

Programmeringsteknik och Matlab

Övning 6

Övningsgrupp 2 (Sal Q22/E32)

Johannes Hjorth
hjorth@nada.kth.se
Rum 4538 på plan 5 i D-huset
08 - 790 69 02

Kurshemsida:
<http://www.nada.kth.se/kurser/kth/2D1312>

Övningsanteckningar:
<http://www.nada.kth.se/~hjorth/teaching/prgi05>

Dagens program

- Kan vi bygga vidare på existerande klasser?
Kort introduktion till arv
- Hur hittar vi information om inbyggda klasser?
J-uppgiften är inte alls lika styrd som tidigare labbar
- Hur visar vi grafik på skärmen?
Ett smakprov på grafik

Vad är arv?

Antag att det finns en klass som har *nästan* allt vi behöver. Den gör det vi vill att den ska göra utom i ett par fall.

Ska vi då behöva skriva om allt från början eller kan vi bygga vidare på den redan existerande klassen?

Till exempel så finns det en datumklass, men den kan inte automatiskt berätta hur lång tid det är kvar till jullovet!

Vi skulle vilja kunna skriva:

```
System.out.println(idag.timmarTillJullovet());
```

Kan vi då på något vis göra en egen klass som får ära egenskaper från klassen `GregorianCalendar`?

Vi ärver från `GregorianCalendar`

Genom att skriva `extends` `GregorianCalendar` så ärver vår klass smidigt kalenderegenskaperna och vi kan bygga vidare på dem.

```
import java.util.*;

public class Datum extends GregorianCalendar {

    private GregorianCalendar lov; // ny instansvariabel

    public Datum() {
        super(); // anropa GregorianCalendar's konstruktor

        // Ytterligare ett anrop till konstruktorn
        lov = new GregorianCalendar(2005,12,19,13,00);
    }

    public double timmarTillJullovet() {

        // getTimeInMillis() ger tid i millisekunder sedan 1 januari 1970
        return (lov.getTimeInMillis() - getTimeInMillis())/(1000.0*60*60);
    }
}
```

Vad är skillnaden mellan de två konstruktoranropen?

Med hjälp av `super()` så anropar vi ursprungs-klassens konstruktor så den får en chans att initiera de *ärvda* delarna av vårt objekt.

Vi testar vår nya klass

```
import java.util.*;
import java.text.*;

public class ArvExempel {

    public static void main(String[] parametrar) {
        DecimalFormat df = new DecimalFormat("###");
        Datum idag = new Datum();

        System.out.println("Datorklockan är nu "
            + idag.get(Calendar.HOUR_OF_DAY) + ":"
            + idag.get(Calendar.MINUTE) + ".");

        double tidKvar = idag.timmarTillJullovet();

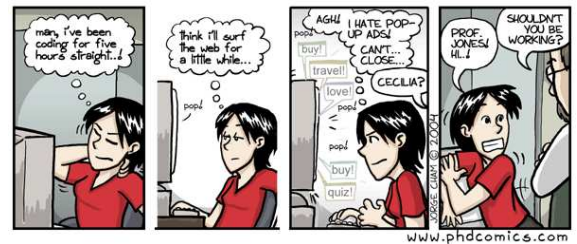
        System.out.println("Det är " + df.format(tidKvar)
            + " timmar kvar till jullovet.");
    }
}
```

Här använder vi oss av `DecimalFormat` för att utskriften inte ska ha för många decimaler.

En detalj kvar...

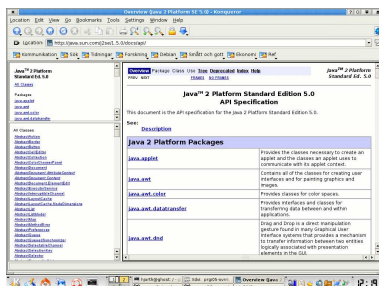
När jag testkörde programmet i söndags blev det ett fel i utskriften, hur ska vi göra för att rätta till det?

```
>javac ArvExempel.java
>java ArvExempel
Datorklockan är nu 12:8.
Det är 2449.86 timmar kvar till jullovet.
```

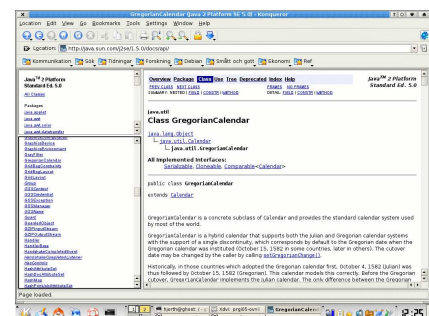
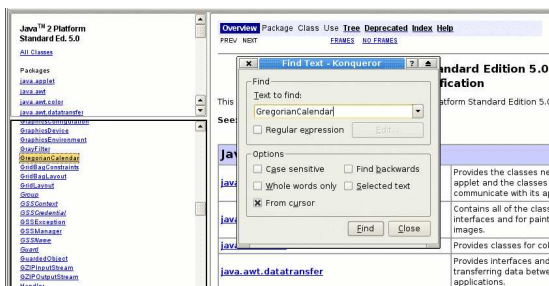


Java API

Om vi behöver kolla upp hur vi använder en viss klass så kan vi göra det på Suns websida.



Enklast är att söka på klassen i nedre vänstra listan



Method Summary	
void	add (int field, int amount) Adds the specified (signed) amount of time to the given calendar field, based on the calendar's rules.
Object	clone () Creates and returns a copy of this object.
protected void	computeFields () Converts the time value (millisecond offset from the Epoch) to calendar field values.
protected void	computeTime () Converts calendar field values to the time value (millisecond offset from the Epoch).
boolean	equals (Object obj) Compares this GregorianCalendar to the specified Object.

Constructor Detail

GregorianCalendar

```
public GregorianCalendar()
```

Constructs a default **GregorianCalendar** using the current time in the default time zone with the default locale.

GregorianCalendar

```
public GregorianCalendar(TimeZone zone)
```

Constructs a **GregorianCalendar** based on the current time in the given time zone with the default locale.

Parameters:
zone - the given time zone.

Hur får vi grafik?

Ett första grafikexempel, HelloWorld.java

```
import java.awt.*;

public class HelloWorld {

    public static void main(String parametrar[]) {
        Frame enFrame = new Frame("Hello World!");

        Label enLabel = new Label("Hej kompis!");
        enFrame.add(enLabel);

        enFrame.setSize(200, 100);
        enFrame.show();
    }
}
```

Vi har här skapat en Frame och sedan placerat en Label på den genom att använda metoden add.

Med setSize bestämmer vi storleken på vår Frame och slutligen så gör vi den synlig med show.



Svårare än så behöver det inte vara.

Fler komponenter - menyer

Förutom text går det även att visa menyer

```
import java.awt.*;

public class Inmatning {

    public static void main(String argv[]) {
        Frame enFrame = new Frame("Menyval");

        Choice humörVal = new Choice();
        humörVal.add("Glad som en sol");
        humörVal.add("Sur som en citron");
        humörVal.add("Pillemarisk som en ekorre");
        enFrame.add(humörVal);

        enFrame.setSize(200, 25);
        enFrame.show();
    }
}
```

Vi bygger ut vårt program genom att använda färdiga komponenter.



LayoutManager

Hittills har vi bara visat en komponent på skärmen åt gången. Nu ska vi använda en LayoutManager för att visa flera komponenter samtidigt.

```
import java.awt.*;

public class LayoutExample {

    public static void main(String argv[]) {
        Frame enFrame = new Frame("Border Layout");
        enFrame.setLayout(new BorderLayout());

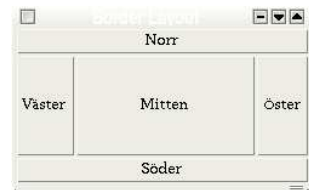
        Button norr = new Button("Norr");
        Button söder = new Button("Söder");
        Button öster = new Button("Öster");
        Button väster = new Button("Väster");
        Button mitten = new Button("Mitten");

        enFrame.add(norr, BorderLayout.NORTH);
        enFrame.add(söder, BorderLayout.SOUTH);
        enFrame.add(öster, BorderLayout.EAST);
        enFrame.add(väster, BorderLayout.WEST);
        enFrame.add(mitten, BorderLayout.CENTER);

        enFrame.setSize(250, 150);
        enFrame.show();
    }
}
```

Vi lägger här till fem stycken knappar för att illustrera hur BorderLayout placerar ut de olika komponenterna på skärmen.

Kör vi koden kommer det att se ut så här:



Med raden

```
enFrame.add(norr, BorderLayout.NORTH);
```

placerar vi ut en knapp i norra fältet, på samma sätt placeras de övriga knapparna ut i de olika fälten.

Trycker vi på knapparna händer ingenting...

Det vill vi ändra på!

Hur ger vi knapparna liv?

Vad vet vi?

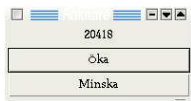
Varje gång en knapp trycks ner sker ett Event. Genom att skapa en metod som kan hantera händelser kan vi få knapparna att fungera.

Hur använder vi det?

Vi låter vår klass implementera `ActionListener` och kopplar den till knapparna. Varje gång en knapp trycks ner körs då metoden `actionPerformed`.

Hur ser det ut i kod?

Se `LevandeKnappar`-klassen på nästa sida.



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

public class LevandeKnappar extends Frame implements ActionListener{

    private Label display;
    private Button öka, minska;
    private int summa = 0;

    LevandeKnappar(String titel) {
        super(titel); // Vi har ärvt av Frame, bör anropa Frame-konstruktorn

        setLayout(new GridLayout(3,1));
        setSize(200,100);

        display = new Label("0", Label.CENTER);
        öka = new Button("Öka");
        minska = new Button("Minska");

        // Koppla ihop knapparna med vår ActionListener
        öka.addActionListener(this);
        minska.addActionListener(this);

        add(display); add(öka); add(minska);
    }

    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == öka) { // Var det en knapp som trycktes ner?
            summa++;
            display.setText("" + summa);
        } else if(e.getSource() == minska) {
            summa--;
            display.setText("" + summa);
        }
    }

    public static void main(String argv[]) {
        LevandeKnappar minRäknare = new LevandeKnappar("Räknare");
        minRäknare.setVisible(true);
    }
}
```

Hur läser vi in text?

Det är ganska vanligt att vi vill fråga användaren om ett namn eller ett tal. Då är `TextField` användbart.

Här kodar vi en grafisk temperaturomvandlare mellan Fahrenheit och Celsius.

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

class TextInlasning extends Frame implements ActionListener {

    private Label titel, fahrText, celText;
    private TextField fahrenheit, celsius;

    TextInlasning() {

        setFont(new Font("Arial",Font.BOLD,20));
        setTitle("Temperaturomvandling");
        setSize(500,130);
        setLayout(new FlowLayout());

        titel = new Label("Omvandling mellan Fahrenheit och Celsius");
        fahrText = new Label("Fahrenheit:");
        celText = new Label("Celsius:");
        fahrenheit = new TextField(4);
        celsius = new TextField(4);

        add(titel); add(fahrText); add(fahrenheit); add(celText); add(celsius);

        fahrenheit.addActionListener(this);
        celsius.addActionListener(this);
    }
}
```

```
private int fahrToCel(int f) { return (f - 32) * 5 / 9; }
private int celToFahr(int c) { return 9 * c / 5 + 32; }

public void actionPerformed(ActionEvent event) {
    if(event.getSource() == fahrenheit) {
        int f = Integer.parseInt(fahrenheit.getText());
        celsius.setText("" + fahrToCel(f));
    } else if(event.getSource() == celsius) {
        int c = Integer.parseInt(celsius.getText());
        fahrenheit.setText("" + celToFahr(c));
    }
}

public static void main(String[] inparam) {
    TextInlasning t = new TextInlasning();
    t.setVisible(true); // Glöm inte att visa fönstrena!

    // Programmet är interaktivt!
}
}
```

Vi implementerar en `ActionListener` för att känna av när användaren har skrivit in en temperatur.



Med metoden `fahrenheit.getText()` läser vi in den text som skrivits in i Fahrenheit-rutan.

Hur visar man en bild?

Antag att vi vill kunna få upp följande på skärmen?



Hur gör vi då?

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

public class VisaBild extends Frame { // Vi ärver från Frame-klassen

    private Image bild;

    VisaBild(String titel, String filnamn) {
        super(titel); // Anropa Frame-klassens superkonstruktor

        Toolkit toolkit = Toolkit.getDefaultToolkit();
        bild = toolkit.getImage(filnamn);

        MediaTracker mediaTracker = new MediaTracker(this);
        mediaTracker.addImage(bild, 0);

        try { // Vänta på att bilden laddas in
            mediaTracker.waitForID(0);
        }
        catch (InterruptedException ie) {
            System.err.println(ie);
            System.exit(1);
        }

        // Detta behövs för att kunna stänga fönstret med musen
        addWindowListener(new WindowAdapter()
        { public void windowClosing(WindowEvent e) { System.exit(0); } });

        setSize(bild.getWidth(null), bild.getHeight(null));
    }

    // Metod som ritas upp bilden på skärmen
    public void paint(Graphics graphics) {
        graphics.drawImage(bild, 0, 0, null);
    }

    public static void main(String argv[]) {
        VisaBild vb = new VisaBild("Party!", "kräftskiva.jpg");
        vb.setVisible(true);
    }
}
```

Hur gör jag för att...

Det som vi har gått igenom idag är bara ett smakprov på vad man kan göra i java. Mer information finns att hitta på nätet.

Java API

<http://java.sun.com/j2se/1.5.0/docs/api/>



JORGE CHAY ©THE STANFORD DAILY