Task Management System

Masterclass in Java

Luka Shamatava
Luka.shamatava.1@iliauni.edu.ge

Description

Create a Task Management System (TMS) in Java, a software widely used across industries and terms to organize tasks and projects efficiently. This example provides a basic implementation with the following features:

- Task Storage
- Task Addition
- Task removal

TMS structure

We will need following classes for the software:

- Task – Represents a task with attributes like Id, title, description and etc.
- Task Manager – Manages tasks by providing functionalities for adding task and removing tasks
- Main - Contains the main method to initiate the Task Management System and handle user input

Class Task

The class “Task” should represent a task with various attributes such as task ID,title,description,assignee,due date, completion status and priority.the constructor initializes theres attributes when a new task object is created:

```
import java.util.Date;

public class Task {
    private int taskId;
    private String title;
    private String description;
    private String assignee;
    private Date dueDate;
    private boolean isCompleted;
    private String priority;

    public Task(int taskId, String title, String description, String
assignee, Date dueDate, String priority) {
        this.taskId = taskId;
        this.title = title;
        this.description = description;
        this.assignee = assignee;
        this.dueDate = dueDate;
        this.isCompleted = false;
        this.priority = priority;
    }

    public int getTaskId() {
        return taskId;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public String getAssignee() {
        return assignee;
    }

    public Date getDueDate() {
        return dueDate;
    }

    public boolean isCompleted() {
        return isCompleted;
    }

    public String getPriority() {
        return priority;
    }

    public void markAsCompleted() {
        this.isCompleted = true;
    }

    @Override
    public String toString() {
        return "Task ID: " + taskId + "\nTitle: " + title + "\nDescription: "
+ description +
                "\nAssignee: " + assignee + "\nDue Date: " + dueDate +
"\nPriority: " + priority +
                "\nCompleted: " + isCompleted;
    }
}
```

Getters and setters are provided for accessing and modifying the task attributes. The ‘markAsCompleted()’ method allows marking a task as completed by setting the ‘isCompleted’ flag to ‘true’. The ‘toString()’ method overrides the default implementation provided by attributes.

Class TaskManager

The ‘TaskManager’ class is responsible for managing tasks within the Task Management System. Its key components are constructor, add Task method, remove Task method, get task by id and markAsComplated:

```java
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class TaskManager {
    private List<Task> tasks;
    private int nextTaskId;

    public TaskManager() {
        this.tasks = new ArrayList<>();
        this.nextTaskId = 1; // Initialize the next task ID to 1
    }

    public void addTask(String title, String description, String assignee,
Date dueDate, String priority) {
        Task task = new Task(nextTaskId++, title, description, assignee,
dueDate, priority);
        tasks.add(task);
    }

    public void removeTask(int taskId) {
        Task taskToRemove = null;
        for (Task task : tasks) {
            if (task.getTaskId() == taskId) {
                taskToRemove = task;
                break;
            }
        }
        if (taskToRemove != null) {
            tasks.remove(taskToRemove);
        } else {
            System.out.println("Task with ID " + taskId + " not found.");
        }
    }

    public Task getTaskById(int taskId) {
        for (Task task : tasks) {
            if (task.getTaskId() == taskId) {
                return task;
            }
        }
        return null; // Task not found
    }

    public void markTaskAsCompleted(int taskId) {
        Task task = getTaskById(taskId);
        if (task != null) {
            task.markAsCompleted();
            System.out.println("Task with ID " + taskId + " marked as
completed.");
        } else {
            System.out.println("Task with ID " + taskId + " not found.");
        }
    }
    public List<Task> getTasks() {
        return tasks;
    }
}
```

Class Main

Now lets create a simple tester to test our Task management system:

```
import java.util.Date;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        TaskManager taskManager = new TaskManager();

        // Adding tasks with additional details
        taskManager.addTask("Task 1", "Description of Task 1", "Luka
Shamatava", new Date(), "High");
        taskManager.addTask("Task 2", "Description of Task 2", "Nikoloz
Smith", new Date(), "Medium");
        taskManager.addTask("Task 3", "Description of Task 3", "Alex
Johnson", new Date(), "Low");

        // Displaying tasks
        List<Task> tasks = taskManager.getTasks();
        System.out.println("All Tasks:");
        for (int i = 0; i < tasks.size(); i++) {
            System.out.println("Task " + (i + 1) + ":");
            System.out.println(tasks.get(i));
            System.out.println();
        }

        // Marking a task as completed
        taskManager.markTaskAsCompleted(1);

        // Displaying tasks after marking one as completed
        System.out.println("All Tasks after marking Task 2 as completed:");
        for (int i = 0; i < tasks.size(); i++) {
            System.out.println("Task " + (i + 1) + ":");
            System.out.println(tasks.get(i));
            System.out.println();
        }

        // Removing a task by ID
        taskManager.removeTask(3);

        // Displaying tasks after removing a task
        System.out.println("All Tasks after removing Task 3:");
        for (int i = 0; i < tasks.size(); i++) {
            System.out.println("Task " + (i + 1) + ":");
            System.out.println(tasks.get(i));
            System.out.println();
        }

        // Searching for a task by ID
        Task task = taskManager.getTaskById(2);
        if (task != null) {
            System.out.println("Task found by ID 2:");
            System.out.println(task);
        } else {
            System.out.println("Task with ID 2 not found.");
        }
    }
}
```

Future Improvements

This example above is simple for the TMS. It can be improved by adding some other features for example:

- User Authentication
- Database Integration
- Notification System
- Gui Implementation

(I implemented my actual code for task 3)