Einführung

- JAVA: altenglisches Wort für Kaffee
- Entwicklung: 1990 - 1995;
  Erfolg erst durch World Wide Web
- einfach (keine Zeiger, keine Mehrfachvererbung,
  automatische Speicherverwaltung)
- objektorientiert
- robust (strenges Typsystem, keine Zeiger,
  automatische Speicherverwaltung)
- dynamische Bindung, dynamisches Laden
- interpretiert (Byte-Code-Interpreter:
  Java Virtual Machine)
- sicher (Byte-Code-Verifizierer)
- portierbar, da architekturunabhängig
- multithreadfähig
- umfangreiche Klassenbibliothek

Datentypen und Variablen

Standard-, Feld-, und Klassentypen

```
int zaehler;
int antwort = 42;
int sinPiHalbe = Math.sin(Math.PI/2);
```

mehrdimensionale Felder

```
int Feld3 [][][];

int x,y;

    ⋮

int Feld3 [][][]
    = new int [3][4][Math.max(x,y)];

int Feld1 [] = {1,2,3,4};

int Feld2 [][] = new int [3][];
Feld2 [0] = new int [2];
Feld2 [1] = new int [4];
Feld2 [2] = new int [6];
```

Kontrollanweisungen

```
if (...) ... else ...
for (...; ...; ...) ...
while (...) ...
do ... while (...);

switch (E) {
   case K_1: ... break;
   ...
   case K_k: ... break;
   default:  ...
}

schleife:
for (int i=0; i<100; i++) {
  for (int j=0; j<100; j++) {
    if (j >=i) break schleife;
    System.out.println("Die Zahl ", j,
                  "ist kleiner als ",i);
  }
}
```

Klassen

```
public class K extends O
                implements S_1, ..., S_k
{
  // Variablen- und Methodendefinitionen
}
```

Beispiel:

```
public class Adresse
{
  public String Strasse;
  public String Ort;

  public void print() {
    System.out.println(Strasse);
    System.out.println(Ort);
  }
}
```

Vererbung

```
public class Adresse
{
  public String Strasse;
  public String Ort;

  public void print() {
    System.out.println(Strasse);
    System.out.println(Ort);
  }
}

public class Privatadresse extends Adresse
{
  public String Vorname;
  public String Nachname;

  public void print() {
    System.out.print(Vorname + " ");
    System.out.println(Nachname);
    super.print();
  }
}
```

Objekte

Instantiierung

```
Privatadresse adr = new Privatadresse();
```

Zugriff auf Objektvariablen

```
adr.Vorname = "Fred";
adr.Nachname = "Feuerstein";
adr.Strasse = "Im Steinbruch 4";
adr.Ort = "99999 Dinodorf";
```

Methodenaufruf

```
adr.print();
```

Konstruktoren

```
public class Adresse
{
  public String Strasse;
  public String Ort;

  public Adresse(String s, String o) {
    Strasse = s;
    Ort = o;
  }
}
public class Privatadresse extends Adresse
{
  public String Vorname;
  public String Nachname;

  public Privatadresse(String v, String n,
                       String s, String o) {
    super(s,o);
    Vorname = v;
    Nachname = n;
  }
}
```

Initialisierung per Konstruktor

```
Privatadresse adr =
  new Privatadresse ("Fred",
                     "Feuerstein",
                     "Im Steinbruch 4",
                     "99999 Dinodorf");
```

Polymorphie

Inklusionspolymorphie

```
public void printAll (Adresse adr) {
  adr.print();
}
```

Methodenüberladung

```
public void printAll(Adresse adr) {
  adr.print();
}

public void printAll(String Strasse,
                     String Ort) {
  System.out.println(Strasse);
  System.out.println(Ort);
}
```

Klassenvariablen, Klassenmethoden, Konstanten

```
public class Adresse
{
  public String Strasse;
  public String Ort;
  public static int adr_anzahl;
  public static final int max_adr = 1000;

  public adresse () {
    Adresse.adr_anzahl ++;
    if (Adresse.adr_anzahl > Adresse.max_adr)
      System.out.print("Maximalzahl ");
      System.out.println("ueberschritten");
  }

  public static void print_anz () {
    System.out.println(adr_anzahl);
  }
}
```

Aufrufsyntax

Zugriff auf Klassenvariablen:

```
System.out.println(Adresse.adr_anzahl);
```

Aufruf von Klassenmethoden:

```
Adresse.print_anz();
```

Zugriff auf Objektvariablen:

```
Adresse adr = new Adresse(
                  "Im Steinbruch 4",
                  "99999 Dinodorf");

System.out.println(adr.Strasse);
```

Aufruf von Objektmethoden:

```
adr.print();
```

Pakete, Sichtbarkeit

Pakete:

```
package MeinPaket;

import java.io.*;
import java.util.Hashtable;

//class definitions
```

Sichtbarkeit:

- Standard: alle Klassen im eigenen Paket
- `private`: eigene Klasse
- `protected`: eigene Klasse und alle Teilklassen
- `public`: alle Klassen in allen Paketen