3. Grundlegende Sprachkonstruktionen imperativer Programme

Java-Beispiele:

Temperature.java
Keyboard.java

Schwerpunkte

- Imperative Programmierung
- Beispiel für ein Programm aus drei Komponenten
- Variable, Datentyp, Zuweisung
- Einfache Ein- und Ausgabe
- Standardbibliothek: Java-API

Imperative Programmierung

Beispiel: Temperaturumwandlung

Temperature.java

```
class Temperature {
   // Convert temperature
   // from Fahrenheit to Centigrade (Celsius)

   public static void main (String[] args) {
        double tempFahr; // Fahrenheit
        double tempCels; // Celsius

        System.out.print("Temperature (deg F): ");
        tempFahr = Keyboard.readDouble();

        tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;

        System.out.print(tempFahr);
        System.out.print(" deg F is ");
        System.out.print(tempCels);
        System.out.println(" deg C");
   }
}
```

ein imperatives Programm

```
% javac Temperature.java
% java Temperature

Temperature (deg F): 10
10 deg F is -12.222222222222221 deg C
```

Temperatur-Beispiel: als Pascal-Programm
(Standard-Pascal)

```
PROGRAM Temperature;
   {Convert temperature
    from Fahrenheit to Centigrade (Celsius)}

VAR tempFahr: real;
    tempCels: real;

BEGIN
   writeln("Temperature (deg F): ");
   readln(tempFahr);

   tempCels := (5.0 * (tempFahr - 32.0)) / 9.0;

   write(tempFahr);
   write(" deg F is ");
   write(tempCels);
   writeln(" deg C");
END.
```


Imperative Programmierung: Grundansatz

Imperative Programmierung orientiert auf die Beschreibung von Algorithmen.

- **Algorithmus:**
  Verfahren zur Berechnung gesuchter Werte aus gegebenen Werten, . . . das auf der schrittweisen Ausführung von elementaren Verarbeitungsoperationen beruht.

- **Imperatives Programm:**
  Algorithmen durch Bearbeitung der Variablenwerte beschrieben
  (Verändern und Lesen der Werte)

```
tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;
```


Imperatives Pascal-Programm

```
PROGRAM Temperature ;

VAR tempFahr : real ;
    tempCels : real ;


BEGIN
   ...
   readln (tempFahr);

   tempCels := (5.0 * tempFahr ...);

   write (tempCels);

END.
```

Eingabewerte in Eingangsvariablen

Algorithmus

Ausgabewerte in Ausgangsvariablen

Imperatives Java–Programm

```
class Temperature   {
   public static void main (...)   {
        double tempCels;
        double tempFahr;
        ...

        tempFahr = Keyboard.readDouble();


        tempCels = (5.0 * tempFahr ...);

        System.out.print(tempCels);
   }
}
```

Eingabewerte in Eingangsvariablen

Algorithmus

Ausgabewerte in Ausgangsvariablen

Imperative Programmierung: Eigenschaften im Detail

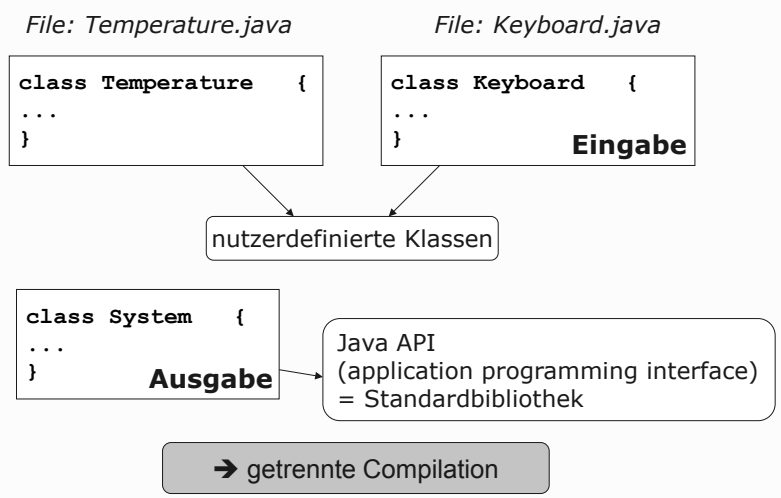
- Basiskonzepte: Variable, Anweisung
- Variable:
  besitzt Wert, der durch Anweisungen gelesen und verändert wird
- Anweisung:
  dient dem Zugriff auf Variablen-Werte
  (Lesen und Verändern von Werten)
- Grundlegende Strukturierungsmethode imperativer Programme:
  Prozedur (Funktion, Methode)
- Prozedur (Funktion, Methode):
  Teilalgorithmus: mit Anweisungen der Sprache

Mehrkomponenten-Programm

Aus wie vielen Komponenten besteht das Java-Programm?

```
class Temperature {
   // Convert temperature
   // from Fahrenheit to Centigrade (Celsius)

   public static void main (String[] args) {
        double tempFahr; // Fahrenheit
        double tempCels; // Celsius

        System.out.print("Temperature (deg F): ");
        tempFahr = Keyboard.readDouble();

        tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;

        System.out.print(tempFahr);
        System.out.print(" deg F is ");
        System.out.print(tempCels);
        System.out.println(" deg C");
   }
}
```


Temperatur-Programm: besteht aus d r e i Komponenten



Klasse 'Keyboard'

Keyboard.java

```
import java.io.*;

class Keyboard {

// Author: M. Dennis Mickunas,
// June 9, 1997 Primitive Keyboard
// input of integers, reals,
// strings, and characters.

static boolean iseof = false;
static char c;
static int i;
static double d;
static String s;

/* WARNING: THE BUFFER VALUE IS SET
TO 1 HERE TO OVERCOME ** A KNOWN BUG
IN WIN95 (WITH JDK 1.1.3 ONWARDS)*/

static BufferedReader input
   = new BufferedReader (
         new
InputStreamReader(System.in),1);
```

```
public static int readInt () {
   if (iseof) return 0;
   System.out.flush();
   try {
         s = input.readLine();
   }
   catch (IOException e) {
         System.exit(-1);
   }
   if (s==null) {
         iseof=true;
         return 0;
   }
   i = new
Integer(s.trim()).intValue();
   return i;
}

public static char readChar () {
   if (iseof) return (char)0;
   System.out.flush();
   ...
```

Aufgabe der Klasse?

Mangel: keine übersichtliche Darstellung für die Nutzer der Klasse

Klasse Keyboard: eine Abstraktion

Nur das ist für die Anwendung wichtig

```
class Keyboard {

   public static int readInt () ;

   public static char readChar () ;

   public static double readDouble () ;

   public static String readString () ;

   public static boolean eof () ;
}
```

→ **Sammlung nützlicher Funktionen zur Eingabe von ganzen Zahlen, Zeichen, reellen Zahlen, Zeichenketten von der Tastatur**

Temperatur-Programm: Interface zwischen Komponenten

File: Temperature.java

```
class Temperature {

   public static void main (String [] args) {
        ...
        tempFahr = Keyboard.readDouble();

        System.out.print(" deg F is ");
        ...
   }
}
```

Java API (=Standardbibliothek)

nutzerdefinierte Klassen

```
class System {
   ...
   public....out;
}
```

```
class Keyboard {
   ...
   public readDouble (...)
        ...
}
```

File: Keyboard.java

Grundelemente von Java-Programmen

Grundstruktur von Java-Programmen

K l a s s e = grundlegende Komponente

Klassenname

Klassenkörper (Beginn)

```
class Temperature {
   public static void main (String [] args)   {
        double tempFahr;
        ...
   }
}
```

Methodendeklaration

Variablendeklaration

Klasse: 'Sammlung' von
- Variablendeklarationen
- Methodendeklarationen
  (Methode = Algorithmus, Prozedur, Funktion)

*je Gesamtprogramm: e i n e Methode **m a i n ( )***
→ dort beginnt die Abarbeitung !
(wie Hauptprogramm in Pascal)

Variablendeklaration

```
double tempFahr;
```

Typ

Variable

Wirkung:

1. Wertebereich der Variablen festlegen
2. Speicherplatzgröße entsprechend Typ festgelegt
   z. B. double: 8 Byte
3. erlaubte Operationen

einfache Typen:

Java-EBNF: 'Standardtyp'
boolean, char, byte, short, int, long, float, double

Kommentare

```
class Temperature {
      // Convert temperature
      // from ...

      double tempFahr; // Fahrenheit
// bis Zeilenende

                        /* dazwischen */
      int /* nur hier */ temp;
```


Einfache Anweisungen: Zuweisung und Aufruf einer Methode

aus Klasse Keyboard: Methode readDouble

```
tempFahr = Keyboard.readDouble();
```

Zuweisung

```
tempCels = (5.0 * (tempFahr - 32.0)) / 9.0;


System.out.print(tempFahr);
```

Aufruf einer Methode
(Eingabe / Ausgabe)