Javaprogrammering - Del 1

- Java allmänt
- Objektorientering - polymorfism

Kursupplägg

- föreläsningar
- övningar 1, 2, 3 och 4
- INLÄMNINGSUPPGIFT 2

- tentamen

Hello world

hello_c - Anteckningar

Arkiv Redigera Sök Hjälp

```
public class hello_c
{
  public static void main( String args[] )
  {
    System.out.println( "Hello world!" );
  }
}
```


Kompilera

Kommandotolken

```
C:\eget\java\ht00\folex\hello>javac -verbose hello_c.java
[parsed hello_c.java in 911 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/lang/Object.class) in 50 ms]
[checking class hello_c]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/lang/String.class) in 40 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/io/Serializable.class) in 0 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/lang/Comparable.class) in 0 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/lang/System.class) in 20 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/io/PrintStream.class) in 10 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/io/FilterOutputStream.class) in 0 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/io/OutputStream.class) in 0 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/io/IOException.class) in 0 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/lang/Exception.class) in 0 ms]
[loaded c:\jdk1.2.2\jre\lib\rt.jar(java/lang/Throwable.class) in 50 ms]
[wrote hello_c.class]
[done in 2003 ms]

C:\eget\java\ht00\folex\hello>
```


Kör

Kommandotolken

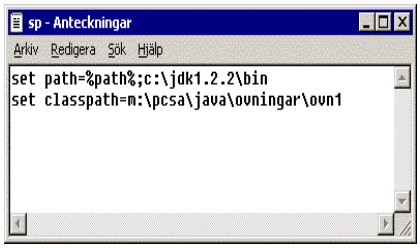
```
C:\eget\java\ht00\folex\hello>java hello_c
Hello world!

C:\eget\java\ht00\folex\hello>
```


Environmentvariabler

- path
  - ska peka ut C:\JDKkatalog\bin
  - JDKkatalog = installationsrot för JDK
- classpath
  - ska minst peka ut lokal katalog
- skriv SP.BAT . . .
  - får ordna paths

SP.BAT



Rektangelprogram

- läs två sidlängder för en rektangel
- presentera area och omkrets
- skriv som DOS-tillämpning

Rektangel-pgm 1

```
import java.io.*;
class rektangel_c
{
  public static void main( String args[] )
  {
    int a = 0, b = 0;
    String buf;

    BufferedReader kbd_reader =
      new BufferedReader(
        new InputStreamReader( System.in ) );
```


Rektangel-pgm 2

```
    try
    {
      System.out.print( "Ge sida a: ");
      buf = kbd_reader.readLine();
      a = Integer.parseInt( buf );

      System.out.print( "Ge sida b: ");
      buf = kbd_reader.readLine();
      b = Integer.parseInt( buf );
```


Rektangel-pgm 3

```
      System.out.println( "Area: " + a * b );
      System.out.println(
        "Omkrets: " + (2*a + 2*b) );
    } // try
    catch ( IOException exc )
    {
      System.out.println( "Kunde inte läsa!" );
    } // catch

  } // main
} // rektangel_c
```


Härnäst

- samma problem, andra lösningar
- rektangelklass
- main-metod med objekt av klassen
- och sedan …
  - rektangelklass med main-metod

Rektangelklass - 1

```
class rektangel_c
{

  public rektangel_c(
    int a,
    int b )
  {
    this.a = a;
    this.b = b;
  } // rektangel_c
```


Rektangelklass - 2

```
public int omkrets()
{
  return 2*this.a + 2*this.b;
} // omkrets


public int area()
{
  return this.a*this.b;
} // area
```


Rektangelklass - 3

```
  private int a = 0, b = 0;

} // class rektangel_c
```


Main med rektangelobjekt - 1

```
import java.io.*;
import rektangel_c;
class main_c
{
  public static void main( String args[] )
  {
    BufferedReader kbd_reader =
      new BufferedReader(
      new InputStreamReader( System.in ) );
    int a, b;
    String buf;
```


Main med rektangelobjekt - 2

```
    try
    {
      System.out.print( "Ge sida a: ");
      buf = kbd_reader.readLine();
      a = Integer.parseInt( buf );

      System.out.print( "Ge sida b: ");
      buf = kbd_reader.readLine();
      b = Integer.parseInt( buf );
```


Main med rektangelobjekt - 3

```
      rektangel_c rektangel =
        new rektangel_c( a, b );

      System.out.println(
        "Area: " + rektangel.area() );
      System.out.println(
        "Omkrets: " + rektangel.omkrets() );
    } // try
```

Main med rektangelobjekt - 4

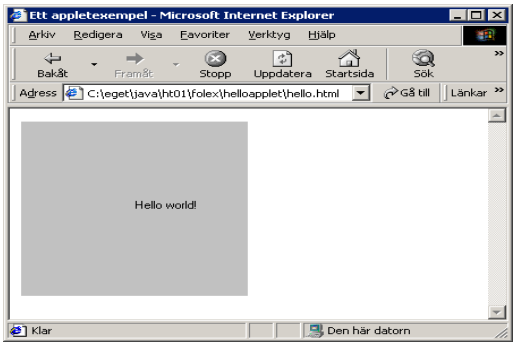
```
    catch ( IOException exc )
    {
      System.out.println( "Kunde inte läsa!" );
    } // catch

  } // main
} // main_c
```


Main i rektangelklassen

- sist i rektangel_c
- samma utseende som ovan
- köra programmet
  - java rektangel_c
  - mainmetoden i klassen anropas
- möjligt: 1 mainfunktion per klass
  - för utprovning

Hello World i Browser



Hello World - Applet

```
import java.applet.*;
import java.awt.*;
public class hello_c extends Applet
{
  public void paint( Graphics gr )
  {
    gr.drawString( "Hello world!", 100, 100 );
  } // paint
} // class hello_c
```


Hello World - HTML-fil

```
<html>
<head><TITLE>
Ett appletexempel
</TITLE></head>
<body>
<APPLET
  CODE=hello_c.class
  WIDTH=200 HEIGHT=200>
</APPLET>
</body>
</html>
```


Grundtankar om Java

- ett programspråk
- ett klassbibliotek
- en virtuell maskin
  - d.v.s. ett datorprogram
  - bytekodstolkare

Virtuell maskin

- tolkar java byte-kod
- översätter till OS-anrop
- bytekoden
  - minimal för effektiv nättransport
- koden alltid portabel (nja, nej ...)
  - OS-specifik VM "det enda som behövs”
  - olika fel beroende på OS

Utvecklingsmiljöer

- Java Developer's Kit (JDK)
  - Javasoft, GRATIS
  - kommandoradsbaserat
- Sun Java Workshop
- Borland JBuilder
- IBM Visual Age
- Symantec Visual Cafe
- Visual J++, m.fl.

Javafiler

- källkod i klassnamn.java
  - eller annat . . . undvik
- bytekod i klassnamn.class
- klassbibliotek
  - vanligen i ZIP-filer
  - även JAR - Java Archive

Programtyper

- Applikationer
  - körs m.h.a. Java VM
- Applets
  - körs m.h.a. webbläsare
  - inbyggd Java VM
  - eller JDK appletviewer

Runtimemiljö

- applikationer
  - Java Runtime Kit
  - ingår i JDK
- applets
  - HotJava, Netscape, Mozilla eller IE m.fl.
- CLASSPATH
  - pekar ut klasskataloger

100% Pure Java

- Write once, run everywhere
  - (skriv en gång, hoppas på det bästa)
- Utnyttja standardklasser
- UNDVIK native-metoder
  - metoder beroende av OS eller andra språk

Språkets grunder

- liknar C++
  - ofta väldigt mycket
  - ibland litegrann
  - INTE enkelt . . .
- repetera OOP-kursen . . .
  - där finns enkla övningar

Kommentarer

- // gäller resten av raden
- /* Kan gälla
- över flera rader */

Viktiga kommentarer

- ovanför klass
  - författare, datum, historik m.m.
- ovanför funktionshuvuden
  - gränsytan
    - vad ska komma in ?
    - vad kommer ut ?

JAVADOC

- dokumentationsgenerator
- ger dok på HTML-format
- kräver att kommentarer ser ut som följer …

JAVADOC-kommentarer

```
/**
Ordnar uppkoppling mot databasen
@param username Användarnamn i DB
@param password Passord i DB
@return
  Kastar bil_db_exception vid fel.
*/
public void connect(
  String username,
  String password )
throws bil_db_exception;
```


Kodblock

- som C++
- kapsla in i måsvingar {}
- sats avslutas med ;
- ; behövs ej efter }

Datatyper

- klass (struct med metoder)
- Primitiva typer
  - byte, short int, long för heltal
  - char för tecken
  - float, double för heltal
  - boolean för logiska värden
  - wrapperklasser finns
- String är en klass, ändå primitiv

Primitiva typer - storlek

- `boolean  1 bit    false eller true`
- `char    16 bits   Unicode character`
- `byte     8 bits   -128 .. 127`
- `short   16 bits   -32768 .. 32767`
- `int     32 bits   -2147483648 .. 2147483647`
- `long    64 bits   -9223372036854775808 .. 9223372036854775807`
- `float   32 bits   3.40282347E+38 .. 1.40239846E-45`
- `double  64 bits   1.79769313486231570E+308 .. 4.94065645841246544E-324`

Variabeldeklarationer

- kan göras var som helst inom {}
- new på objekt
  - inom så snävt scope som möjligt
  - så sällan som möjligt
  - (blås upp alla vid start …)
- int antal ELLER int antal = 5

Vektorer

- INTE som C++
- kan lagra primitiva typer eller objekt
- man får (oftast) gör new

Vektorer

- `int[] intarray = new int[10];`
- `int intarray[] = new int[10]`
- `String[] fruits = { 'Äpplen', 'Päron', 'Apelsiner' };`

- `int intarray[10];`
  - `ger KOMPILERINGSFEL`

Tilldelning och test

- som C++
- tilldelning =
- test ==

Räkneoperationer

- som C++
- +, -, *, /
- ++, --, +=, /=
  - de finns alla med
- <<, >> m.fl. bitoperatorer
  - INTE för in-/utmatning

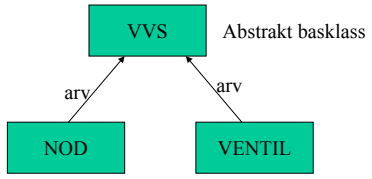
Kontrollflöden

- som C++
- if, while, do .. while, for
- switch, break, continue
- jodå, det är likadant

Skillnad då ?

- hanteringen av objekt
- inga pekare (alltid referenser = pekare)
- endast värdeöverförda parametrar
  - call-by-value
- undantagshantering
- standardbibliotek
- klasser och filer

Objektorientering - Kylsim



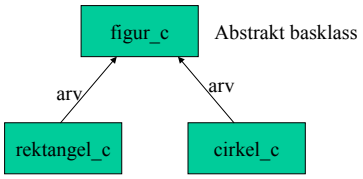
Filer C++

- VVS.H
  - gränssnitt
- VVS.CPP
  - implementation
- NOD.H/NOD.CPP
- VENTIL.H/VENTIL.CPP
- KYLSIM.CPP
  - mainfunktion

Filer Java

- VVS.JAVA
  - gränssnitt OCH implementation
- NOD.JAVA
- VENTIL.JAVA
- KYLSIM.JAVA
  - mainmetod

Nästa exempel - Polymorfism



figur_c kod - 1

```
public abstract class figur_c
{
  public figur_c(
    int x,
    int y )
  {
    this.x_center = x;
    this.y_center = y;
  } // figur_c
```


figur_c kod - 2

```
  public abstract int max_x();
  public abstract int max_y();
  public abstract int area();
  public abstract int omkrets();

  protected int x_center, y_center;
} // class figur_c
```


cirkel_c kod - 1

```
public class cirkel_c extends figur_c
{
  public cirkel_c(
    int x,
    int y,
    int radie )
  {
    super( x, y );
    this.radie = radie;
  } // cirkel_c
```


cirkel_c kod - 2

```
  public int max_x()
  {
    return super.x_center + this.radie;
  } // max_x

  public int max_y()
  {
    return super.y_center + this.radie;
  } // max_y
```


cirkel_c kod - 3

```
  public int area()
  {
    return (int)(
      Math.PI * this.radie * this.radie );
  } // area

  public int omkrets()
  {
    return (int)( 2*Math.PI * this.radie );
  } // omkrets
```

cirkel_c kod - 4

```
  public int diameter()
  {
    return this.radie + this.radie;
  } // diameter

  private int radie;
} // class cirkel_c
```


rektangel_c kod - 1

```
public class rektangel_c extends figur_c
{
  public rektangel_c(
    int x,
    int y,
    int sida,
    int hojd )
  {
    super( x, y );
    this.sida = sida;
    this.hojd = hojd;
  } // rektangel_c
```


rektangel_c kod - 2

```
  public int max_x()
  {
    return super.x_center + this.sida / 2;
  } // max_x

  public int max_y()
  {
    return super.x_center + this.hojd / 2;
  } // max_y
```


rektangel_c kod - 3

```
  public int area()
  {
    return this.sida * this.hojd;
  } // area

  public int omkrets()
  {
    return 2*this.sida + 2*this.hojd;
  } // omkrets

  private int sida, hojd;
} // class rektangel_c
```


figur_c - Polymorfism

- provprogram
  - skapar vektor med 4 figurer
  - loopar igenom, anropar instansmetoder
  - låter Java VM avgöra vad som ska anropas
  - special: instanceof

figmain_c - 1

```
public class figmain_c
{
  public static void main( String args[] )
  {
    figur_c[] figurer = new figur_c[4];

    figurer[0] = new rektangel_c(
      10,   // x
      10,   // y
      10,   // sida
      10 ); // hojd
```

figmain_c - 2

```
figurer[1] = new cirkel_c(
  10,   // x
  10,   // y
  10 ); // radie

figurer[2] = new rektangel_c(
  20,   // x
  20,   // y
  20,   // sida
  20 ); // hojd
```


figmain_c - 3

```
figurer[3] = new cirkel_c(
  20,   // x
  20,   // y
  20 ); // radie
```


figmain_c - 4

```
for ( int index = 0;
      index < figurer.length; index++ )
{
  if ( figurer[index] instanceof cirkel_c )
  {
    System.out.println( "Cirkel:" );
    System.out.println(
      "Diameter: " +
   ((cirkel_c)figurer[index]).diameter() );
  }
  else
  {
    System.out.println( "Rektangel:" );
  } // if
```


figmain_c - 5

```
    System.out.println(
      "Högsta x: " +
      figurer[index].max_x() );
    System.out.println(
      "Högsta y: " +
      figurer[index].max_y() );
    System.out.println(
      "Area: " + figurer[index].area() );
    System.out.println(
      "Omkrets: " +
      figurer[index].omkrets() );
    System.out.println( "***********" );
  } // for
 } // main
} // class figmain_c
```


Vad är en Javaklass ?

- samling av data (medlemsvariabler)
- samling medlemsfunktioner
  - opererar på data
- nja, det där var C++ termer

Vad är en Javaklass (terminologi)?

- samling av instansvariabler
- samling instansmetoder
  - opererar på instansvariabler

Vad är ett objekt ?

- en instans av en klass
- t.ex.
  - cirkel_c cirkel; // får värdet null
  - cirkel = new cirkel_c(10, 10, 10);
    - nu har vi en instans av cirkel_c
    - cirkel pekar dit
- delete FINNS INTE!
- skräpsamling

Syntax

```
<modifier> class <name>
[extends <baseclassname]
[implements <interfacename>]
{
  . . .
  <variable definitions>
  . . .
  <function definitions>
  . . .
}
```


Anrop av instansmetod

```
cirkel_c cirkel = new cirkel_c( 10, 10, 10 );
System.out.println( "Area: " + cirkel.area());
```

- anrop av area()-metoden i cirkel_c
- vad är *this ??*

Peta på data i ett objekt

- gör radie public i cirkel_c
  - public int radie;
  - kan petas på utifrån

```
cirkel_c cirkel = new cirkel_c( 10, 10, 10 );
cirkel.radie = 27;  // som en struct i C
```


Konstruktor

- metod med samma namn som klassen
- **!! OBS !! INGEN RETURTYP**
- varje klass har konstruktor
- default:
  - en utan parametrar
  - gör ingenting

Destruktor

- "glöm det"
  - nästan
  - skräpsamling går ju av sig självt
- men det är klart
  - finalize-metoden

Metod - överlagring

- funktioner med samma namn
  - olika typer på parametrar
  - olika antal parametrar
  - olika returtyper

T.ex. i cirkel_c:

```
public void rita( int skala );
public void rita( int x, int y );
```


Flera konstruktorer

- överlagring även här

T.ex. i figur_c:

```
public figur_c( int x, int y );
public figur_c();
```