TP1		
Module/Semestre	M33/M7	Programmation Evénementielle (Java)
Activité	☑ Création des classes d'événement. ☑ Utiliser les écouteurs d'événement. ☑ Créer des composants graphiques. ☑ Associer les événements à un contrôle.	
Détails sur les objectifs visés par l'activité	Cette activité d'apprentissage doit vous permettre de : ▪ Comprendre la notion d'événement. ▪ Les observateurs d'événement (Listeners). ▪ Créer et gérer les événements. ▪ Utiliser les observateurs et les gestionnaires d'événements ▪ Associer les événements aux composants graphiques. ▪ Rendre les composants réactifs.	

DESCRIPTION DE L'ACTIVITE

Dans ce travail vous aller apprendre à créer et utiliser les événements pour développer les interfaces graphiques.

Comme exemple vous allez Réalisez :

Un programme capable d'afficher une phrase dans la console à chaque fois que l'on clique sur un bouton.

Un programme capable d'afficher la phrase entrée dans un TextField dans la console à chaque fois que l'on tape RETOUR CHARRIOT.

Travail 1 :

Utiliser la plateforme BetBeans.
Créer un nouveau projet Intitulé « EssaiEvenement »
Ajouter à la classe principale intitulé aussi « EssaiEvenement ». le code ci-dessous :

```
package ex_event;
import java.awt.*;
import java.awt.event.*;

public class EssaiEvenement extends Frame {
    public EssaiEvenement(String s) {
        super(s);
        TextField text = new TextField(20);
        GestionEv gestion = new GestionEv();
        text.addActionListener(gestion);
        setSize(200,200);
        setLayout(new FlowLayout());
        add(text);
        setVisible(true);
    }
   public static void main(String[] args) {
       EssaiEvenement e= new EssaiEvenement("EssaiEvenement");
        e.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e) {System.exit(0);}
        });
    }
}
```

Créer une nouvelle classe intitulé « GestionEv » et ajouter le code ci-dessous.

```
import java.awt.event.*;
public class GestionEv implements ActionListener {
    public void actionPerformed(ActionEvent e){
                System.out.println(e.getActionCommand());
        }
}
```

Répondre aux questions suivantes :

1. Que ce qu'elle représente la classe « EssaiEvenement » ?
2. Que ce qu'elle représente la classe « GestionEv » ?
3. A quoi sert-il l'implémentation du ActionListner et ActionPerformed ?
4. Décrire le constructeur de la classe « EssaiEvenement ».
5. Compiler et exécuter le programme. Décrire son fonctionnement.

Travail 1 :

Créer un nouveau projet.
Ajouter à la classe principale Mapp_event le code ci-dessous :

```
import java.awt.Event;
import javax.swing.*;
public class Mapp_event extends JFrame {

   JButton butt1;
   public Mapp_event(String ttl)
   {
      butt1 = new JButton("Click_me");
      butt1.setSize(20, 20);
      this.setTitle(ttl);
      this.add(butt1);
      this.setSize(500,500);
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      this.setLocationRelativeTo(null);
      Gevent ge = new Gevent(this);
      butt1.addActionListener(ge);
      this.pack();
      this.setVisible(true);

   }
   public static void main(String[] args) {
      Mapp_event new_frm = new Mapp_event("event_ex");
   }

}
```

Créer une nouvelle classe Gevent et ajouter le code ci-dessous.

```
package pprj_event;

import java.awt.event.*;
import javax.swing.JFrame;
public class Gevent implements ActionListener {
   JFrame mjf;
   public Gevent(JFrame jf){
      mjf=jf;
   }
   @Override
   public void actionPerformed(ActionEvent e){
      System.out.println("Done! : " + e.getSource());
      mjf.dispose();
   }
}
```

Répondre aux questions suivantes :

1. Que ce qu'elle représente la classe « Mapp_event » ?
2. Que ce qu'elle représente la classe « Gevent » ?
3. A quoi sert-il l'implémentation du ActionListner et ActionPerformed ?
4. Décrire le constructeur de la classe « Mapp_event ».
5. Expliquer le comportement du programme sur l'intervention d'utilisateur.
6. Compiler et exécuter le programme. Décrire son fonctionnement.