Lyssnare

PropertyChangedEvent
Man har en komponent som kan förändra egenskaperna på ett objekt. När egenskaperna förändras så kan lyssnare anmäla intresse att lyssna.

Händelseklass	Händelser	Kan t.ex. genereras av
`ActionEvent`	`actionPerformed`	`AbstractButton` – knapp tryckt `JTextField` – textändring avslutad `JComboBox` – alternativ valt
`ChangeEvent`	`stateChanged`	`JComponent` – komponentens tillstånd ändrat
`AdjustmentEvent`	`adjustmentValue-Changed`	`JScrollbar` – skjutreglaget ändrat
`ItemEvent`	`itemStateChanged`	`JToggleButton`, `Checkbox`, `Choice` – alternativet valt

`ComponentEvent`	`componentHidden`
	`componentMoved`
	`componentResized`
	`componentShown`
		`Component` – komponenten gömd, flyttad, omskalad eller visad

`ContainerEvent`	`componentAdded` `componentRemoved`	`Container` – en komponent har lagts till eller tagits bort
`FocusEvent`	`focusGained` `focusLost`	`Component` – komponenten fick eller förlorade fokus

`WindowEvent`	`windowActivated`
	`windowClosed`
	`windowClosing`
	`windowDeactivated`
	`windowDeiconified`
	`windowIconified`
	`windowOpened`
		`Window` – fönstret öppnas, stängs, skall stängas, blir ikon eller återställs

`KeyEvent`	`keyPressed`
	`keyReleased`
	`keyTyped`
		`Component` – en tangent har tryckts ner eller släppts

`MouseEvent`	`mouseClicked`
	`mouseEntered`
	`mouseExited`
	`mousePressed`
	`mouseReleased`
		`Component` – musknapp nedtryckt eller släppt, musen flyttad in eller ut i komponenten

	`mouseDragged` `mouseMoved`	`Component` – musen dragen eller flyttad
`MouseWheelEvent`	`mouseWheelMoved`	man har vridit på musens hjul
`PropertyChangeEvent`	`propertyChange`	`JComponent` – egenskaper ändrade

Hur man gör:

Ska man implementera en lyssnarklass måste man implementera alla de metoder vilka finns i klassen.

Fuskväg
Man kan, istället för en lyssnare, lägga in en adapter! En adapter innehåller alla metoder, men de är tomma så man behöver bara implementera de man inte vill ha. Men glöm då för allt i världen inte att skriva **@Override** på den specifika metoden du vill ha - annars blir Skansholm arg!

```
// Struktur för lyssnarklass

class MyFocusListener implements FocusListener {
  public void focusGained(FocusEvent e) {
    ...
  }
  public void focusLost(FocusEvent e) {
    ...
  }
}


FocusListener fl = new MyFocusListener();
addFocusListener(fl);
```

Olika sätt att lyssna

Lyssna inne i en klass. Inne i lyssnaren kan man undersöka om en viss komponent överensstämmer med en annan komponent.

```
// Lyssna själv

class C extends C0 implements ActionListener {
  JComponent comp = new ... ;
  // konstruktorer och övriga metoder i C
  C() {
    comp.addActionListener(this); // registrera lyssnaren
    ...
  }
  ...
  public void actionPerformed(ActionEvent e) { // lyssnarmetod
      ...
  }
}
```

Bandit 2.java

```
import java.awt.*;
  import java.awt.event.*;
  import javax.swing.*;
  import java.util.*;

 public class Bandit2 extends JFrame {
    JButton spela = new JButton("Spela");
    JLabel  n1 = new JLabel("",JLabel.CENTER);
    JLabel  n2 = new JLabel("",JLabel.CENTER);
    JLabel  n3 = new JLabel("",JLabel.CENTER);
    JLabel info = new JLabel(" Ingen vinst ",JLabel.CENTER);
    String nummer = "";

    Bandit2() {
      setTitle(getClass().getName());
      Font font = new Font("Times", Font.PLAIN, 36);
      spela.setFont(font); n1.setFont(font); n2.setFont(font); n3.setFont(font); info.setFont(font);
         Knapplyssnare lyss = new Knapplyssnare();

      class Knapplyssnare implements ActionListener {
        // Här hanterar vi händelserna
      public void actionPerformed(ActionEvent e) {
        int t1, t2, t3;
        t1 = (int)(Math.random() * 9);
        t2 = (int)(Math.random() * 9);
        t3 = (int)(Math.random() * 9);
        n1.setText(t1+"");
        n2.setText(t2+"");
        n3.setText(t3+"");
        if ((t1==t2) && (t1==t3)) {
          info.setText("Vinst 50 kr");
        }
        else
          info.setText("Ingen vinst");
      }
      }
```

Här finns ingen lyssnare.

Den är här.

```
// Intern, separat lyssnarklass

class C extends C0 {
  JComponent comp = new ... ;
  C() {
    comp.addActionListener(l); // registrera lyssnaren
    ...
  }
  class MyListener implements ActionListener {    // lyssnarklass
    public void actionPerformed(ActionEvent e) { // lyssnarmetod
      ...
    }
  }
  MyListener l = new MyListener();              // lyssnare
}
```

6 of 8

```
// Anonym lyssnarklass, med variabel

class C extends C0 {
  JComponent comp = new ... ;
  C() {
    comp.addActionListener(l); // registrera lyssnaren
    ...
  }
  ActionListener l = new ActionListener() {       // anonym klass
    public void actionPerformed(ActionEvent e) {
      ...
    }
  };
}
```

6

Bandit3.java

```
     ActionListener lyss = new ActionListener() {
       // Här hanterar vi händelserna
      public void actionPerformed(ActionEvent e) {
        int t1, t2, t3;
        t1 = (int)(Math.random() * 9);
        t2 = (int)(Math.random() * 9);
        t3 = (int)(Math.random() * 9);
        n1.setText(t1+"");
        n2.setText(t2+"");
        n3.setText(t3+"");
        if ((t1==t2) && (t1==t3)) {
          info.setText("Vinst 50 kr");
        }
        else
          info.setText("Ingen vinst");
      }
      };

    public static void main(String[] s) {
      new Bandit3();
    }
  }
```

Bandit4.java

```
      spela.addActionListener(new ActionListener() {
          // Här hanterar vi händelserna
        public void actionPerformed(ActionEvent e) {
          int t1, t2, t3;
          t1 = (int)(Math.random() * 9);
          t2 = (int)(Math.random() * 9);
          t3 = (int)(Math.random() * 9);
          n1.setText(t1+"");
          n2.setText(t2+"");
          n3.setText(t3+"");
          if ((t1==t2) && (t1==t3)) {
            info.setText("Vinst 50 kr");
          }
          else
            info.setText("Ingen vinst");
```

```
// Anonym lyssnarklass, utan variabel

class C extends C0 {
  JComponent comp = new ... ;
  C() {
    comp.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        ...
      }
    });
    ...
  }
}
```

7

I Java 8 har de implementerat ngt de kallar "funktionsgränssnitt" som innebär att man bara är ute efter en specifik funktion och kompilatorn fattar det och skalar bort allt annat.

Bandit5.java

```
      spela.addActionListener(e -> {
          // Här hanterar vi händelserna
        int t1, t2, t3;
        t1 = (int)(Math.random() * 9);
        t2 = (int)(Math.random() * 9);
        t3 = (int)(Math.random() * 9);
        n1.setText(t1+"");
        n2.setText(t2+"");
        n3.setText(t3+"");
        if ((t1==t2) && (t1==t3)) {
          info.setText("Vinst 50 kr");
        }
        else
          info.setText("Ingen vinst");
```

```
// Anonym lyssnarklass, med lambda-uttryck

class C extends C0 {
  JComponent comp = new ... ;
  C() {
    comp.addActionListener( e -> {...} );
    ...
  }
}
```


Vad ska man nu ha skiten till?

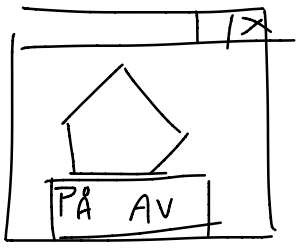
Demo

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class PolyDemo extends JFrame implements ActionListener {
  private JButton på = new JButton("På");
  private JButton av = new JButton("Av");
  private Poly y = new Poly(5, 50);
  public PolyDemo() {    // Konstruktor
    JPanel a = new JPanel();
    add(y, BorderLayout.CENTER);
    add(a, BorderLayout.SOUTH);
    a.add(på); a.add(av);
    på.addActionListener(this);
    av.addActionListener(this);
    setSize(200,180);
    setVisible(true);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == på)
      y.start();
    else
      y.stop();
  }

  public static void main (String[] arg) {
    PolyDemo pd = new PolyDemo();
  }
}
```



PolyClass →

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Poly extends JPanel implements
ActionListener {
   private int n, r;
   private double vinkel;
   private int[] x, y;
   private double dv = 5*2*Math.PI/360;  // 5
grader
   private double vrid = 0;
   private Timer tim = new Timer(100, this);

   public Poly(int antal, int radie) { //
konstruktor
     n = antal; r = radie;
     x = new int[n];
     y = new int[n];
     vinkel = 2*Math.PI/n;
   }

   public void start() {
     tim.start();        // starta timern
   }

   public void stop() {
     tim.stop();        // stoppa timern
   }

   public void actionPerformed(ActionEvent e) {
     // hit kommer man var 100:e ms
     vrid = vrid + dv;
     if (vrid>2*Math.PI)
       vrid -= 2*Math.PI;
     repaint();
   }

   @Override
   public void paintComponent(Graphics g) {
     super.paintComponent(g);  // suddar
bakgrunden
     int x0 = getSize().width/2;
     int y0 = getSize().height/2;
     // räkna ut nya hörnpunkter
     for (int i=0; i<n; i++) {
       double v = i*vinkel - vrid;
       x[i] = x0 + (int)Math.round(r *
Math.cos(v));
       y[i] = y0 - (int)Math.round(r *
Math.sin(v));
     }
     // rita ny bild
     g.fillPolygon(x, y, n);
   }
 }
```

Arv

När en klass ärver en annan klass får du med alla flashiga grejer i superklassen men du får även möjlighet att lägga till egna variabler och så vidare.

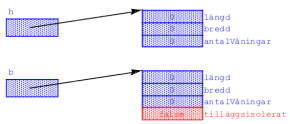
```
public class Hus {
  double längd;
  double bredd;
  int antalVåningar;
  public double yta() {
    return längd * bredd * antalVåningar;
  }
}
```

```
public class Bostadshus extends Hus {
  boolean tilläggsisolerat;
  public void isolera() {
    tilläggsisolerat = true;
  }
}
```

b och h är referensvariabler vilka refererar till de två objekten

Det ena objektet har en instansvariabel mer än det andra.

```
Hus h = new Hus();
Bostadshus b = new Bostadshus();
```

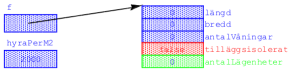


Vi utökar en redan utökad klass och lägger till ytterligare variabler och metoder. I detta exempel är det en metod för att räkna ut hyran för dessa flerfamiljshus.

hyraPerM2 är en klassvariabel och finns bara i ett enda exemplar. Den går att avläsa från Flerfamiljshusklassen men eftersom hpM2 är static så kan vi inte pilla med den efter kompilation.

```
public class Flerfamiljshus extends Bostadshus {
  int antalLägenheter;
  public static final double hyraPerM2 = 2000;
  public double beräknadHyresinkomst() {
    return yta() * hyraPerM2;
  }
}
```

```
Flerfamiljshus f = new Flerfamiljshus();
```



Klassen Object

Object är den enda klassen som saknar superklass. Alla andra klasser är utökningar av denna klass.

```
class C {
  ...
}
```

```
class C extends Object {
  ...
}
```

Terminologi

När h pekar på ett bostadshus så tappar vi informationen om det utökade. Om h pekar på b "försvinner det röda" men det är inte inkorrekt i sak,

Däremot kan b inte peka på h ty b insinuerar att det finns något i h som egentligen inte fanns.

```
Hus h = new Hus();          // h refererar nu till ett Hus
Bostadshus b = new Bostadshus();
Flerfamiljshus f = new Flerfamiljshus();
h = b;                      // h refererar nu till ett Bostadshus
h = f;                      // h refererar nu till ett Flerfamiljshus
h.antalLägenheter = 20;         // FEL!!
h.längd = 30;                   // OK

b = h;                          // FEL!!
b = (Bostadshus) h;             // Korrekt, men farligt
if (h instanceof Bostadshus)
  b = (Bostadshus) h;           // Säkert
```

Ärvs metoder?

Hajemän, men vill du göra om en metod ska du se till att skriva **@Override** annars blir Skansholm - som tidigare nämnt - arg!

Dynamisk bindning

Innebär att anropsmekanismen tar den verkliga typen av objektet och behandlar det.

```
public class Flerfamiljshus extends Bostadshus {
  int antalLägenheter;
  public static final double hyraPerM2 = 2000;
  public double beräknadHyresinkomst() {
    return yta() * hyraPerM2;
  }
  @Override
  public double yta() {
    return längd * bredd * antalVåningar * 0.95;
  }
}
```

Alternativ version:

```
  @Override
  public double yta() {
    return super.yta() * 0.95; // anropa yta() i klassen Hus
  }
```

```
Hus h = new Hus();
Flerfamiljshus f = new Flerfamiljshus();
h.bredd = 10; h.längd=20; h.antalVåningar = 3;
f.bredd = 10; f.längd=20; f.antalVåningar = 3;
System.out.println(h.yta());
System.out.println(f.yta());
```

Självklart:

```
600.0
570.0
```

Men

```
Hus h2;
h2 = f;
System.out.println(h2.yta());  // Vilken metod anropas?
```

Dynamisk bindning ger:

```
570.0
```