Javaprogrammering - Del 2

- Instans- och klassvariabler (static)
- Skräpsamling
- Exceptions
- Arv, klasser och interface, Paket
- Inledande om Applet - roten ur tal

Instans- kontra klassvariabler

- instansvariabel
  - allokeras ny för varje instans
  - INTE static
- klassvariabel
  - allokeras vid programstart
  - ETT ex oavsett antal instanser
  - class-wide-variable
  - SKA VARA static

Instans- kontra klassmetoder

- instansmetod
  - har en osynlig parameter this
    - this refererar struct med instansvariablerna
  - kan peta på instansvariabler
  - kan peta på klassvariabler
- klassmetod
  - har ingen osynlig this-parameter
  - kan INTE peta på instansvariabler
  - kan peta på klassvariabler

Skriva C-program i Java/C++

- gör allting static
- rekommenderas inte
  - kanske ibland ... säkerhetskritiska system

Komplexa tal - program

- en klass complex_c
- klassvariabel antal_komplexa_tal
  - räknar antalet instanser
- privata instansvariabler re och im
- en enkel instansmetod för utskrift
- sist och slutligen en klassmetod main
  - main-metoden

complex_c - 1

```
public class complex_c
{
  public static int antal_komplexa_tal = 0;

  public complex_c()
  {
    this( 0, 0 );
  } // complex_c
```

complex_c - 2

```
public complex_c(
  float re,
  float im )
{
  this.re = re;
  this.im = im;
  complex_c.antal_komplexa_tal++;
} // complex_c
```


complex_c - 3

```
public void skriv_tal()
{
  System.out.print( this.re + " + " +
    this.im + "i" );
} // skriv_tal


private float re, im;
```

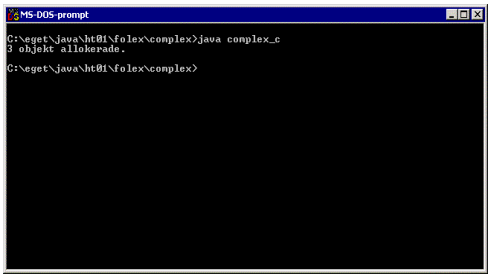

complex_c - 4

```
public static void main( String args[] )
 {
   complex_c cvektor[] =
     new complex_c[3];

   for ( int index = 0; index < 3; index++ )
   {
     cvektor[index] = new complex_c();
   } // while
   System.out.println(
     complex_c.antal_komplexa_tal +
       " objekt allokerade." );
 } // main

} // class complex_c
```

Körexempel - complex_c



main-funktion med problem

- instansräknare
  - räknas upp
  - vet inte när den räknas ner
- om det ändå fanns en destruktor . . .

complex_c - finalize-metod

```
protected void finalize()
  throws Throwable
{
  super.finalize();
  complex_c.antal_komplexa_tal--;
} // finalize
```

Ny mainmetod - 1

```
public static void main( String args[] )
 {
   complex_c cvektor[] =
     new complex_c[3];

   for ( int index = 0; index < 3; index++ )
   {
     cvektor[index] = new complex_c();
   } // while
```

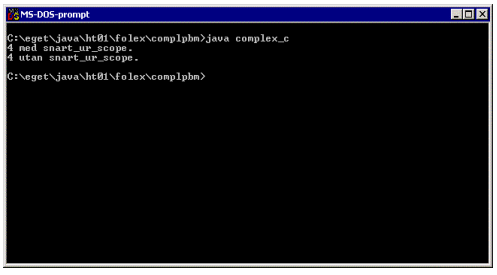

Ny mainfunktion - 2

```
   {
     complex_c snart_ur_scope = new complex_c();
     System.out.println(
       complex_c.antal_komplexa_tal +
       " med snart_ur_scope." );
   }

   System.out.println(
     complex_c.antal_komplexa_tal +
     " utan snart_ur_scope." );
 } // main

} // class complex_c
```


Körexempel



Skräpsamling igen

- snart_ur_scope = null
  - kan provas ovan
  - hjälper inte
  - FÖRSKRÄCKLIGT i C/C++ m.fl.
    - destroy, free e.d. ska användas
  - helt OK i Java
- vill att referensräknare räknas ner

Mer om arv

- kan bara ärva en klass i Java
- multipla arv stöds inte
- flera interface går bra
  - implementeras

Slutgiltiga klasser

- public final complex_c
  - klassen kan inte ärvas
- java.lang.System är en sådan klass
- låser funktionalitet på viss nivå
- inga svårfunna beroenden till subklasser

Superklass

- alla klasser har en superklass
- inget extends
  - då ärver klassen från Object
- Object
  - wait, notify för trådar
  - equals, clone
  - toString m.fl.
  - (jämför void * i C/C++)

Skydda data och metoder - 1

- public
  - alla kan komma åt dem
- protected
  - subklasser kommer åt
  - klasser i samma paket kommer åt

Skydda data och metoder - 2

- paketåtkomst
  - kommer åt inne i paketet
- private
  - bara klassen själv

Tips

- gör oftast instansvariabler privata
  - ha så få instansvariabler som möjligt
- gör interna hjälpmetoder privata
  - då de inte ska anropas utifrån
- public för variabler
  - kanske inte alltid så farligt
  - jämför struct i C

Exceptions

- utnyttjas flitigt i Java
- goto
  - usch och fy
  - finns inte i Java
- throw exception
  - goto nearest exception handler
  - kan poppa anropsstacken

Egen exception - 1

- lagra ett tal
- loopa runt och läs ett nytt tal
  - addera udda tal
  - dividera med jämna tal
- fortsätt vid division med 0
  - egen exceptionhantering
- spåra ur kontrollerat vid fel inmatning

Egen exception - 2

```
// Att kastas vid division med 0
public class div_0_exc
  extends Exception
{
}
```


Egen exception - 3

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

class num_c
{
```


Egen exception - 4

```
  public num_c( int tal )
  {
    this.num = tal;
  } // num_c

  public void addera( int tal )
  {
    this.num = this.num + tal;
  } // addera
```

Egen exception - 5

```
public void dividera( int tal )
  throws div_0_exc
{
  if ( tal == 0 )
  {
    throw new div_0_exc();
  ////////////////////////////
  }
  this.num = this.num / tal;
} // dividera
```


Egen exception - 6

```
public int varde()
{
  return this.num;
}

private int num = 0;
```


Egen exception - 7

```
public static void main( String args[] )
{
  BufferedReader kbd_reader =
    new BufferedReader(
      new InputStreamReader( System.in ) );
  int tal = 1;
  String buf;
  num_c num = new num_c( 14 );
  System.out.println(
    "Startar med: " + num.varde() );
  System.out.println( "Adderar udda tal." );
  System.out.println(
    "Dividerar jämna tal." );
```

Egen exception - 8

```
try
{
  while ( tal != 99 )
  {
    System.out.print(
      "Ge ett tal (99 avslutar): " );
    buf = kbd_reader.readLine();
    tal = Integer.parseInt( buf );
```

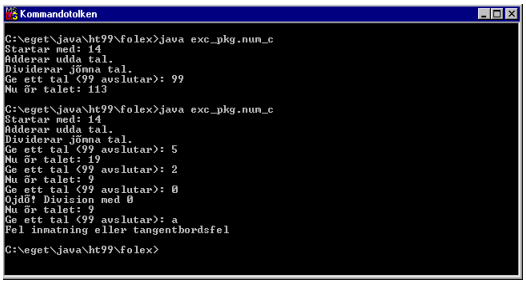

Egen exception - 9

```
    if ( tal % 2 == 1 )
    {
      num.addera( tal );
    }
    else
    try
    {
      num.dividera( tal );
    }
    catch ( div_0_exc exc )
    {
      System.out.println(
        "Ojdå! Division med 0" );
    }
```


Egen exception - 10

```
      System.out.println( "Nu är talet: " +
        num.varde() );
    }
  } // try
  catch ( Exception exc )
  {
    System.out.println(
      "Fel inmatning / tangentbordsfel" );
  } // catch
 } // main
} // class num_c
```

Körexempel



Interface

- klass kan implementera
  - man vet då klassens gränssnitt
- klar fördel vid integration av subsystem
- liknar en abstrakt klass
  - men implementerar ingenting
- nackdel
  - man vet inte vad man får ...

Integralprogram

- funktions-interface
  - specar hur mattefunktioner ska skrivas
- en integrallösare
  - löser integraler
  - vet att funktioner ser ut som interfacet

funktion_i

```
public interface funktion_i
{
  public float f( float x );

  public float derivatan( float x );

  public float anti_derivatan( float x );

}
```


integral_c

```
public class integral_c
{
  public float integralen(
    funktion_i fkn,
    float a,
    float b )
  {
    return fkn.anti_derivatan( b ) -
      fkn.anti_derivatan( a );
  }
}
```


sin_c

```
public class sin_c implements funktion_i
{
  public float f( float x )
  {
    return (float)Math.sin(x);
  }
  public float derivatan( float x )
  {
    return (float)Math.cos(x);
  }
  public float anti_derivatan( float x )
  {
    return -(float)Math.cos(x);
  }
} // sin_c
```

ploynom_c

```
public class polynom_c implements funktion_i
{
  public float f( float x )
  {
    return x*x - x + 3.0f;
  }
  public float derivatan( float x )
  {
    return 2.0f*x - 1.0f;
  }
  public float anti_derivatan( float x )
  {
    return x*x*x/3.0f - x*x/2.0f + 3.0f*x;
  }
} // polynom_c
```


integral_test_c - main - 1

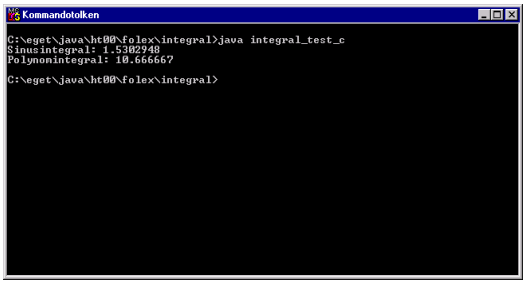
```
import java.io.*;
public class integral_test_c
{
  public static void main( String args[] )
  {

    sin_c sinfunktion = new sin_c();
    polynom_c polynom = new polynom_c();
    integral_c losare = new integral_c();
```


integral_test_c - mainfkn - 2

```
    System.out.println( "Sinusintegral: " +
      losare.integralen(
        sinfunktion, 1.0f, 3.0f ) );
    System.out.println( "Polynomintegral: " +
      losare.integralen(
        polynom, 1.0f, 3.0f ) );
  } // main
} // integral_test_c
```

Körex - integral_test_c



Paket i Java

- specificerar katalogstruktur
- grupperar klasser, objekt, variabler m.m.
- t.ex.
  - java.lang - för javaspråket
  - java.io - in-/utmatning
  - java.awt - grafiska gränssnitt

Paketering av integral

- skapa katalog
  - c:\eget\java\ht01\folex\integral_pkg
- lägg all källkod i katalogen
  - package integral_pkg; överst
- (ta bort public class överallt
  - ==> endast paketåtkomst)
- speca paket vid körning
  - java integral_pkg.integral_test_c

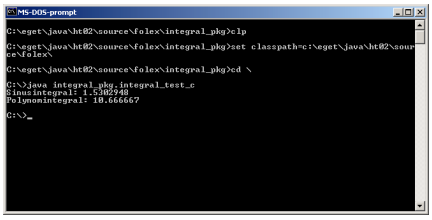
integral_pkg.polynom_c

```
package integral_pkg;

class polynom_c implements funktion_i
{
  public float f( float x )
  {
    return x*x - x + 3.0f;
  }
  . . . som ovan . . .
} // polynom_c
```


Kompilering / körning

1. CLASSPATH ska peka ut katalog OVANFÖR.
2. Kan köra varsomhelst ifrån ...



Applets

- fyrkantig area i en WEB-sida
- man kan rita grafik där
- subklass till Container
  - kan därför innehålla komponenter

Grundläggande

- getAppletContext
  - för kommunik med omvärlden
- hämtning av bilder
  - finns i Appletklassen
- kommunik endast med källservern
  - i normala fall

Paketet java.applet

- Applet - basklass för Applets
- AppletContext - omvkommunik
- AudioClip - för ljud via browsern
- AppletStub - lågnivåkoppling
  - applet - browser

Arvshierarki

» Object
– Component
• Container
– Panel
• Applet

En Applets liv - 1

- appletkod laddas
- instansiering av Applet-objekt
  - init anropas
  - låt BLI konstruktorn
- appleten ska köras
  - start anropas

En Applets liv - 2

- exekvering avslutas
  - stop anropas
- browsern avslutas
  - destroy anropas

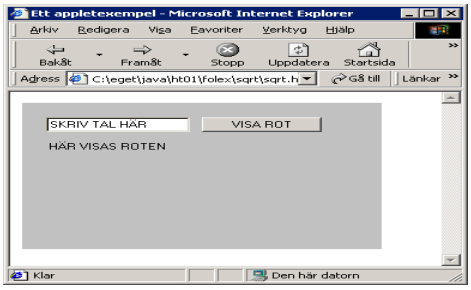
En Applets liv - 3

- applet ska ritas om
  - paint anropas
- jämför windows
  - WM_PAINT-hantering

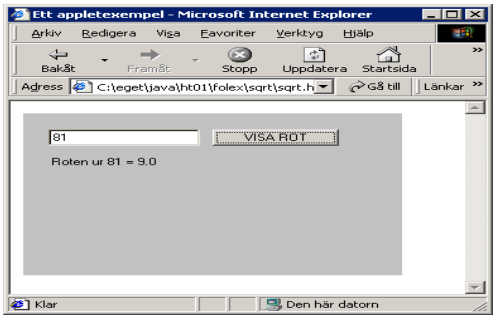
Roten ur

- ett editfält
  - där användaren kan skriva
- en knapp
  - för att beräkna roten ur med felhantering
- en label
  - för att presentera resultat

Körexempel - Roten ur Start



Körexempel - Roten ur Ett tal



sqrt_c - 1

```
import java.awt.Button;
import java.awt.TextField;
import java.awt.Label;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.applet.Applet;

public class sqrt_c
  extends Applet
{
```


sqrt_c - 2

```
  // Inre klass för att lyssna på knapp
  private class sqrt_btn_listener_c
    implements ActionListener
  {
    public void actionPerformed(
      ActionEvent aev )
    {
      handle_sqrt_btn();
    }
  }
```


sqrt_c - 3

```
// Hantera klick på roten-ur-knapp
private void handle_sqrt_btn()
{
  String numbuf = this.num_edt.getText();
  boolean okay = false;
  float num = 0;
```

sqrt_c - 4

```
try
{
  num = new Float( numbuf ).floatValue();
  // !!! Float.parseFloat( numbuf );
  // !!! gick inte i IE !!!
  num = (float)Math.sqrt( num );

  okay = !Float.isNaN( num );
}
catch ( Exception exc )
{
}
```


sqrt_c - 5

```
if ( okay )
{
  this.sqrt_lbl.setText(
    "Roten ur " +
    numbuf +
    " = " +
    Float.toString( num ) );
}
else
{
  this.sqrt_lbl.setText(
    "Det måste vara ett positivt tal." );
}
} // handle_sqrt_btn
```


sqrt_c - 6

```
public void init()
{
  super.init();
  super.setLayout( null );

  this.num_edt = new TextField(
    "SKRIV TAL HÄR" );
  super.add( this.num_edt );
  this.num_edt.setBounds( 20, 20, 120, 20 );

  this.sqrt_lbl = new Label(
    "HÄR VISAS ROTEN" );
  super.add( this.sqrt_lbl );
  this.sqrt_lbl.setBounds( 20, 50, 200, 20 );
```

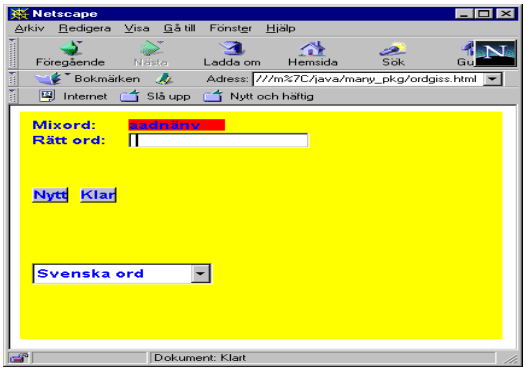
sqrt_c - 7

```
  Button sqrt_btn;
  sqrt_btn_listener_c sqrt_btn_listener =
    new sqrt_btn_listener_c();
  sqrt_btn = new Button( "VISA ROT" );
  sqrt_btn.addActionListener(
    sqrt_btn_listener );
  super.add( sqrt_btn );
  sqrt_btn.setBounds( 150, 20, 100, 20 );
} // init
```


sqrt_c - 8

```
  private Label sqrt_lbl;
  private TextField num_edt;
} // sqrt_c
```


Körexempel - Annan Applet



orgiss_c - källkod

- späckad med kommentarer
- för javadoc
  - kräver speciell kommentarsstil
  - se N:\JAVA\FOLEX\ORDGISS för fullständigt exempel
- plattformsoberoende ?
  - NEJDÅ
  - prov med alla existerande browsers behövs