Java-Sprache

Threads

- ❑ Thread-Objekte repräsentieren Aktivitätsträger
- ❑ Synchronisationskonzepte:
  - Monitore
  - Conditions
- ❑ Aktivitäts-Gruppe (ThreadGroup):
  - Kann Threads und weitere Gruppen enthalten
  - Jeder Thread kann nur andere Threads in seiner Gruppe oder Sub-Gruppen beeinflussen

Java-Sprache

Threads

- ❑ Threads werden erzeugt mit Objekten, die zu "Runnable" kompatibel sind
- ❑ An dem entspr. Objekt wird die Methode "run" asynchron ausgeführt

```
class MyClass implements Runnable {
      public void run()  {
            System.out.println("Hello\n");
      }
}

....
MyClass o1 = new MyClass();   // Objekt erzeugen
Thread t1 = new Thread(o1);   // Thread erzeugen, der an o1 loslaufen
                              // soll
t1.start();                   // Thread starten

Thread t2 = new Thread(o1);   // weiteren Thread erzeugen, der an o1
                              // loslaufen soll
t2.start();                   // jetzt laufen 2 zusätzliche Threads...
```


Java-Sprache

Threads

- ❑ Threads können mit Namen versehen werden

```
t1=new Thread(o1,"hello");
System.out.println(t1.toString());

// Ausgabe: Thread[hello,5,main]
// (Name,Priorität,Thread-Gruppe)
```

- ❑ Das System terminiert, wenn alle Threads terminiert sind
- ❑ Daemon-Threads: Threads, die bei der Terminierung nicht berücksichtigt werden

Java-Sprache

Koordinierung

- ❑ Objekte können als Monitore wirken
- ❑ Bei Deklaration von Methoden: "synchronized" angeben
- ❑ Von einem Objekt kann immer nur eine einzige synchronized-Methode gleichzeitig laufen

```
class Bankkonto {
      int value;
      public synchronized void AddAmmount(int v) {
            value=value+v;
      }
      public synchronized void RemoveAmmount(int v) {
            value=value-v;
      }
}
...
Bankkonto b=....
b.AddAmmount(100);
```

Java-Sprache

- ❏ Objekt kann auch außerhalb des Objektes als Monitor verwendet werden

```
Bankkonto b=....
synchronized(b) {
      if (b.GetAmmount()<100)
            b.AddAmmount(100);
}
```

- ❏ Zusätzlich zur Koordinierung: Conditions
- ❏ Innerhalb eines Monitors kann "wait" aufgerufen werden

  ⇒ Monitor wird freigegeben und Thread legt sich schlafen.

- ❏ Jemand anders kann innerhalb des Monitors "notify" (/"notifyAll") aufrufen

  ⇒ Einer (/alle) der blockierten Threads werden aufgeweckt

Java-Sprache

- ❏ Beispiel: Implementation eines Semaphors

```
public class Semaphor {
      int value=1;
      public synchronized void P() throws InterruptedException {
            while (value == 0) wait();
            value--;
      }
      public synchronized void V() {
            ++value;
            notify();
      }
}
```


Java-Sprache

- ❏ In Java sind Methoden per default virtuell
- ❏ D.h. es wird immer die Methodeninstanz von der "echten" Klasse genommen.
- ❏ Man kann Methoden durch "final" als nicht-virtuell deklarieren
  - Dann ist die Methode nicht reimplementierbar
  - Effizienz
- ❏ Abstrakte Methoden / Abstrakte Klassen: ähnlich wie in C++

```
class Test {
      final void Hallo() {...}
}

abstract class Test2 {
      abstract void MyAbstractMethod();
      void TestIt() {...}
}
```


Java-Bibliothek

- ❏ java.lang: Sprach-Unterstützung (Thread, String,...)
- ❏ java.util: Hilfsklassen, allg. Datenstrukturen: Vector, Stack
- ❏ java.io: io-Klassen (Dateien, Streams,...)
- ❏ java.net: Sockets, URL-Unterstützung
- ❏ java.applet, java.awt: Windows-Toolkit

Java-Bibliothek

java.lang

- ❏ Klasse Object: Absolute Oberklasse aller Klassen
- ❏ Bietet Grundfunktionalität:
  - Object clone(); // 1:1 Kopie des Objektes
    - muß das Interface "Cloneable" implementieren
  - boolean equals(Object o) // Objektidentität / Gleichheit feststellen
    - Kann man reimplementieren, z.B.: String
  - String toString(); // Stringrepräsentation des Objektes
    - Liefert defaultmäßig den Klassennamen und die Objekt-Id
    - Sollte man reimplementieren

Java-Bibliothek

java.lang

- ❏ Klasse Class: Meta-Klasse
- ❏ Eine Class-Instanz repräsentiert eine Klasse oder einen Typ (Interface)
- ❏ z.B. geeignet, um neue Instanzen zu generieren.

```
Test t = new Test();
...
Class aClass = t.getClass();
System.out.println("Class of t is:"+aClass);

Object o = aClass.newInstance();    // Jetzt haben wir ein neues Test
                                    // Objekt
Test t2 = (Test) o;
```


Java-Bibliothek

java.lang

- ❏ Klasse String: speichert Zeichenketten
  - Strings sind konstante Objekte
  - Operator "+"

```
String s="Hello";
char x=s.charAt(1);                  // "e"
String s2=s.substring(1,4);          // "ell"

char[] a=s.toCharArray();            // Wenn man Einzelzeichen ändern
for (int n=0; n<a.length; n++) {     // möchte: als char[]
      if (a[n]==’o’)
            a[n]=’q’;
}

s=new String(a);
System.out.println(s);
```


Java-Bibliothek

java.lang

- ❏ Klasse Math: kapselt die mathematischen Standardfunktionen:
  - double d=Math.sin(0.5);
- ❏ Klassen Integer, Double,...
  - kapseln jeweils einen Standardtyp
  - z.B. nötig um einen "int" in eine generische Liste zu speichern
  - generische Listen nehmen "Object"

    ⇒ kompatibel zu allen Objekten

    ⇒ inkompatibel zu Standardtypen

```
Integer i=new Integer(23);              // 23 als Integer-Objekt
Object o=i;                             // kompatibel zu Object
int realInt=((Integer)o).intValue();    // wieder als "int"
```

Java-Bibliothek

❑ Klasse Vector: generisches Feld

❑ Beispiel:

```
Vector t=new Vector();

t.addElement(new String("Hello"));  // String wird automatisch
t.addElement(new String("Tescht")); // auf "Object" konvertiert
t.addElement(new String("Hi"));

for (Enumeration e = t.elements() ; e.hasMoreElements() ;) {
      String s=(String) e.nextElement();
      System.out.println(s);
}
```

❑ Weitere Methoden zum Suchen, Eintragen,...

Java-Bibliothek

❑ Grundkonzept: Streams für Ein- und Ausgabe

❑ Streams mit Gundfunktionalität

❑ Darauf basierend Streams mit erweiterter Funktionalität
- Puffernde Streams
- Streams für Daten Ein / Ausgabe (ASCII)
- Streams für Datei Ein / Ausgabe (Binär)
- Tokenizer Stream zum parsen von Dateien
- Filter-Streams

❑ Hintereinanderhängen von Streams zum Kombinieren von Eigenschaften

❑ Beispiel: System.out

Java-Bibliothek

❑ Beispiel: Ausgabe in eine Datei

```
import java.io.*;
public class Test {
      public static void main (String argv[]) throws IOException {

            FileOutputStream f = new FileOutputStream ("/tmp/test");
                  // FileOutputStream kann nur Bytearrays ausgeben

            BufferedOutputStream b = new BufferedOutputStream (f);
                  // Daraus gepufferten Stream machen

            PrintStream p = new PrintStream (b);
                  // Aus dem BufferedOutputStream komfortableren
                  // Stream machen...

            p.println("Hallo");
            p.println(12+13);
                  // der kann z.B. println mit allen Standardtypen

            p.close();
      }
}
```


Java-Bibliothek

❑ Dateibehandlung mit Klasse "File"

❑ Eine Instanz von File repräsentiert ein Verzeichnis oder eine Datei
- Man kann Dateien löschen, Verzeichnisse anlegen etc.

❑ Beispiel: Verzeichnis lesen

```
File f = new File(".");       // aktuelles Verzeichnis (".")

String[] s = f.list();        // Directory einlesen

for (int n = 0; n < s.length; n++) {
      System.out.println(s[n]);    // alles ausgeben
}
```

Java-Bibliothek
java.net

❑ Klassen für Netzwerk: z.B. Sockets

❑ Besondere Unterstützung von URLs

```
import java.io.*;
import java.net.*;

class Hello {
      public static void main (String argv[]) throws IOException {

            URL u=new URL(
      "http://www4.informatik.uni-erlangen.de/~riechman/test.txt"
            );    // URL definieren

            InputStream f = u.openStream();// URL als Stream aufmachen

            DataInputStream i = new DataInputStream(f);

            String s;
            while ((s = i.readLine()) != null) {System.out.println(s);}
      }
}
```


Java im World Wide Web
Einführung

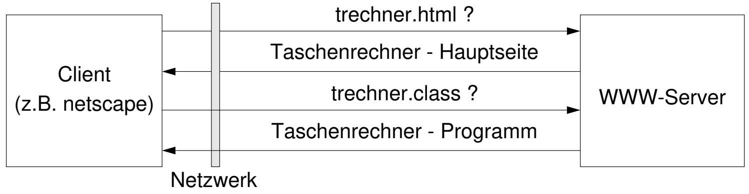
❑ Prinzip vom WWW:
- Der WWW-Server bietet Seiten (Bilder, Text, etc) an
- Der WWW-Klient (z.B. netscape) ruft Seiten ab und zeigt diese an
- Forms: Formulare, die auf dem Klienten ausgefüllt werden

❑ Sämtliche "Intelligenz" liegt beim Server

❑ Java im WWW:
- Applikationen (Applets) können auf den Klienten geladen werden
- Die Applikation (Applet) wird dort ausgeführt
- Problem Sicherheit: wird mit Objektorientierung (sichere Programmiersprache) gelöst
- Kritische Kommandos (Dateien lesen/schreiben, etc) sind nicht ausführbar

Java im World Wide Web
Einführung

❑ Beispiel: Taschenrechner



❑ Taschenrechner-Programm wird auf den Klienten geladen

❑ Und wird dort ohne weiteren Netzwerkverkehr ausgeführt

❑ Solche Programme werden "Applets" genannt (gegenüber "Applications")

Java im World Wide Web
html-Tag

❑ Die java-Klassen und die html-Seiten liegen auf dem WWW-Server

❑ Prinzip: Neuer html-Tag, der Applets referenziert

❑ Festlegung des Codes, der ausgeführt wird

❑ Größe

❑ Parameter (beliebig viele mit beliebigem Namen)

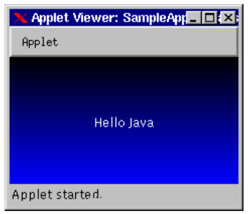
```
...
<APPLET
      ....
      CODE="SampleApplet.class"
      width=200
      height=200
>
<param name=displayString value="Hello Java">
</APPLET>
...
```

Java im World Wide Web

- ❏ Die in der html-Datei angesprochene Klasse muß Subklasse von "Applet" sein
- ❏ Ablauf:
  - Die entsprechende Klasse (SampleApplet) wird geladen
  - Die Klasse wird instanziiert.
  - An dem entstandenen SampleApplet-Objekt wird "init" aufgerufen
  - Dann wird "start" aufgerufen
  - Danach "läuft" das Applet
- ❏ Appletprogrammierung: arbeitet ähnlich wie bei anderen X-Toolkits
  - Ein Applet-Objekt verwaltet einen rechteckigen Bildausschnitt
  - Bei "Events" werden Methoden des Applet-Objektes aufgerufen
  - Durch Reimplementation: anwendungsspezifisches Verhalten

Java im World Wide Web

- ❏ Nach dem Start eines Applets wird bei allen Events eine Methode des Applet-Objektes aufgerufen
- ❏ Beispiele:
  - start, stop, init, destroy
  - mouseEnter, gotFocus, mouseMove, mouseExit, mouseDown, keyDown,...
  - paint, reshape
- ❏ Beispiel: SampleApplet



Java im World Wide Web

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

public class SampleApplet extends Applet {
	String s;
	public void init() {
		s = getParameter("displayString");
	}
	public void paint(Graphics g) {
		for (int y = 0; y < size().height; y++) {
			g.setColor(new Color(0,0,y*255/size().height));
			g.drawLine(0,y,size().width,y);
		}
		g.setColor(Color.white);
		g.drawString(s,
			size().width/2-g.getFontMetrics().stringWidth(s)/2,
			size().height/2+g.getFontMetrics().getHeight()/2
		);
	}
}
```


Java im World Wide Web

- ❏ ".class" Datei und ".html" Datei müssen im gleichen Verzeichnis sein
- ❏ Applets laufen mit netscape: einfach entspr. html-Seite laden ("disable Java" muß deaktiviert sein)
- ❏ Problem: bei "reload" lädt netscape die Klassen nicht neu (Möglich ist allerdings "Shift-reload")
- ❏ Besser daher zum testen: appletviewer
- ❏ Aufruf mit: appletviewer test.html
- ❏ Appletviewer lädt und startet alle Applets der html-Seite