2 Programmieren in Java I – noch ohne Nachbearbeitung

2.1 Was sind Programme?

$$\boxed{\text{Eingabe}} \Longrightarrow \boxed{\text{Verarbeitung}} \Longrightarrow \boxed{\text{Ausgabe}}$$

Die **Eingabe** kann sein

- Konstanten im Quelltext;
- Kommandozeilenparameter oder interaktive Eingabe oder GUI Dialog;
- eine lesbare oder binär kodierte Datendatei.

Die **Ausgabe** kann erfolgen als

- Text auf der Kommandozeile;
- Element einer *grafischen Benutzeroberfläche* (GUI) wie Tabelle, Grafik;
- Datendatei.

2.2 Minimales Java–Programm

java/Muster.java

```
class Muster{

 public static void main(String[] args){
 }
}
```

- progname.java enthält die Objektklasse class progname
- die Hauptklasse eines Programms enthält eine Methode main, die vom java–Interpreter gesucht und ausgeführt wird. Alle anderen Programmbestandteile werden von main aufgerufen.
  - main erhält in (String [] args) eine Liste von Zeichenketten mit den Kommandozeilenargumenten.
  - main hat den Rückgabetyp
    void, leer, oder
    int für einen ganzzahligen Rückgabewert als Fehlernummer.
  - main hat die Eigenschaften
    public, von außen sichtbar und aufrufbar, und
    static, ist Methode der Klasse, unabhängig von Instanzen.

java/Muster.java

```
class Muster{

 public static void main(String[] args){
 }
}
```

2.3 Einfache Textausgabe

java/Hallo.java

```
class Hallo{

 public static void main(String[] args){
  System.out.println("Nummer␣5␣lebt!");
  if(args.length > 0) System.out.println(args[0]);
 }
}
```

Von den Systemfunktionen wird die Konsolenausgabe out angesprochen und von dieser die Methode println „print line“ = „Drucke eine Zeile“

2.4 Programm mit Konstruktor

java/Hello.java

```
class Hello{

  Hello(){
    System.out.println("Hello␣World!");
  }

  public static void main(String[] args){
    new Hello();
  }
}
```

Vorteil: Erlaubt die Definition globaler Variablen, empfohlene Struktur für Turtle– und allgemein GUI-Programme.

2.5 Ein einfaches Turtle–Programm

aplu/Step.java

```
import ch.aplu.turtle.*;
```

```
class Step{

  public static void main(String[] args){
    Turtle om=new Turtle();
    om.left(13);  om.forward(50);
    om.right(47); om.back(100);
  }

}
```

Turtle om; — Deklarieren der Turtle om;

om=**new** Turtle(); — Anlegen und Initialisieren der Turtle om.

2.6 Variable

Variablen werden deklariert (dem Compiler bekanntgegeben) durch eine Anweisung der Form

Typ Name;

oder

Typ Name=Anfangswert;

Mehrere Deklarationen desselben Typs können, durch Kommata getrennt, zusammengefasst werden:

int max=100, i, j, k, zaehler=0;

Typen sind

char	ganze Zahlen mit Vorzeichen 8Bit=1Byte von $-2^7 = -128$ bis $2^7 - 1 = 127$;
short	ganze Zahlen mit Vorzeichen 16Bit von $-2^{15} = -32768$ bis $2^{15} - 1 = 32767$;
int	ganze Zahlen mit Vorzeichen 32Bit=4Byte von $-2^{31} \approx -2.1 \cdot 10^9$ bis $2^{31} - 1 \approx 2.1 \cdot 10^9$;
long	ganze Zahlen mit Vorzeichen 64Bit=8Byte von $-2^{63} \approx -9 \cdot 10^{18}$ bis $2^{63} - 1 \approx 9 \cdot 10^{18}$;
float	Gleitkommazahlen einfache Genauigkeit 7 Dezimalstellen
double	Gleitkommazahlen doppelte Genauigkeit 16 Dezimalstellen
boolean	Wahrheitswert
String	Zeichenkette (eine Objektklasse)
andere	Objektklassen

Mit Zahlentypen kann gerechnet werden:

java/Rechnen.java

```
class Rechnen{

 public static void main(String[] args){
   int a=13, b=5;
   System.out.println("a␣=␣" + a + ",␣b␣=␣" + b);
   System.out.println("a+b␣=␣" + (a+b));
   System.out.println("a*b␣=␣" + (a*b));
   System.out.println("a/b␣=␣" + (a/b)+"␣␣␣(Ganzzahldivision)");
   System.out.println("a%b␣=␣" + (a%b)+"␣␣␣(Rest␣der␣Division)");
 }
}
```

Ergebnis der Ganzzahldivision ist wieder eine ganze Zahl. Vorsicht, $1/3$ ergibt dabei 0.

Bei positiven Dividend und Divisor ergibt q=a/b; r=a%b die korrekte Division mit Rest, $a = q \cdot b + r$. Untersuchung des Einflusses von negativen Vorzeichen als Übung.

2.7 Weitere Rechenoperationen

Gleitkomma:	+ − ∗ /	
Integer:	+ − ∗ / %	
Boolean:	! && ||	(nicht, und, oder)
String:	+	
Vergleiche:	< > <= >= == !=	

Klammern legen die Auswertungsreihenfolge eindeutig fest.

Beispiele:

- y=a∗(b+10);
- **if**((a<=10) || (a>99)){...}
- str="Guten␣Morgen"+name+".";

2.8 Iteration und andere Schleifen

Zählschleife: **for**(**int** k=0; k<N; k++){ Anweisungen;}

führt die Anweisung für jedes k von 0 bis $N-1$ aus.

while-Schleife: **while**(Bedingung){ Anweisung; }

führt die Anweisung aus, solange die Bedingung erfüllt ist.

do-while-Schleife: **do**{ Anweisung; } **while**(Bedingung);

führt die Anweisung aus, bis Bedingung das erste Mal verletzt ist.

2.9 Verzweigungen

Bedingte Ausführung: **if**(Bedingung) { Anweisung; }

Entscheidung: **if**(Bedingung) {Anweisung1;} **else** {Anweisung2;}

Auswahl: Der Selektor k sei eine Integer–Variable

```
switch(k){
 case 1: Anweisung1; break;
 case 2: Anweisung2; break;
 case 3: Anweisung3; break;
 default: Anweisung4;
}
```

Auch **char** c ist als Selektor möglich, dann Fallanwahl mit **case** 'a' etc.

2.10 Funktionen und Prozeduren

2.11 Rekursive Aufrufe