Applikationsprogrammering i Java
7,5 hp

Lektion 1

Personal

- Kursansvarig, Föreläsare och handledare
  - Johan Eliasson (johane@cs.umu.se)
- Föreläsare
  - P-O Östberg (p-o@cs.umu.se)
- Handledare
  - Johan Granberg (johang@cs.umu.se)
  - Tor Sterner-Johansson (tors@cs.umu.se)
  - Daniel Henriksson (danielh@cs.umu.se)

Information

- Websajt
  - http://www.cs.umu.se/kurser/5DV085/HT08
- Nyheter från mig
  - Email
  - Nyhetssidan på sajten
- Handledning
  - IRC - #5dv085 på irc.acc.umu.se

Kursens innehåll

- Mera om Javas bibliotek
- Grafiska gränssnitt
- Reaktiv programmering
- Trådar
- Databaser
- XML
- Design patterns
- etc

Kursens mål

- Kunna förstå mer avancerade delar av Java
- Kunna programmera och förstå GUI
- Kunna programmera och förstå databaser (inte i detalj dock)
- Lite kunskap om XML
- Lite kunskap om design patterns
- Ge exempel från ett annat OO språk

Det verkliga målet

Ni ska ha en uppsättning verktyg som ni kan använda i era fortsatta studier (och även efteråt)

Vad innebär detta

- Tyngdpunkten i kursen ligger i att konstruera program
  - Skriva mycket kod
  - Skriva bra kod

Vad som krävs av dig

Följande krävs av dig:

- Att du är aktiv
- Att du lägger ner tid på kursen
- Att du själv skriver kod
- Att du själv designar program

Examination

- Kursen är uppdelad i 2 moment
- Teorimoment 3 hp
  - Examineras genom 3h tenta på slutet av kursen
  - Betyg U,3,4,5
- Labmoment 4,5 hp
  - Examineras genom ett antal laborationer
  - Betyg U,3,4,5
- Totala betyget på kursen blir viktat resultat på momenten

Inlämningsuppgifter

- Lab1 Interspection och GUI-intro. (5p)
  - Enskilt
- Lab2 XML - RSS-läsare (10p)
  - Enskilt
- Lab3 Distributed data storage using web services(5p)
  - Enskilt
- Lab4 AntiTowerDefence (20p)
  - I par. 3 st redovisningstillfällen
  - 2008-12-18 Milestone. Behöver inte vara fungerande, men ni ska ha kommit en bra bit på vägen (demo av hur det gått hittills/plan för resten av arbetet för handledare)
  - 2008-01-13 Demo av spelet för lärare/handledare och medstudenter
  - 2008-01-13 Inlämning av dokumentation

Krav på inlämningsuppgifterna

- Ska lämnas in i tid (planera din tid !)
- Uppskov beviljas av mig
- Det räcker inte med att programmet fungerar!

Krav på inlämningsuppgifterna

Rapporten ska

- vara klart och tydligt utformat
  (på svenska *eller* engelska ... inte bägge)
- innehålla en beskrivning av idén som lösningen bygger på
- innehålla en beskrivning av design/lösning
- innehålla en översikt av klasserna
- beskrivning av val, begränsningar, problem, alternativa lösningar

Krav på inlämningsuppgifterna

- Kod enl god OO modell
  - Arv där så är lämpligt
  - Interface där så är lämpligt
- Bedöms efter design, räcker *inte* med att programmet funkar
- God standard på koden

Om boken

- Lite gammal
- Täcker inte Java 1.5 - 1.6
- Bra bok - fått positiva kommentar från tidigare år
- Rekommenderas

- Utöver boken kommer ni tex behöva använda API beskrivningen som finns på nätet i rätt stor utsträckning

Bibliotek

- På kursen kommer vi använda en mängd klasser ur javas bibliotek
- Stort, mycket stort
- För stort?
- Portabelt — lite beroende på vad man gör hur portabelt det blir
- Objektorienterat
- Använder sig i stor grad av Design Patterns

Verktyg

- javac
- java
- javadoc
- jar
- etc

Olika programtyper

- Applikationer (vanliga program)
- Applets (program för websidor)
- JavaBeans (komponenter)
- Servlets (server program för websidor)
- JSP (annan variant)
- Enterprise JavaBeans (distribuerad)

Java varianter

- Standard Edition
  - Den "vanliga" varianten
- Enterprise Edition
  - API, beskrivning
- Micro Edition
  - Lite resurser

IDE

- BlueJ www.bluej.org
- Eclipse www.eclipse.org
- jGrasp www.jgrasp.org
- etc

Introspection

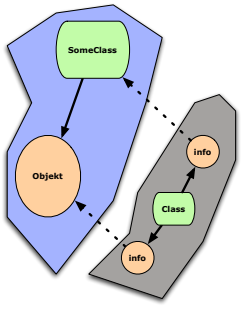
Javas förmåga att undersöka sig själv

Run-Time Type Info

- RTTI
- Kolla vad ett objekt innehåller och vilka egenskaper det har
- instanceof

Class

- Objekt som skapas av run-time systemet (JVM)
- Håller reda på informationen om ett annat objekt eller om en klass



- Class java.lang.Object.getClass()
  - Class<?> classOlle = olle.getClass()
- Class SomeClassName.class
  - Class<Member> classMember = Member.class
- Class java.lang.Class.forName(x)
  - Class<?> classMember = java.lang.Class.forName("xfile.Member")

Metoder i ett Class-objekt (exempel)

- String getName()
- Class getSuperclass()
- boolean isArray()
- boolean isInstance(Object obj)
- boolean isInterface()
- Object newInstance()


```
public class CreateByName{
   public static void main ( String[] args ) {
        for( int i = 0 ; i < args.length ; i++ ) {
            try {
                Class<?> x = Class.forName( args[i] );
                Object y;
                try {
                    if( x.isInterface() ) {
                        System.out.println(args[i] + " är ett interface och inget objekt kan skapas");
                    } else {
                        y = x.newInstance();
                        if( y instanceof java.awt.Component ){
                            System.out.println( args[i] + " är en GUI komponent");
                        } else {
                            System.out.println( args[i] + " är inte en GUI komponent");
                        }
                    }
                }
                catch( InstantiationException e ) {
                    e.printStackTrace();
                }
                catch( IllegalAccessException e ) {
                    e.printStackTrace();
                }
            }
            catch( ClassNotFoundException e ) {
                System.out.println( "Kunde inte hitta " + args[i] );
            }
        }
    }
}
```



```
Class<?> x = Class.forName( args[i] );
Object y;
try {
    if( x.isInterface() ) {
        System.out.println(args[i] + " är ett interface och inget"
                                   + "objekt kan skapas");
    } else {
        y = x.newInstance();
        if( y instanceof java.awt.Component ){
            System.out.println( args[i] + " är en GUI komponent");
        } else {
            System.out.println( args[i] + " är inte en GUI komponent");
        }
    }
}
```


"Casting"

- Casting är en annan sak som görs möjlig tack vare RTTI.
- Säker
- Osäker
- Run-time: ClassCastException

Reflection API

- Ger mer detaljer
- java.lang.reflect.Constructor
- java.lang.reflect.Method
- java.lang.reflect.Field
- java.lang.reflect.Array

java.lang.reflect.Constructor

- getParameterTypes
- newInstance

java.lang.reflect.Method

- getParametersTypes
- getReturnType
- invoke


```
import java.lang.reflect.*;

public class ListStringMethods{
    public static void main( String[] args )
        throws ClassNotFoundException {
        Method[] ma = String.class.getMethods();
        for( int i = 0 ; i < ma.length ; i++ ){
            System.out.println( ma[i] );
        }
    }
}
```

```
> java ListStringMethods
public int java.lang.String.hashCode()
public int java.lang.String.compareTo(java.lang.String)
public int java.lang.String.compareTo(java.lang.Object)
public boolean java.lang.String.equals(java.lang.Object)
public int java.lang.String.length()
```


Konstruktorer

- Initierar ett nytt objekt
- Kan finnas flera olika - minst en

```
public class Example
{
    ...
    public Example( ) { ... }
    public Example( String s ) { ... }
    public Example( int i ) { ... }
    ...
}
```



```
public class Coordinate {
    private int x;
    private int y;

    public Coordinate( Integer x, Integer y ) {
        this.x = x.intValue();
        this.y = y.intValue();
    }

    public String toString() {
        return "(" + x + "," + y + ")";
    }
}
```



```
import java.lang.reflect.*;
public class Demo {
    public static void main(String[] args)
        throws java.lang.ClassNotFoundException,
            java.lang.NoSuchMethodException,
            java.lang.InstantiationException,
            java.lang.IllegalAccessException,
            java.lang.reflect.InvocationTargetException {
        Class<?> c = Class.forName( "Coordinate" );

        Class[] p = new Class[2];

        p[0] = Integer.class;
        p[1] = Integer.class;

        Constructor con = c.getConstructor(p);

        Object[] par = new Object[2];

        par[0] = new Integer(12);
        par[1] = new Integer(34);

        Object o = con.newInstance(par);

        System.out.println( o );
    }
}
```

Finalize

- Finalize är en metod som körs när objektet tas bort, kan användas för att rensa upp efter ett objekt.
- Anropas när objektet tas bort
- **Inte säkert att objektet tas bort!!!**
- **Man ska aldrig anropa finalize själv!!**


```
public class Example {

    public Example() {
        System.out.println("Hello, wonderful world")
    }

    protected void finalize()
    throws Throwable {
        System.out.println("Goodby cruel world");
    }
}
```



```
public class Demo {
    public static void main(String[] args) {
        Example eo = new Example();
        eo = null;
    }
}
```

```
> java Demo
Hello, wonderful world
>
```



```
public class Demo {
    public static void main(String[] args) {
        Example eo = new Example();
        eo = null;
        System.gc();
        System.runFinalization();
    }
}
```

```
> java Demo
Hello, wonderful world
Goodbye cruel world
>
```


GUI-intro

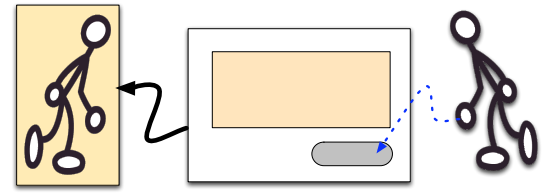
Målet

- Användaren - ge användaren ett sätt att mata in information
- Systemet - göra det möjligt att använda ett GUI för att presentera information

Historia

- Java 1.0 - AWT
- Java 1.1 - AWT
  - java.awt
- Swing - först separat nerladdning
- Swing - sedan en del av standarddistributionen (Java 1.2)
  - javax.swing

Principer - händelsestyrt



Händelsestyrt

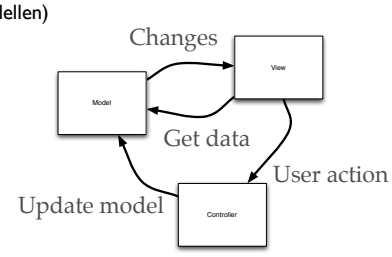
- Princip
  - Programmet "sitter och rullar tummarna"
  - Användaren gör nåt
  - JVM/JFC genererar en eller flera händelser och skickar dessa till ditt program
  - Programmet hoppar upp, gör något snabbt och fortsätter sedan att "rulla tummarna".

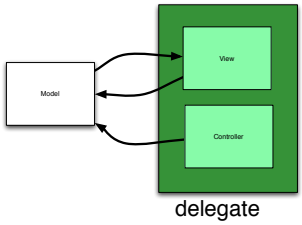
Begrepp

- **Components** - basen för de andra klasserna (knappar, listor, etc)
- **Controls** - knappar, listor, etc
- **Containers** - kan innehålla andra komponenter
- **LayoutManager** - sköter placeringen av olika GUI komponenter

MVC

- Model-View-Controller
  - Logiken (modellen)
  - Utseende
  - Kontrolldel

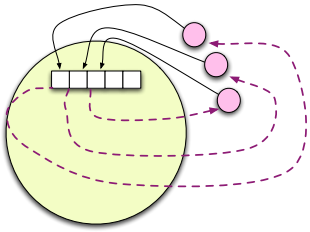




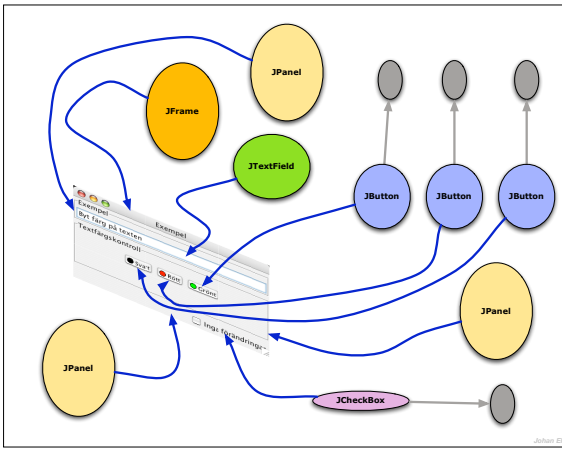
delegate

Hur funkar det?

- addXXXXListener
- XXXXListener interface



Swing



JPanel

```
private JPanel      upperPanel;
private JPanel      middlePanel;
private JPanel      lowerPanel;

public ThreePanels( String title )
{
    super( title );

    Container cp = getContentPane();

    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

    upperPanel = new JPanel();

    cp.add( upperPanel, BorderLayout.NORTH );

    middlePanel = new JPanel( );

    cp.add( middlePanel, BorderLayout.CENTER );

    lowerPanel = new JPanel( );

    cp.add( lowerPanel, BorderLayout.SOUTH );

    pack();
    setVisible(true);
}
```


BorderLayout