POM.XML

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>taskapi</artifactId>

<version>1.0-SNAPSHOT</version>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.3</version>

</parent>

<dependencies>

<!-- Spring Boot Web dependency -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<properties>

<java.version>17</java.version>

</properties>

</project>

Task.java

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

// TaskExecution class

class TaskExecution {

private Date startTime;

private Date endTime;

private String output;

public TaskExecution() {}

public TaskExecution(Date startTime, Date endTime, String output) {

this.startTime = startTime;

this.endTime = endTime;

this.output = output;

}

public Date getStartTime() { return startTime; }

public void setStartTime(Date startTime) { this.startTime = startTime; }

public Date getEndTime() { return endTime; }

public void setEndTime(Date endTime) { this.endTime = endTime; }

public String getOutput() { return output; }

public void setOutput(String output) { this.output = output; }

}

// Task class

class Task {

private String id;

private String name;

private String owner;

private String command;

private List<TaskExecution> taskExecutions = new ArrayList<>();

public Task() {}

public Task(String id, String name, String owner, String command, List<TaskExecution> taskExecutions) {

this.id = id;

this.name = name;

this.owner = owner;

this.command = command;

this.taskExecutions = taskExecutions;

}

public String getId() { return id; }

public void setId(String id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getOwner() { return owner; }

public void setOwner(String owner) { this.owner = owner; }

public String getCommand() { return command; }

public void setCommand(String command) { this.command = command; }

public List<TaskExecution> getTaskExecutions() { return taskExecutions; }

public void setTaskExecutions(List<TaskExecution> taskExecutions) { this.taskExecutions = taskExecutions; }

}

// Main class to test

public class Main {

public static void main(String[] args) {

TaskExecution exec1 = new TaskExecution(new Date(), new Date(), "Hello World!");

List<TaskExecution> executions = new ArrayList<>();

executions.add(exec1);

Task task = new Task("123", "Print Hello", "John Smith", "echo Hello World!", executions);

System.out.println("Task Name: " + task.getName());

System.out.println("Owner: " + task.getOwner());

System.out.println("Command: " + task.getCommand());

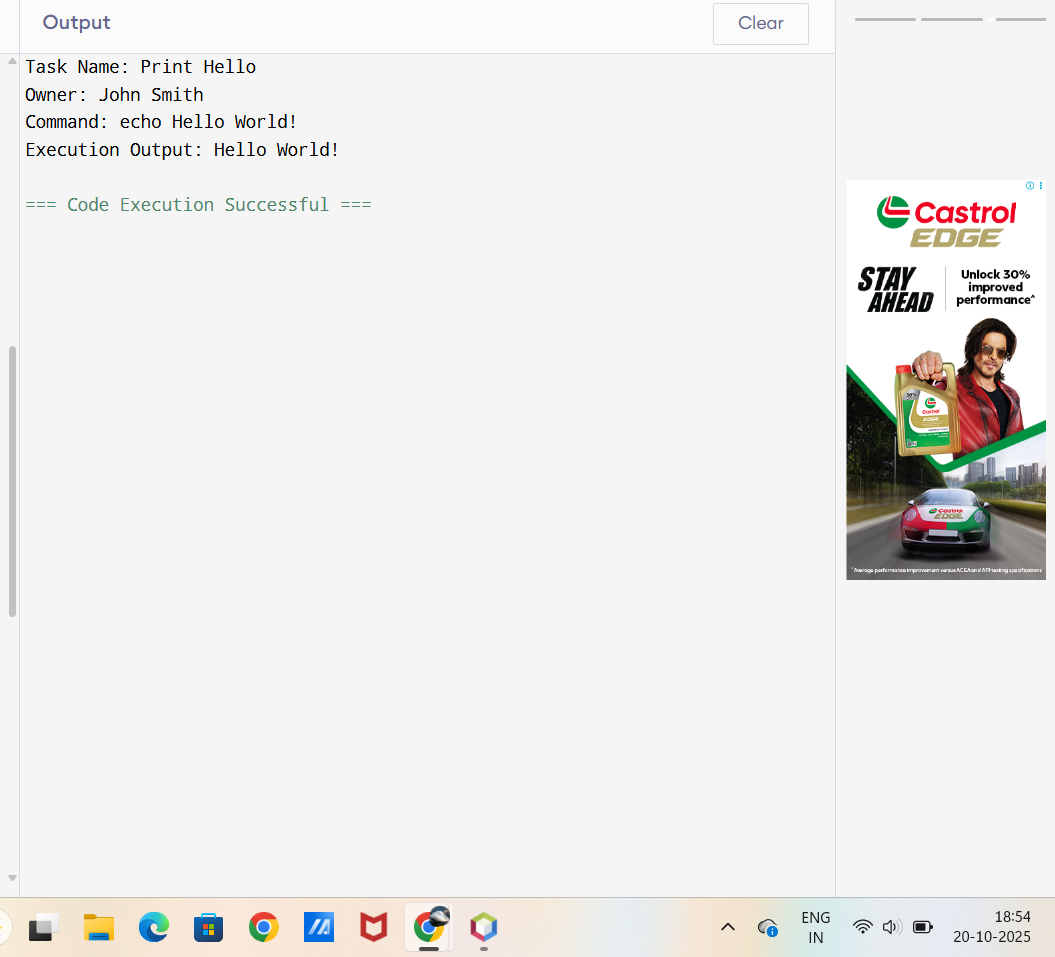
for (TaskExecution te : task.getTaskExecutions()) {

System.out.println("Execution Output: " + te.getOutput());

}

}

}



TaskExecution.java

Import java.util.Date;

public class TaskExecution {

// ✅ Fields (class variables)

private Date startTime;

private Date endTime;

private String output;

// ✅ Default constructor

public TaskExecution() {}

// ✅ Constructor with parameters

public TaskExecution(Date startTime, Date endTime, String output) {

this.startTime = startTime;

this.endTime = endTime;

this.output = output;

}

// ✅ Getters and setters

public Date getStartTime() { return startTime; }

public void setStartTime(Date startTime) { this.startTime = startTime; }

public Date getEndTime() { return endTime; }

public void setEndTime(Date endTime) { this.endTime = endTime; }

public String getOutput() { return output; }

public void setOutput(String output) { this.output = output; }

// ✅ Optional main method to test the class

public static void main(String[] args) {

TaskExecution exec = new TaskExecution(

new Date(), new Date(), "Hello World"

);

System.out.println(exec);

}

// ✅ toString() for display

@Override

public String toString() {

return "TaskExecution{" +

"startTime=" + startTime +

", endTime=" + endTime +

", output='" + output + '\'' +

'}';

}

}



import java.util.\*;

import java.io.\*;

class TaskExecution {

Date startTime;

Date endTime;

String output;

TaskExecution(Date startTime, Date endTime, String output) {

this.startTime = startTime;

this.endTime = endTime;

this.output = output;

}

public String toString() {

return "Start: " + startTime + "\nEnd: " + endTime + "\nOutput:\n" + output;

}

}

class Task {

String id;

String name;

String owner;

String command;

List<TaskExecution> executions = new ArrayList<>();

Task(String id, String name, String owner, String command) {

this.id = id;

this.name = name;

this.owner = owner;

this.command = command;

}

public String toString() {

return "ID: " + id + "\nName: " + name + "\nOwner: " + owner + "\nCommand: " + command + "\nExecutions: " + executions.size();

}

}

public class Main { // <-- This is the class with main method

static Map<String, Task> tasks = new HashMap<>();

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

while (true) {

System.out.println("\n=== TASK MANAGER ===");

System.out.println("1. Create Task");

System.out.println("2. List All Tasks");

System.out.println("3. Search Task by Name");

System.out.println("4. Delete Task by ID");

System.out.println("5. Execute Task by ID");

System.out.println("6. Show Task Executions");

System.out.println("0. Exit");

System.out.print("Choose option: ");

String choice = sc.nextLine();

switch (choice) {

case "1": createTask(); break;

case "2": listTasks(); break;

case "3": searchTask(); break;

case "4": deleteTask(); break;

case "5": executeTask(); break;

case "6": showExecutions(); break;

case "0": System.exit(0);

default: System.out.println("Invalid choice");

}

}

}

static void createTask() {

System.out.print("ID: "); String id = sc.nextLine();

System.out.print("Name: "); String name = sc.nextLine();

System.out.print("Owner: "); String owner = sc.nextLine();

System.out.print("Command: "); String command = sc.nextLine();

if (!isCommandSafe(command)) {

System.out.println("Unsafe command detected!");

return;

}

Task t = new Task(id, name, owner, command);

tasks.put(id, t);

System.out.println("Task created!");

}

static void listTasks() {

if (tasks.isEmpty()) { System.out.println("No tasks found"); return; }

for (Task t : tasks.values()) {

System.out.println("-------------------");

System.out.println(t);

}

}

static void searchTask() {

System.out.print("Enter name to search: ");

String name = sc.nextLine().toLowerCase();

boolean found = false;

for (Task t : tasks.values()) {

if (t.name.toLowerCase().contains(name)) {

System.out.println("-------------------");

System.out.println(t);

found = true;

}

}

if (!found) System.out.println("No task found");

}

static void deleteTask() {

System.out.print("Enter Task ID to delete: ");

String id = sc.nextLine();

if (tasks.remove(id) != null) System.out.println("Task deleted");

else System.out.println("Task not found");

}

static void executeTask() {

System.out.print("Enter Task ID to execute: ");

String id = sc.nextLine();

Task t = tasks.get(id);

if (t == null) { System.out.println("Task not found"); return; }

Date start = new Date();

StringBuilder output = new StringBuilder();

try {

Process process = Runtime.getRuntime().exec(t.command);

BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));

String line;

while ((line = reader.readLine()) != null) { output.append(line).append("\n"); }

process.waitFor();

} catch (Exception e) { output.append("Error: ").append(e.getMessage()); }

Date end = new Date();

TaskExecution exec = new TaskExecution(start, end, output.toString());

t.executions.add(exec);

System.out.println("Execution done:\n" + exec);

}

static void showExecutions() {

System.out.print("Enter Task ID to view executions: ");

String id = sc.nextLine();

Task t = tasks.get(id);

if (t == null) { System.out.println("Task not found"); return; }

if (t.executions.isEmpty()) { System.out.println("No executions yet"); return; }

for (TaskExecution e : t.executions) {

System.out.println("-------------------");

System.out.println(e);

}

}

static boolean isCommandSafe(String cmd) {

String[] blacklist = {"rm","shutdown","reboot","&&","|","curl","wget"};

for (String b : blacklist) if (cmd.contains(b)) return false;

return true;

}

}

