1. **Problem Statement:**

Task Conflict Notification System Using the Observer Pattern

You must organize other users’ notifications if any task that the user attempts to add conflicts to with an existing task. In this case, the user will be able to add tasks and keep record of the list of observers (users) who shall be notified incase of any conflicts whenever new tasks are added. A task conflict is defined to be an occurrence of a task with two or more system resources containing the same task name.

Key Requirements:

Users should get a conflict warning whenever they attempt to create a task which already exists in the methods database.

When a conflict arises and needs to be reported, there can be more than one user (observers) who will be sent the notifications.

Task classification system must also hold the thinking observables and task observables.

1. **Explanation of the Code :**

**Imports:**

import java.util.ArrayList;

import java.util.List; These import statements bring in the ArrayList and List classes from the Java Collections Framework and these are applied in tasks and observers storage resources.

**Observer Interface:**

The Observer interface declares an update(String message) method that a relative class that desires to get the notifications (endeavor) further seeks to implement out.

**TaskManager Class (Subject):**

Field:

observers (List of Observer objects) - Keeps track of all users who agreed to receive the notifications.

tasks (List of String objects) - Holds all the tasks added to the system.

Method:

addObserver(Observer observer): It appends a new observer (a user) to the observers list, which encompasses all the users.

addTask(String task): Task is added into the system by the user. Before adding, it checks if the task exists already. If there is such task already, there would be a message to all observers about this conflict. NotifyObservers(String message): The procedure whereby the message about the conflict is sent to all the observers by looping round the observers list and calling each observer’s update method with the conflict message.

**User Class (Observer Implementation):**

Field:

name – Keeps the user information in terms of a name.

Constructor: Assigns a name value to the user.

Method:

Implementing the update(String message) method from the Observer interface this time, the user receives the conflict message and outputs it to the console upon the finishing of this method.

**Main Class (ObserverPatternExample):**

Task: Demonstrate how the observer pattern is used.

Code Flow:

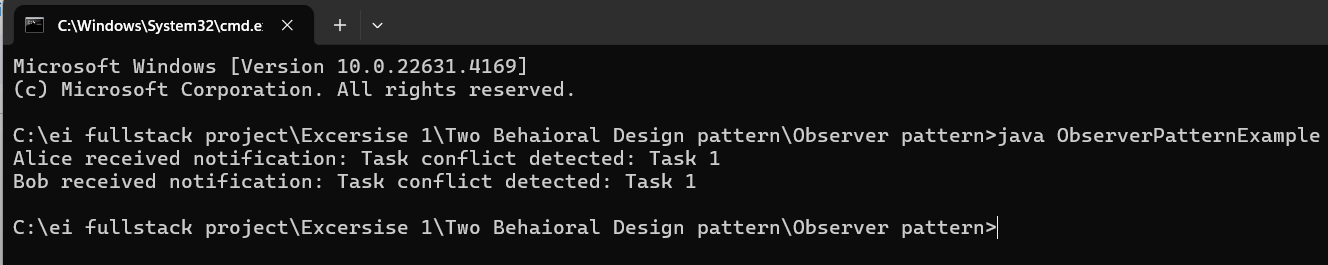
Create a TaskManager - this object will be responsible for tasks and it will notify observers to declare changes in tasks.

Two User objects are created and added as observers into the TaskManager: user1 with the name "Alice" and user2 with the name "Bob"

A couple of tasks are added using addTask().

If some task is added already existing (e.g. "Task 1" was added twice), both users are notified of conflict.

**3.Output**

****