// ScheduleManager.java

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Logger;

public class ScheduleManager {

private static ScheduleManager instance;

private List<Task> tasks;

private TaskFactory taskFactory;

private Logger logger;

private ScheduleManager() {

tasks = new ArrayList<>();

taskFactory = new TaskFactory();

logger = Logger.getLogger(ScheduleManager.class.getName());

}

public static ScheduleManager getInstance() {

if (instance == null) {

instance = new ScheduleManager();

}

return instance;

}

public void addTask(String description, String startTime, String endTime, String priority) {

Task task = taskFactory.createTask(description, startTime, endTime, priority);

if (task != null) {

if (hasConflict(task)) {

logger.warning("Task conflicts with existing task");

System.out.println("Error: Task conflicts with existing task");

} else {

tasks.add(task);

logger.info("Task added successfully");

System.out.println("Task added successfully. No conflicts.");

}

} else {

logger.warning("Invalid task creation");

System.out.println("Error: Invalid task creation");

}

}

public void removeTask(String description) {

Task task = findTask(description);

if (task != null) {

tasks.remove(task);

logger.info("Task removed successfully");

System.out.println("Task removed successfully");

} else {

logger.warning("Task not found");

System.out.println("Error: Task not found");

}

}

public void viewTasks() {

if (tasks.isEmpty()) {

logger.info("No tasks scheduled for the day");

System.out.println("No tasks scheduled for the day");

} else {

tasks.sort((t1, t2) -> t1.getStartTime().compareTo(t2.getStartTime()));

for (Task task : tasks) {

System.out.println(task.toString());

}

}

}

private boolean hasConflict(Task newTask) {

for (Task existingTask : tasks) {

if (existingTask.getStartTime().compareTo(newTask.getEndTime()) < 0

&& existingTask.getEndTime().compareTo(newTask.getStartTime()) > 0) {

return true;

}

}

return false;

}

private Task findTask(String description) {

for (Task task : tasks) {

if (task.getDescription().equals(description)) {

return task;

}

}

return null;

}

}

// TaskFactory.java

import java.time.LocalTime;

public class TaskFactory {

public Task createTask(String description, String startTime, String endTime, String priority) {

try {

LocalTime start = LocalTime.parse(startTime);

LocalTime end = LocalTime.parse(endTime);

return new Task(description, start, end, priority);

} catch (Exception e) {

return null;

}

}

}

// Task.java

import java.time.LocalTime;

public class Task {

private String description;

private LocalTime startTime;

private LocalTime endTime;

private String priority;

public Task(String description, LocalTime startTime, LocalTime endTime, String priority) {

this.description = description;

this.startTime = startTime;

this.endTime = endTime;

this.priority = priority;

}

public String getDescription() {

return description;

}

public LocalTime getStartTime() {

return startTime;

}

public LocalTime getEndTime() {

return endTime;

}

public String getPriority() {

return priority;

}

@Override

public String toString() {

return String.format("%s - %s: %s [%s]", startTime, endTime, description, priority);

}

}

// Main.java

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

ScheduleManager scheduleManager = ScheduleManager.getInstance();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("1. Add task");

System.out.println("2. Remove task");

System.out.println("3. View tasks");

System.out.println("4. Exit");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline left-over

switch (choice) {

case 1:

System.out.print("Enter task description: ");

String description = scanner.nextLine();

System.out.print("Enter start time (HH:mm): ");

String startTime = scanner.nextLine();

System.out.print("Enter end time (HH:mm): ");

String endTime = scanner.nextLine();

System.out.print("Enter priority: ");

String priority = scanner.nextLine();

scheduleManager.addTask(description, startTime, endTime, priority);

break;

case 2:

System.out.print("Enter task description: ");

description = scanner.nextLine();

scheduleManager.removeTask