Mittuniversitetet
MID SWEDEN UNIVERSITY

Lektion 7 –
GUI: Grafiska användargränssnitt

Java för C++-programmerare

Syfte:	Att förstå uppbyggnaden av och kunna använda klasserna för olika typer av fönsterhantering i paketen javax.Swing och java.awt.Att förstå Javas händelsehantering och kunna använda olika lyssnarklasser.
Att läsa:	Skansholm kapitel 6, 8, 12, 14 och 15 Oracle's tutorial "Creating a GUI with JFC/Swing" http://docs.oracle.com/javase/tutorial/uiswing/index.html

GUI och händelsestyrd programmering

Denna lektion behandlar grafiska komponenter som olika typer av fönster, menyer, knappar, dialogrutor osv. Användning av dessa typer komponenter leder till en annan typ av styrning av ett program än den traditionella modell som finns i ett text/konsol-baserat användargränssnitt.
Med en mus (eller på en pekskärm) kan användaren markera/klicka på många olika ställen och därigenom generera olika typer av *händelser (events)* som programmet måste hantera. I Java gör man det genom att skapa olika typer *lyssnare (listeners)* med metoder som automatiskt anropas för olika specifika händelser under exekveringen. Denna typ av programmering brukar kallas för *händelsestyrd programmering*.

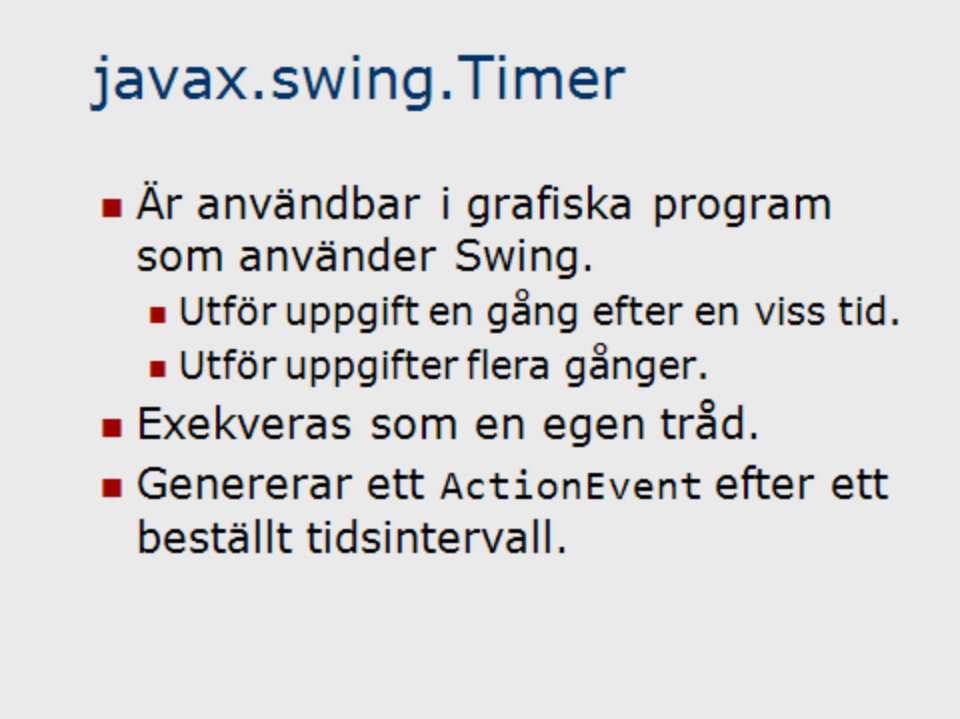
Det är framförallt i två paket man hittar de klasser man behöver för att för att använda grafiska komponenter och händelsestyrd programmering:

java.awt (Abstract Window Toolkit)
javax.swing

Du ska utforska dessa områden på egen hand. Rekommenderade källor är:

- Skansholm kapitel 6, 8, 12, 14 och 15
- Oracle's on-line tutorial
  http://docs.oracle.com/javase/tutorial/uiswing/index.html
- Bifogade presentationer och exempel

För att direkt illustrera några viktiga koncept så tar vi ett inledande exempel.



Skapa en Timer

- I konstruktorn anger vi intervallet mellan varje händelse samt en referens till en händelselyssnare.

```
Timer t = new Timer(1000, this);
```

- Koden genererar en händelse varje sekund med anropande objekt som lyssnare.
- Startas med metoden `start()`.

```
t.start();
```

Metoder

- Några användbara metoder i `Timer`:

```
t.start();              // startar timern t
t.stop();               // stoppar timern t
t.restart();            // startar om timern t
t.setDelay(ms);         // ändrar tidsintervallet
t.setInitialDelay(ms);// tid i ms till första händelsen
t.setRepeats(false);  // bara en händelse ska genereras
t.isRunning();         // är t aktiv?
```

Vi gör en egen grafisk komponent där vi använder klassen Timer för att låta två trådar räkna upp två olika räknare. De aktuella värdena skrivs ut i ett fönster.

Vår första klass är Counter som är en grafisk komponent baserad på (ärvd från) klassen JLabel. På så vis blir en Counter ”en slags JLabel” enligt OOPs grunder. Dessutom implementerar klassen interfacet ActionListener genom att definiera metoden
`public void actionPerformed(ActionEvent).`
Detta är en s.k. ”call-back-metod” som automatiskt anropas av Javas runtime-system när en viss händelse har inträffat. I vårt fall handlar det om att angivet tidsintervall för Timer-objektet har passerat och då anropas alltså actionPerformed, med ett ActionEvent som beskriver händelsen, som argument.

```
import java.awt.*;
import java.awt.event.*;  // ActionListener
import javax.swing.*;  // JLabel

class Counter extends JLabel implements ActionListener {
  private int value, speed;

  public Counter(int value, int speed) {
     this.value = value;
     this.speed = speed;
     setText(Integer.toString(value));
     Timer timer = new Timer(1000/speed, this);
     timer.start();
  }

  public void actionPerformed(ActionEvent ae) {
    setText(Integer.toString(++value));
  }
}
```

Vi har två instansvariabler i klassen för att hålla reda på aktuellt värde och för att specificera i vilken fart uppräkningen ska ske (antal uppräkningar per sekund). I konstruktorn sätter vi vilket initialt värde uppräknaren ska ha samt vilken fart den ska räknas upp med. Därefter skapar vi ett Timer-objekt och initierar det så att en händelse genereras efter (`1000/speed`) millisekunder. Vi registrerar den aktuella klassen som en lyssnare på händelser från Timer-objektet genom att `this`. Med metoden `start()` startar vi Timern.

I metoden `actionPerformed`, som anropas varje gång timern genererar en händelse, ökar vi värdet med 1 varefter vi uppdaterar texten i komponenten `setText` som är ärvd från JLabel.

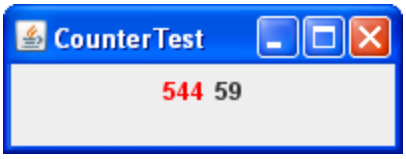
Se exemplet **Counter.java** som följer med lektionen.

```
public class CounterTest extends JFrame {

    public CounterTest() {

        super("CounterTest");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());
        setSize(200, 75);

        Counter counter1 = new Counter(0, 10);
        counter1.setForeground(Color.red);

        Counter counter2 = new Counter(0, 1);

        add(counter1);
        add(counter2);
        setVisible(true)
    }

    public static void main(String[] args) {
         CounterTest test = new CounterTest();
    }
}
```

För att testa vår Counter-komponent skapar vi ett fönster baserat på (ärvt från) JFrame. JFrame är en s.k. ”top-level container” vilket innebär att det är ett huvudfönster vars främsta uppgift är att vara en kontainer för andra grafiska komponenter.

I konstruktorn initierar vi basklassen JFrame genom att anropa `super` med det namn som ska visas i fönstrets övre ram. Sedan skapar vi två objekt av klassen Counter (counter1 och counter2). Vi initierar den första uppräknaren till att börja från värdet 0 och sätter farten till 10 (räkna upp 10 gånger per sekund). Den andra räknaren initierar vi till startvärdet och att värdet ska räknas upp 1 gång per sekund. För att skilja de olika räknarna åt byter vi förgrundsfärg på första uppräknaren (kom ihåg att Counter ärver JLabel).

Med metoden `add` (från JFrame) lägger vi till JLabel-objekten (dvs Counter.objekten) i kontainern CounterTest (som genom arvet nu är en ”sorts JFame”).



När vi startar programmet kommer de båda Counter-objektetn att exekvera parallellt i två separata trådar men skriva ut sitt värde i den gemensamma CounterTest. Se exemplet **CounterTest.java** . Trådar behandlas mer ingående i en kommande lektion.