Javaprogrammering - Del 3

- Trådar
- Databaskoppling, Exempel för Inl2
- Java och nätverk

Trådar

- multitasking
  - tidsdelning mellan flera exekeverande program
- multithreading
  - tidsdelning mellan delar av samma exekverande program
- tidsdelning
  - hanteras av operativsystemet

Trådexempel

- trådklass som presenterar textsträng
- presentation förskjuts åt höger
- två prestrådar körs parallellt
- den ena var 100:e ms
- den andra var 300:e ms

Trådexempel - körning

pres_c - 1

```
public class pres_c extends Thread
{

  public pres_c(
    String to_display,
    int sleep_time )
  {
    this.buf = to_display;
    this.step = 0;
    this.last = 80 - to_display.length();
    this.sleep_time = sleep_time;
  } // pres_c
```


pres_c - 2

```
  public void run()
  {
    while ( pres_c.keep_running )
    {
      try
      {
        Thread.sleep( this.sleep_time );
      }
      catch ( InterruptedException exc )
      {
      }
```

pres_c - 3

```
    for ( int index = 0;
          index < this.step; index++ )
    {
      System.out.print( " " );
    }

    // Skriv strängen
    System.out.println( this.buf );

    this.step =
     ( this.step + 1 ) % this.last;
  }
} // run
```


pres_c - 4

```
// Sträng att presentera
// ---------------------
private String buf;


// Variabler för att hålla koll på steg
// och sovtid mellan presentationer
// -----------------------------------
private int step, last, sleep_time;
public static boolean keep_running;
```


pres_c - 5

```
 public static void main( String args[] )
 {
   // Skapa trådar så att a-tråden körs oftare
   pres_c a_thread = new pres_c(
     "A-sträng LÅNGLÅNG", // to_display
     100 );               // sleep_time
   pres_c b_thread = new pres_c(
     "B-sträng", // to_display
     300 );      // sleep_time

   pres_c.keep_running = true;
   a_thread.start();
   b_thread.start();
 } // main
} // class pres_c
```

Javastöd

- som del av språket
  - nyckelordet synchronized
  - vid "samtidig" dataaccess
- klassen Thread
- interface Runnable
- java VM hanterar mot OS

Trådar - status

- skapad
  - har ännu ej börjat exekvera
- körbar
  - tråden skeduleras
- suspenderad
  - tillfälligt avstannad
- död
  - run-metoden har avslutats/stop anropad

synchronized - 1

- då kod inte får avbrytas
- typiskt vid samtidig access av data
- här
  - tråd write_thread lagrar i buffert
  - tråd read_thread hämtar ur buffert
  - lagring får INTE avbrytas

synchronized - 2

```
class buffer_c
{
  public buffer_c(
    int size )
  {
    // Skapa buffert med plats för
    // size stycken element
  } // buffer_c
```


synchronized - 3

```
  public synchronized void put( String str )
  {
    // Behandla str under viss tid
    // Lagra i buffert
  } // put

  public String last_item()
  {
    // Hämta data ur buffert
    // returnera till anroparen
  } // get
} // class buffer_c
```


Runnable-interface

- om man vill ha applet som tråd
- m.a.o. applet ska skeduleras
  - för anrop ofta . . .
- skilj på
  - en applet som implementerar Runnable
  - en applet som kör flera trådar

Skedulering av trådar

- sätt olika prioritet
- a_thread.setPriority(Thread.MAX_PRIORITY)
- b_thread.setPriority(Thread.MIN_PRIORITY)
- c_thread.setPriority(Thread.NORMAL_PRIORITY)
- a_thread körs oftast . . .

JDBC

- Java Database Connectivity
- gränssnitt mot databaser
- API
  - Application Programmer's Interface
- JDBC API via
  - import java.sql.*
  - klasser och metoder

JDBC - konstruktion

- JDBC Driver Manager
  - del av Javas bibliotek
- JDBC Driver
  - görs av DB-leverantör

JDBC - grundelement

- java.sql.DriverManager
  - laddar rätt drivrutiner mot DB
- java.sql.Connection
  - en anslutning mot DB
- java.sql.Statement
  - för att utföra SQL-kommandon
- java.sql.ResultSet
  - resultat av SQL-kommando

JDBC - finesser

- java.sql.CallableStatement
  - anrop av lagrade procedurer
- java.sql.PreparedStatement
  - vid samma SQL-sats flera gånger
- java.sql.DatabaseMetaData
  - info om databasen
- java.sql.ResultSetMetaData
  - info om kolumner i ett ResultSet

Exempelprogram/Inl 2 - 1

- användare JAVAINL/JAVAINL
- tabell BIL
  - REGNR varchar
  - MÄRKE varchar
  - ÅR int
  - NYPRIS int

Exempelprogram/Inl 2 - 2

- tabell BIL_STAT
  - MÄRKE varchar
  - ÅR int
  - ANTAL int
  - SUMMA int

Exempelprogram/Inl 2 - 3

- lagrad procedur

```
procedure SELECT_1(
  IN_REGNR in char,
  UT_MÄRKE out varchar,
  UT_ÅR out int,
  UT_NYPRIS out int,
  ALLT_OK out int );
```


Exempelprogram/Inl 2 - 4

- skapar koppling mot DB
  - DB måste vara igång
- listar kolumner ur BIL_STAT
- listar alla datarader i BIL_STAT
- anropar SELECT_1
  - presenterar resultatet

Exempelprogram/Inl 2 - 5

```
import java.sql.*;
public class bil_c
{

  public static void main( String args[] )
  {
    String url =
"jdbc:interbase://loopback/c:/temp/bil.gdb:";
```


Exempelprogram/Inl 2 - 6

```
try
{
  // Sök klass och länka in ...
  Class.forName(
    "interbase.interclient.Driver" );

  // Skapa connection
  Connection db_conn =
    DriverManager.getConnection(
      url, "java", "javainl" );
```


Exempelprogram/Inl 2 - 7

```
  // Hämta data ur DB
  Statement statement =
    db_conn.createStatement();
  ResultSet rs = statement.executeQuery(
    "select * from BIL_STAT" );

  // Skriv ut kolumnnamnen
  ResultSetMetaData rsmd =
    rs.getMetaData();
  int kolantal = rsmd.getColumnCount();
```

Exempelprogram/Inl 2 - 8

```
for ( int index = 1;
      index <= kolantal; index++ )
{
  if ( index > 1 )
  {
    System.out.print( ", " );
  } // if
  System.out.print(
    rsmd.getColumnLabel( index ) );
} // for
System.out.println();
```


Exempelprogram/Inl 2 - 9

```
// Skriv ut datarader
while ( rs.next() )
{
  String marke = rs.getString( 1 );
  int ar = rs.getInt( 2 );
  int antal = rs.getInt( 3 );
  int summa = rs.getInt( 4 );

  System.out.println(
    marke + ", " + ar + ", " +
    antal + ", " + summa );
} // while
```


Exempelprogram/Inl 2 - 10

```
// Anropa en lagrad procedur
CallableStatement cs =
  db_conn.prepareCall(
    "{call SELECT_1( ?, ?, ?, ?, ? )}" );

String regnr = "BBC323";
cs.setString( 1, regnr );
cs.registerOutParameter( 2,
  java.sql.Types.VARCHAR );
cs.registerOutParameter( 3,
  java.sql.Types.INTEGER );
cs.registerOutParameter( 4,
  java.sql.Types.INTEGER );
cs.registerOutParameter( 5,
   java.sql.Types.INTEGER );
cs.execute();
```

Exempelprogram/Inl 2 - 11

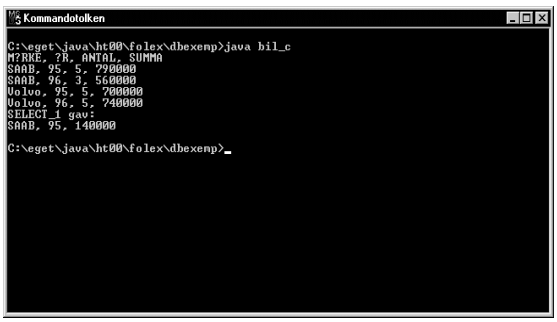
```
if ( cs.getInt( 5 ) > 0 )
{
  // Felflagga > 0, skriv ut data
  String marke = cs.getString( 2 );
  int ar = cs.getInt( 3 );
  int nypris = cs.getInt( 4 );

  System.out.println( "SELECT_1 gav:" );
  System.out.println( marke + ", " +
    ar + ", " + nypris );
} // if
```


Exempelprogram/Inl 2 - 12

```
    // Stäng DB-kopplingar
    statement.close();
    cs.close();
    db_conn.close();
  }
  catch ( Exception exc )
  {
    System.out.println( "FEL PÅ DATAN" );
  }
 } // main
} // class bil_c
```


Körexempel



Java och nätverk

- imponerande stöd
- socketkommunikation
- internet
  - läs webbsidor
  - visa webbsidor

Nätverk

- OSI-modellen
  - 7 nivåer
- TCP/IP
  - enklare, 4 nivåer:
    - Applikationslager
    - Transportlager
    - Nätverkslager
    - Länklager

Protokollexempel

- Applikation
  - ftp, gopher, http
- Transport
  - TCP, UDP
- Nätverk
  - IP
- Länk
  - drivrutiner mot hw

Lågnivåprotokoll

- IP
  - på "paketnivå"
- TCP, UDP
  - logisk förbindelsenivå
  - stöd för applikationsnivån

Högnivåprotokoll

- FTP
  - File Transmission Protocol
  - filer mellan FTP-server och -klient
- HTTP
  - HyperText Transmission Protocol
  - web-sidor mellan web-server och -klient
- TELNET
  - för terminalemulering

TCP/IP - Portar 1

- mjukvaruportar
- server går på en port
  - "Porten är öppen"
  - process som svarar på tilltal via porten
- identifieras av portnummer (16 bitar)
  - max 65536 portar
  - kan ha ännu fler klienter

TCP/IP - Portar 2

- kan ha textsträngsnamn
  - se c:\winnt\system32\drivers\etc\services
- portnumren 0 .. 1024
  - well known ports
  - utnyttja INTE för egna servrar
  - 7 - ECHO, 21- FTP
  - 23 TELNET, 80 - HTTP

Nätverkning med Java

- java.net
  - där finns "allt man behöver"
- t.ex klasserna
  - URL, URLConnection - för stdprotokoll
  - Socket, ServerSocket - TCP-kommunik
  - DatagramSocket - UDP-kommunik

Högnivåprotokollstöd

- stöd via URL-klassen för
  - HTTP
  - FTP
  - Gopher
  - SMTP
  - m.fl.
- protokollval enkelt …

URL - högnivåprotokoll

- man anger
  - protokoll
  - källadress
  - eventuellt portnummer
- getInputStream
  - då kan man läsa data från källan
- getOutputStream
  - då kan man skriva data till källan

Läsning av URL - 1

- skapa URL mot http://www.yahoo.com
- ordna koppling mot källan
- skaffa läsanordning
- läs källan rad för rad
- lista på terminalen

Läsning av URL - 2

```
import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

class url_c
{
```

Läsning av URL - 3

```
public static void main( String args[] )
{
  try
  {
    URL yahoo_url = new URL(
      "http://www.yahoo.com" );
    URLConnection yahoo_conn =
      yahoo_url.openConnection();

    BufferedReader reader = new
      BufferedReader(
        new InputStreamReader(
          yahoo_conn.getInputStream() ) );
```


Läsning av URL - 4

```
    String buf = reader.readLine();
    while ( buf != null )
    {
      System.out.println( buf );
      buf = reader.readLine();
    } // while

    reader.close();
  } // try
```

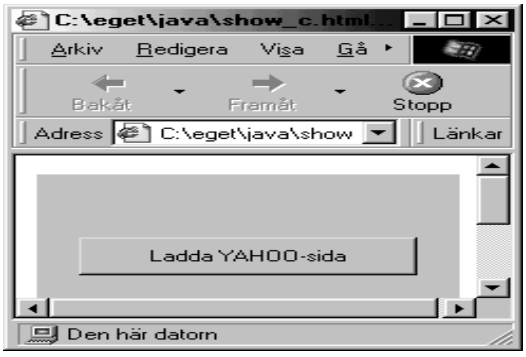

Läsning av URL - 5

```
  catch ( java.net.MalformedURLException exc )
  {
    System.out.println( "Fel på WEB-adressen" );
  } // java.net.MalformedURLException
  catch ( java.io.IOException exc )
  {
    System.out.println(
      "Fel vid getInputStream" );
  } // IOException
 } // main
} // url_c
```

Visning av WEB-sida

- Applet
- 1 enda knapp
  - Ladda YAHOO-sida
- kod för att visa en annan WEB-sida

Körexempel



show_c - 1

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

public class show_c
  extends Applet
  implements ActionListener
{
```

show_c - 2

```
public void actionPerformed(
  ActionEvent aev )
{
  handle_show_btn();
} // actionPerformed
```


show_c - 3

```
  public void handle_show_btn()
  {
    String yahoo_page = "http://www.yahoo.com";
    URL yahoo_url = null;
    try
    {
      yahoo_url = new URL( yahoo_page );
    } // try
    catch ( MalformedURLException exc )
    {
    } // catch
    if ( yahoo_url != null )
    {
      super.getAppletContext().showDocument(
        yahoo_url );
    } // if
  } // handle_show_btn
```


show_c - 4

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

    Button show_btn = new Button(
      "Ladda YAHOO-sida" );
    super.add( show_btn );
    show_btn.setBounds( 20, 50, 160, 30 );
    super.show_btn.addActionListener( this );
  } // init
} // show_c
```

Utökad URL-hantering

- URLStreamHandler
- URLStreamHandlerFactory
- URLConnection

Socket

- tillhandahålls av OS
- för TCP-kommunik
- döljer krångel
  - speciellt via Javas gränssnitt
- ServerSocket
  - öppnar en port i servern
- Socket
  - kan ansluta till server

Socketpgm

- server
  - lyssnar
  - accepterar en klient
  - svarar HEJ HEJ! på HEJ!
  - svarar NOK på annat
- Klient
  - kopplar upp
  - skickar sträng
  - läser svar

Serversidan

- hanteras av ServerSocket-klassen
- öppnar en viss server-port
- inväntar en klient
  - accept
- svarar på tilltal

Mera ServerSocket

- porten bara för att lyssna
  - inte för kommunik
- flera samtidiga klienter
  - kör trådar . . .

HEJ! HEJ! - Serversidan

- Försöker lyssna på port 2747
- accepterar 1 klient
- svarar på tilltal
  - HEJ! HEJ! På HEJ!
  - NOK på annat

HEJ! HEJ! - serv_c - 1

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.IOException;

public class serv_c
{
```


HEJ! HEJ! - serv_c - 2

```
  public static void close_server(
    ServerSocket server )
  {
    try
    {
      server.close();
    }
    catch ( Exception exc )
    {
    }
  } // close_server
```


HEJ! HEJ! - serv_c - 3

```
  public static void close_client(
    Socket client )
  {
    try
    {
      client.close();
    }
    catch ( Exception exc )
    {
    }
  } // close_client
```

HEJ! HEJ! - serv_c - 4

```
public static void close_reader(
  BufferedReader reader )
{
  try
  {
    reader.close();
  }
  catch ( Exception exc )
  {
  }
} // close_reader
```


HEJ! HEJ! - serv_c - 5

```
public static void main( String args[] )
{
  ServerSocket server = null;
  try
  {
    server = new ServerSocket( 2747 );
  } // try
  catch ( IOException exc )
  {
    System.out.println(
      "Kunde inte lyssna!" );
    System.exit( 1 );
  ////////////////////
  } // catch
```


HEJ! HEJ! - serv_c - 6

```
  Socket client = null;
  try
  {
    System.out.println(
      "Lyssnar på port 2747" );
    client = server.accept();
  } // try
```

HEJ! HEJ! - serv_c - 7

```
catch ( IOException exc )
{
  System.out.println(
    "Fel vid avlyssning!" );
  close_server( server );
  System.exit( 1 );
////////////////////
} // catch
System.out.println(
  "OK. Klient ansluten" );
```


HEJ! HEJ! - serv_c - 8

```
PrintWriter writer = null;
BufferedReader reader = null;
try
{
  writer = new PrintWriter(
    client.getOutputStream(), true );
  reader = new BufferedReader(
    new InputStreamReader(
      client.getInputStream() ) );
```


HEJ! HEJ! - serv_c - 9

```
  String buf = reader.readLine();
  while ( buf != null )
  {
    if ( buf.equals("HEJ!") )
      writer.println("HEJ! HEJ!");
    else
      writer.println(”NOK”);
    buf = reader.readLine();
  } // while
} // try
```

HEJ! HEJ! - serv_c - 10

```
catch ( Exception exc )
{
  exc.printStackTrace();
} // catch
finally
{
  if ( writer != null )
  {
    writer.close();
    close_reader( reader );
  } // if
```


HEJ! HEJ! - serv_c - 11

```
    close_server( server );
    close_client( client );
  } // finally
 } // main
} // serv_c
```


finally - block

- bra grej
- avslut på try .. catch
- körs oavsett det gick fel
  - eller inte
- för upprensningskod
  - avslutningsmeddelande
  - m.m. …

HEJ! Klientsidan

- tar servernamn som parameter
  - eller IP-adress
- försöker koppla upp
- läser från tangentbordet
  - skickar sträng till server
  - skriver ut svaret

HEJ! client_c - 1

```
import java.net.Socket;
import java.net.UnknownHostException;
import java.io.InputStreamReader;
import java.io.BufferedReader;
import java.io.PrintWriter;
import java.io.IOException;

public class client_c
{
```


HEJ! client_c - 2

```
  public static void close_client(
    Socket client )
  {
    try
    {
      client.close();
    }
    catch ( Exception exc )
    {
    }
  } // close_client
```

HEJ! client_c - 3

```
public static void close_reader(
  BufferedReader reader )
{
  try
  {
    reader.close();
  }
  catch ( Exception exc )
  {
  }
} // close_reader
```


HEJ! client_c - 4

```
public static void main( String args[] )
{
  if ( args.length != 1 )
  {
    System.out.println(
      "Ange IP-adress eller maskinnamn" );
    System.exit( 1 );
  ////////////////////
  }
```


HEJ! client_c - 5

```
  Socket client = null;
  PrintWriter writer = null;
  BufferedReader reader = null;
  BufferedReader kbd_reader = null;
```

HEJ! client_c - 6

```
try
{
  System.out.println( "Koplar upp ..." );
  client = new Socket( args[0], 2747 );
  writer = new PrintWriter(
    client.getOutputStream(), true );
  reader = new BufferedReader(
    new InputStreamReader(
      client.getInputStream() ) );
  kbd_reader = new BufferedReader(
    new InputStreamReader(
      System.in ) );
} // try
```


HEJ! client_c - 7

```
catch ( UnknownHostException exc )
{
  System.out.println(
    "Kan inte koppla upp mot " + args[0] );
  System.exit( 1 );
//////////////////////
} // catch
```


HEJ! client_c - 8

```
catch ( IOException exc )
{
  System.out.println(
    "IO-fel mot " + args[0] );
  close_client( client );
  if ( writer != null )
  {
    writer.close();
    close_reader( reader );
    close_reader( kbd_reader );
  } // if
  System.exit( 1 );
//////////////////////
} // catch
```

HEJ! client_c - 9

```
System.out.println(
 "OK. Klientsidan ansluten" );
try
{
  System.out.print( "Skriv HEJ!" );
  String buf = kbd_reader.readLine();
  while ( buf.length() > null )
  {
    writer.println( buf );
    String from_server = reader.readLine();
    System.out.println( from_server );
```


HEJ! client_c - 10

```
    System.out.print( "Skriv HEJ!" );
    buf = kbd_reader.readLine();
  } // while
} // try
catch ( Exception exc )
{
  exc.printStackTrace();
} // catch
```


HEJ! client_c - 11

```
  finally
  {
    writer.close();
    close_reader( reader );
    close_reader( kbd_reader );
    close_client( client );
  } // finally
 } // main
} // client_c
```

Datagram - UDP

- UDP - User Datagram Protocol
- fristående meddelande (ej punkt-till-punkt)
- kanske inte når fram ...
- ingen garanterad ordning på paketen
- används av NFS, nätverksspel m.m.

UDP - Javastöd

- DatagramSocket
  - för sändning och mottagning
- DatagramPacket
  - datapaket
- MulticastSocket
  - för sändning till flera