**Assignment 2**

**Altair Alimzhanov IT-2004**

**Observer Pattern.**

Observer is a behavioral design pattern. It specifies communication between objects: observable and observers. An observable is an object which notifies observers about the changes in its state.

My example: A teacher in an online school leaves a request for working hours with groups and later he should receive notifications about free groups or windows.

package com.company;  
  
public class Main {  
  
 public static void main(String[] args) {  
 TutorOnlineSite tutorSite = new TutorOnlineSite();  
  
 tutorSite.addWindow("First free learning window");  
 tutorSite.addWindow("Second free learning window");  
  
 Observer firstSubcriber = new Tutor("Altair Alimzhanov");  
 Observer secondSubcriber = new Tutor("Umma Turman");  
  
 tutorSite.addObserver(firstSubcriber);  
 tutorSite.addObserver(secondSubcriber);  
  
 tutorSite.addWindow("Third free learning window");  
 }  
}

package com.company;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class TutorOnlineSite implements Observed{  
  
 List<String> freeWindows = new ArrayList<>();  
  
 List<Observer> Tutor = new ArrayList<>();  
  
 public void addWindow(String window){  
 this.freeWindows.add(window);  
 notifyObservers();  
 }  
  
 public void removeWindow(String window){  
 this.freeWindows.remove(window);  
 notifyObservers();  
 }  
  
 @Override  
 public void addObserver(Observer observer) {  
 this.Tutor.add(observer);  
 }  
  
 @Override  
 public void removeObserver(Observer observer) {  
 this.Tutor.remove(observer);  
 }  
  
 @Override  
 public void notifyObservers() {  
 for(Observer observer: Tutor){  
 observer.eventHandling(this.freeWindows);  
 }  
 }  
}

package com.company;  
  
import java.util.List;  
  
public class Tutor implements Observer{  
 String name;  
  
 public Tutor(String name){  
 this.name = name;  
 }  
  
 @Override  
 public void eventHandling(List<String> freeWindows){  
 System.*out*.println("Dear tutor " + name + "\nWe have some free windows among groups:\n" + freeWindows + "\n---------------------------------------------------------------\n" );  
 }  
}

package com.company;  
  
public interface Observed {  
 public void addObserver(Observer observer);  
  
 public void removeObserver(Observer observer);  
  
 public void notifyObservers();  
}

package com.company;  
  
import java.util.List;  
  
public interface Observer {  
 public void eventHandling(List<String> freeWindows);  
}