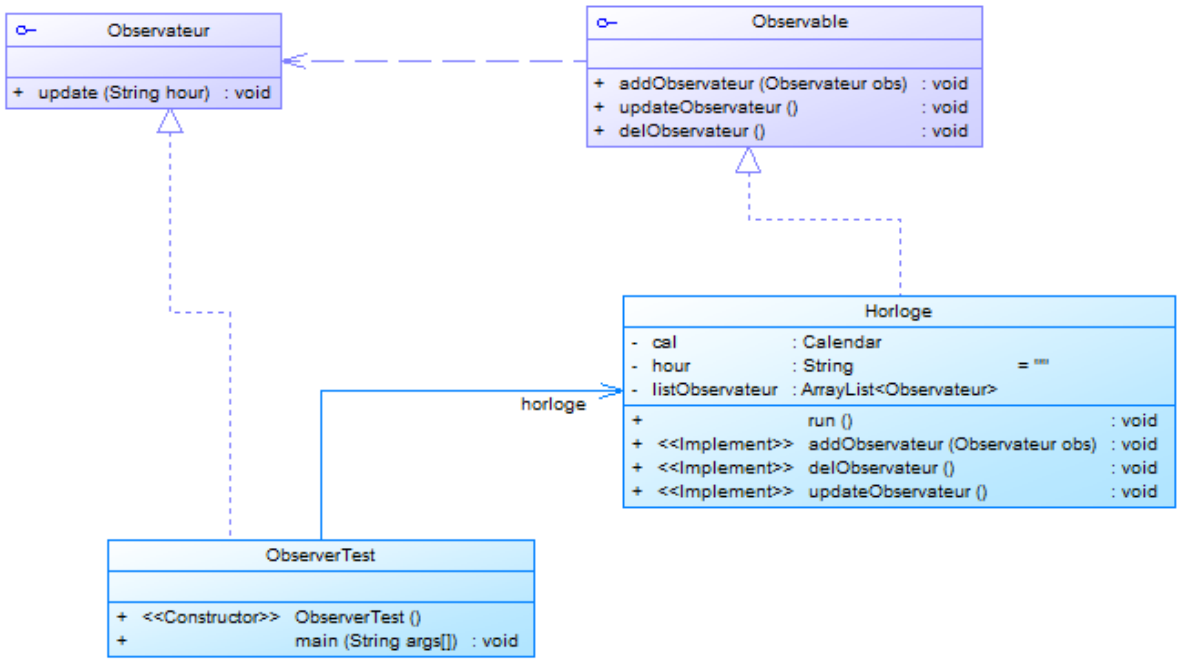
Observer Pattern

Etape 1 : ***Créez un projet java sous eclipse***

nom **: Gof_Observer**



Etape 2 : ***Les classes***

Observateur.java

```
package com.formation.observer;

public interface Observateur {
	public void update(String hour);
}
```

Observable.java

```
package com.formation.observer;

public interface Observable {
	public void addObservateur(Observateur obs);

	public void updateObservateur();

	public void delObservateur();
}
```

Horloge.java

```
package com.formation.observer;

import java.util.ArrayList;
import java.util.Calendar;

public class Horloge implements Observable{
  //On récupère l'instance d'un calendrier
  //Elle va nous permettre de récupérer l'heure actuelle
  private Calendar cal;
  private String hour = "";
  //Notre collection d'observateurs
  private ArrayList<Observateur> listObservateur = new ArrayList<Observateur>();

  public void run() {
    while(true){
      this.cal = Calendar.getInstance();
      this.hour =  //Les heures
        this.cal.get(Calendar.HOUR_OF_DAY) + " : "
        +
        (      //Les minutes
          this.cal.get(Calendar.MINUTE) < 10
          ? "0" + this.cal.get(Calendar.MINUTE)
          : this.cal.get(Calendar.MINUTE)
        )
        + " : "
        +
        (      //Les secondes
          (this.cal.get(Calendar.SECOND)< 10)
          ? "0"+this.cal.get(Calendar.SECOND)
          : this.cal.get(Calendar.SECOND)
        );
      //On avertit les observateurs que l'heure a été mise à jour
      this.updateObservateur();

      try {
        Thread.sleep(1000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }

  //Ajoute un observateur à la liste
  public void addObservateur(Observateur obs) {
    this.listObservateur.add(obs);
  }
  //Retire tous les observateurs de la liste
  public void delObservateur() {
    this.listObservateur = new ArrayList<Observateur>();
  }
  //Avertit les observateurs que l'objet observable a changé
  //et invoque la méthode update() de chaque observateur
  public void updateObservateur() {
    for(Observateur obs : this.listObservateur )
      obs.update(this.hour);
  }
}
```

Etape 3 : ***ObserverTest.java***

```
package com.formation.test;
….

 public class ObserverTest  extends JFrame {


        private static final long serialVersionUID = 1L;
        private JLabel label = new JLabel();
        private Horloge horloge;

        public ObserverTest() {
               // On initialise la JFrame
               this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
               this.setLocationRelativeTo(null);
               this.setResizable(false);
               this.setSize(200, 80);

               // On initialise l'horloge
               this.horloge = new Horloge();
               // On place un écouteur sur l'horloge
               this.horloge.addObservateur(new Observateur() {
                      public void update(String hour) {
                             label.setText(hour);
                      }
               });

               // On initialise le JLabel
               Font police = new Font("DS-digital", Font.TYPE1_FONT, 30);
               this.label.setFont(police);
               this.label.setHorizontalAlignment(JLabel.CENTER);
               // On ajoute le JLabel à la JFrame
               this.getContentPane().add(this.label, BorderLayout.CENTER);
               this.setVisible(true);
               this.horloge.run();
        }

        // Méthode main() lançant le programme
        public static void main(String[] args) {
               ObserverTest fen = new ObserverTest();
        }
 }
```

Sortie