPriority = 1;
            try {
                newPriority = scanner.nextInt();
                scanner.nextLine();
                if (newPriority < 1 || newPriority > 5) {
                    System.out.println("Invalid priority! Using default (1).");
                    newPriority = 1;
```

```
    }
} catch (Exception e) {
    System.out.println("Invalid input! Using default priority (1).");
    scanner.nextLine();
    newPriority = 1;
}

// Category selection
System.out.println("\nSelect new category:");
for (int i = 0; i < CATEGORIES.length; i++) {
    System.out.println((i + 1) + ". " + CATEGORIES[i]);
}
System.out.print("Choose category (1-5): ");

String newCategory = CATEGORIES[4]; // Default to "Other"
try {
    int categoryChoice = scanner.nextInt();
    scanner.nextLine();
    if (categoryChoice >= 1 && categoryChoice <= 5) {
        newCategory = CATEGORIES[categoryChoice - 1];
    } else {
        System.out.println("Invalid choice! Using 'Other' category.");
    }
} catch (Exception e) {
    System.out.println("Invalid input! Using 'Other' category.");
    scanner.nextLine();
}
```

```
            // Update the task with all new details
            taskManager.updateTaskComplete(taskNum - 1, newDescription, newPriority,
newCategory);

        } else {
            System.out.println("Invalid choice!");
        }

    } catch (Exception e) {
        System.out.println("Invalid input! Please enter a number.");
        scanner.nextLine();
    }
  }

  // Delete a task
  public void deleteTask() {
    if (taskManager.getTotalTasks() == 0) {
        System.out.println("No tasks available to delete.");
        return;
    }

    taskManager.displayAllTasks();
    System.out.print("Enter task number to delete: ");
    try {
        int taskNum = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Are you sure? (y/n): ");
```

```
            String confirm = scanner.nextLine();
            if (confirm.toLowerCase().startsWith("y")) {
                taskManager.deleteTask(taskNum - 1);
            } else {
                System.out.println("Delete cancelled.");
            }
        } catch (Exception e) {
            System.out.println("Invalid input! Please enter a number.");
            scanner.nextLine();
        }
    }

    // Mark a task as complete or incomplete
    public void markTaskCompletion() {
        if (taskManager.getTotalTasks() == 0) {
            System.out.println("No tasks available.");
            return;
        }

        taskManager.displayAllTasks();
        System.out.print("Enter task number: ");
        try {
            int taskNum = scanner.nextInt();
            scanner.nextLine();

            System.out.println("1. Mark as COMPLETED");
            System.out.println("2. Mark as INCOMPLETE");
            System.out.print("Choose option (1-2): ");
```

```
            int choice = scanner.nextInt();
            scanner.nextLine();

            if (choice == 1) {
                taskManager.markTaskComplete(taskNum - 1);
            } else if (choice == 2) {
                taskManager.markTaskIncomplete(taskNum - 1);
            } else {
                System.out.println("Invalid choice!");
            }
        } catch (Exception e) {
            System.out.println("Invalid input! Please enter a number.");
            scanner.nextLine();
        }
    }

    // View tasks by category using predefined categories
    public void viewTasksByCategory() {
        System.out.println("\nSelect Category:");
        for (int i = 0; i < CATEGORIES.length; i++) {
            System.out.println((i + 1) + ". " + CATEGORIES[i]);
        }
        System.out.print("Choose category (1-5): ");

        try {
            int categoryChoice = scanner.nextInt();
            scanner.nextLine();
```

```
            if (categoryChoice >= 1 && categoryChoice <= 5) {
                String category = CATEGORIES[categoryChoice - 1];
                taskManager.displayTasksByCategory(category);
            } else {
                System.out.println("Invalid choice!");
            }
        } catch (Exception e) {
            System.out.println("Invalid input! Please enter a number.");
            scanner.nextLine();
        }
    }

    // Main program loop
    public void run() {
        initialize();

        while (true) {
            showMenu();

            try {
                int choice = scanner.nextInt();
                scanner.nextLine();

                switch (choice) {
                    case 1:
                        selectDay();
                        break;
```

```
case 2:
    addTask();
    break;
case 3:
    taskManager.displayAllTasks();
    break;
case 4:
    taskManager.displayTasksByDay(currentDay);
    break;
case 5:
    updateTask();
    break;
case 6:
    deleteTask();
    break;
case 7:
    markTaskCompletion();
    break;
case 8:
    viewTasksByCategory();
    break;
case 9:
    taskManager.sortTasksByPriority();
    break;
case 10:
    taskManager.showDayStatistics(currentDay);
    break;
case 11:
```

```
System.out.println("\n==========================================================");
                System.out.println("Thank you " + taskManager.getUserName() + " for using Task Manager!");
                System.out.println("         Have a productive day!                ");

System.out.println("==========================================================");
                return;
              default:
                System.out.println("Invalid choice! Please select 1-11.");
            }

            // Pause before showing menu again
            System.out.print("\nPress Enter to continue...");
            scanner.nextLine();

        } catch (Exception e) {
            System.out.println(" Invalid input! Please enter a number.         ");

System.out.println("==========================================================");
            scanner.nextLine(); // Clear invalid input
        }
    }
}

// Main method
public static void main(String[] args) {
```

```
        TaskManagerApp_ST10114043 app = new TaskManagerApp_ST10114043();
        app.run();
    }
}
```

TaskManager:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
 */
package com.mycompany.taskmanagerapp;

/**
 *
 * @author Sizwe Majola
 */

public class TaskManager {
    // Declarations
    private Task[] tasks;
    private int taskCount;
    private final int INITIAL_SIZE = 5;
    private String userName;

    // Constructor
    public TaskManager(String userName) {
        this.userName = userName;
        tasks = new Task[INITIAL_SIZE];
        taskCount = 0;
    }

    // // Array to Enable User to Add Tasks
```

```
private void expandArray() {
    System.out.println("Array is full! Expanding...");

    // Making Array Twice as Big as Previous Array
    Task[] newTasks = new Task[tasks.length * 2];



    for (int i = 0; i < taskCount; i++) {
        newTasks[i] = tasks[i];
    }

    //Putting New Array in the Place of the Old One
    tasks = newTasks;
    System.out.println("Array expanded to size: " + tasks.length);
}

// Adding a New Task
public void addTask(String description, int priority, String category, String currentDay) {

    if (taskCount >= tasks.length) {
        expandArray();
    }

    // Add new task to array and set the day
    tasks[taskCount] = new Task(description, priority, category);
    tasks[taskCount].setDayOfWeek(currentDay);
    taskCount++;
    System.out.println("Task added successfully for " + currentDay + "!");
```

```
    }

    // Method Displaying All Tasks
    public void displayAllTasks() {
        if (taskCount == 0) {
            System.out.println("No tasks available.");
            return;
        }

        System.out.println("\n=== ALL TASKS ===");
        for (int i = 0; i < taskCount; i++) {
            System.out.println("Task #" + (i + 1) + ":");
            tasks[i].displayTask();
        }
    }

    // Method to Display Tasks For Selected Day
    public void displayTasksByDay(String dayOfWeek) {
        System.out.println("\n=== TASKS FOR " + dayOfWeek.toUpperCase() + " ===");
        boolean found = false;

        for (int i = 0; i < taskCount; i++) {
            if (tasks[i].getDayOfWeek() != null &&
tasks[i].getDayOfWeek().equals(dayOfWeek)) {
                System.out.println("Task #" + (i + 1) + ":");
                tasks[i].displayTask();
                found = true;
            }
```

```
    }

    if (!found) {
      System.out.println("No tasks found for " + dayOfWeek + ".");
    }
  }

  // Method to Show Stats For Selected Day
  public void showDayStatistics(String dayOfWeek) {
    int totalTasksForDay = 0;
    int completedTasksForDay = 0;

    for (int i = 0; i < taskCount; i++) {
      if (tasks[i].getDayOfWeek() != null &&
tasks[i].getDayOfWeek().equals(dayOfWeek)) {
        totalTasksForDay++;
        if (tasks[i].isCompleted()) {
          completedTasksForDay++;
        }
      }
    }

    System.out.println("\n=== " + dayOfWeek.toUpperCase() + " STATISTICS ===");
    System.out.println("Total tasks: " + totalTasksForDay);
    System.out.println("Completed tasks: " + completedTasksForDay);
    System.out.println("Pending tasks: " + (totalTasksForDay - completedTasksForDay));

    if (totalTasksForDay > 0) {
```

```
        double completionRate = (double) completedTasksForDay / totalTasksForDay *
100;
        System.out.println("Completion rate: " + String.format("%.1f", completionRate) +
"%");
      } else {
        System.out.println("Completion rate: 0.0%");
      }
    }

    // Get user name
    public String getUserName() {
      return userName;
    }

    // Method to Update Selected Task
    public boolean updateTask(int index, String newDescription) {

      if (index < 0 || index >= taskCount) {
        System.out.println("Invalid task number!");
        return false;
      }

      // Updating Task Description
      tasks[index].setDescription(newDescription);
      System.out.println("Task updated successfully!");
      return true;
    }

    // Method Updating Task Properties
```

```
    public boolean updateTaskComplete(int index, String newDescription, int newPriority,
String newCategory) {
        // Check if index is valid
        if (index < 0 || index >= taskCount) {
            System.out.println("Invalid task number!");
            return false;
        }



        tasks[index].setDescription(newDescription);
        tasks[index].setPriority(newPriority);
        tasks[index].setCategory(newCategory);

        System.out.println("Task updated successfully!");
        System.out.println("Updated details:");
        System.out.println("- Description: " + newDescription);
        System.out.println("- Priority: " + newPriority);
        System.out.println("- Category: " + newCategory);
        return true;
    }

    // Method to Delete Task
    public boolean deleteTask(int index) {
        // Check if index is valid
        if (index < 0 || index >= taskCount) {
            System.out.println("Invalid task number!");
            return false;
        }
```

```
    // Shift all elements after the deleted element one position left
    for (int i = index; i < taskCount - 1; i++) {
        tasks[i] = tasks[i + 1];
    }

    // Remove reference to last element and decrease count
    tasks[taskCount - 1] = null;
    taskCount--;

    System.out.println("Task deleted successfully!");
    return true;
}

// Array Searching for Selected Category
public void displayTasksByCategory(String category) {
    System.out.println("\n=== TASKS IN CATEGORY: " + category + " ===");
    boolean found = false;

    for (int i = 0; i < taskCount; i++) {
        if (tasks[i].getCategory().equals(category)) {
            System.out.println("Task #" + (i + 1) + ":");
            tasks[i].displayTask();
            found = true;
        }
    }

    if (!found) {
```

```
        System.out.println("No tasks found in category: " + category);
    }
}

//Array Used to Sort in order of Priority, going from High to Low
public void sortTasksByPriority() {
    if (taskCount <= 1) {
        System.out.println("Not enough tasks to sort.");
        return;
    }

    //New Array to Efficiently Sort Array
    Task[] sortedTasks = new Task[taskCount];
    for (int i = 0; i < taskCount; i++) {
        sortedTasks[i] = tasks[i];
    }

    // Bubble Sort Algorithm, sorting Array from High to Low
    for (int i = 0; i < taskCount - 1; i++) {
        for (int j = 0; j < taskCount - i - 1; j++) {
            if (sortedTasks[j].getPriority() < sortedTasks[j + 1].getPriority()) {
                // Swaping and Sorting Array Accordingly
                Task temp = sortedTasks[j];
                sortedTasks[j] = sortedTasks[j + 1];
                sortedTasks[j + 1] = temp;
            }
        }
    }
```

```
    // Method Displaying Sorted Tasks
    System.out.println("\n=== TASKS SORTED BY PRIORITY (Highest First) ===");
    for (int i = 0; i < taskCount; i++) {
        System.out.println("Task #" + (i + 1) + ":");
        sortedTasks[i].displayTask();
    }
}

// Method to Mark Task as Complete
public boolean markTaskComplete(int index) {
    if (index < 0 || index >= taskCount) {
        System.out.println("Invalid task number!");
        return false;
    }

    tasks[index].setCompleted(true);
    System.out.println("Task marked as complete!");
    return true;
}

// Method to Mark Task as Inomplete
public boolean markTaskIncomplete(int index) {
    if (index < 0 || index >= taskCount) {
        System.out.println("Invalid task number!");
        return false;
    }
```

```
        tasks[index].setCompleted(false);
        System.out.println("Task marked as incomplete!");
        return true;
    }

    // Get total number of tasks
    public int getTotalTasks() {
        return taskCount;
    }

    // Get number of completed tasks
    public int getCompletedTasks() {
        int completed = 0;
        for (int i = 0; i < taskCount; i++) {
            if (tasks[i].isCompleted()) {
                completed++;
            }
        }
        return completed;
    }
}
```

Task Item:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
 */
package com.mycompany.taskmanagerapp;
/**
 *
 * @author Sizwe Majola
 */

public abstract class TaskItem {

    // Declarations
    protected String description;
    protected boolean completed;
    protected String dayOfWeek;

    // Constructor for these fields
    public TaskItem(String description) {
        this.description = description;
        this.completed = false;
        this.dayOfWeek = null;
    }

    // Getter and setter methods
    public String getDescription() {
```

```
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public boolean isCompleted() {
        return completed;
    }

    public void setCompleted(boolean completed) {
        this.completed = completed;
    }

    public String getDayOfWeek() {
        return dayOfWeek;
    }

    public void setDayOfWeek(String dayOfWeek) {
        this.dayOfWeek = dayOfWeek;
    }

    // Child Class implementing Abstract Class
    public abstract String getStatus();
    public abstract void displayTask();
}
```

Task:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java
 */
package com.mycompany.taskmanagerapp;
/**
 *
 * @author Sizwe Majola
 */

public class Task extends TaskItem {

    // Declarations for Priorities and Categories
    private int priority;
    private String category;

    // Constructor calling parent constructor using super class
    public Task(String description, int priority, String category) {
        super(description); //
        this.priority = priority;
        this.category = category;
    }

    // Getter and setter methods
    public int getPriority() {
        return priority;
```

```
}

public void setPriority(int priority) {
   this.priority = priority;
}

public String getCategory() {
   return category;
}

public void setCategory(String category) {
   this.category = category;
}

// Implement Abstract Method from Parent Class
@Override
public String getStatus() {
   if (completed) {
      return "COMPLETED";
   } else {
      return "INCOMPLETE";
   }
}

// Implement abstract method from parent class
@Override
public void displayTask() {
   System.out.println("Task: " + description);
```

```
        System.out.println("Day: " + (dayOfWeek != null ? dayOfWeek : "Not specified"));
        System.out.println("Priority: " + priority);
        System.out.println("Category: " + category);
        System.out.println("Status: " + getStatus());
        System.out.println("--------------------");
    }
}
```