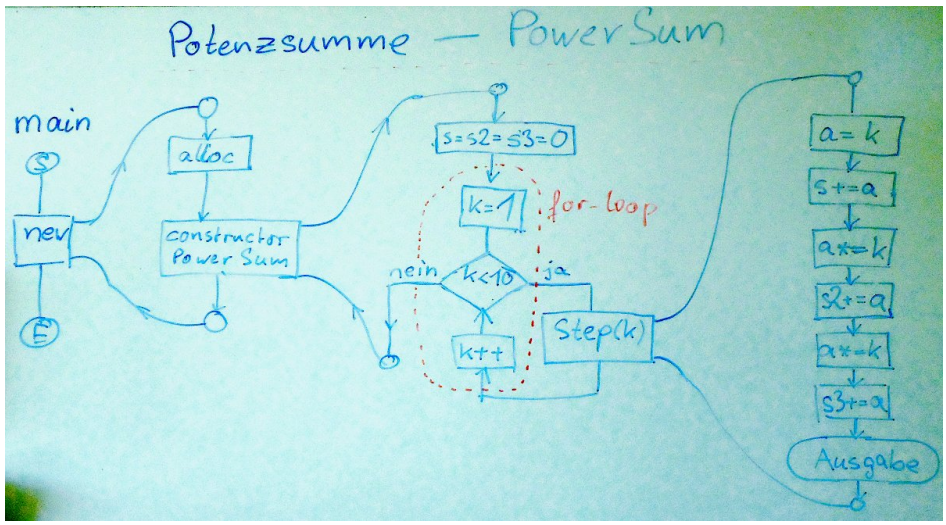
3 Programmieren in Java II

3.1 Programmablaufplan (für PowerSum.java)



Kreise für Anfang und Ende des Programms oder Unterprogramms

Rechtecke für Anweisungen

Rhomben für Verzweigungen

3.2 Unterprogramme

- Ein Unterprogramm (UP) ist ein benannter Block von Anweisungen.
- Bei Aufruf eines UP kehrt die Programmsteuerung nach Abarbeiten des UP zur nächsten Anweisung nach dem Aufruf zurück.
- Ein UP kann *Parameter* und Rückgabewerte haben.
- Eine *Funktion* ist ein Unterprogramm mit, eine *Prozedur* ein UP ohne Rückgabewert.
- *Methoden* sind die einer Klasse zugeordneten Unterprogramme.

In Java hat ein Unterprogramm die Form

Modifikatoren Rückgabetyp Name(Parameterliste) Anweisungsblock

- Modifikatoren sind Schlüsselworte wie **static**, **public**, **private**, ...

- Eine Parameterliste] ist leer oder besteht aus Komma–getrennten Paaren von Datentyp und Variablenname. Gilt auch als Variablendeklaration.
- Prozeduren haben den Rückgabetyp void.

  Alle anderen Rückgabetypen erfordern eine **return**–Anweisung, in der ein Ausdruck vom Ergebnistyp steht.

Ein Anweisungsblock enthält, jeweils mit Semikolon abgeschlossen,

- Variablendeklarationen,
- Zuweisungen, wobei die linke Seite immer eine Variable ist und rechte Seite ein arithmetischer Ausdruck oder ein Funktionsaufruf sein kann,
- Prozeduraufrufe
- Schleifen und Verzweigungen
- durch { } eingeschlossene Anweisungsblöcke

Binom.java

3.3 Interaktive Eingabe mit Java

Zwei Varianten: mit Hilfsklassen oder mit Scanner

Hilfsklassen `int n=Integer.parseInt(string)`

- ist die ältere Variante,
- kann kürzer sein,
- Abfangen von Fehleingaben ist schwieriger.

```
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
```

Scanner relativ neu,

erfordert das Erzeugen einer Instanz der Klasse Scanner,

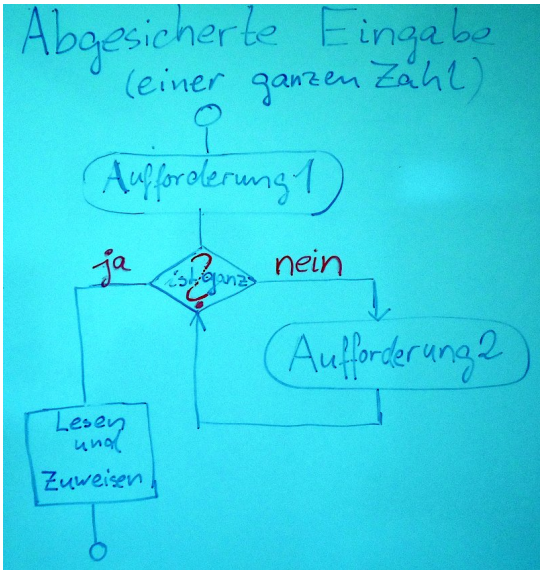
kann vor dem Zuweisen die Korrektheit der Eingabe prüfen.

java/ScanInput.java

```
import java.util.*;

public class ScanInput {
```

```
  public static void main(String args[]) {
    Scanner sc=new Scanner(System.in);

    System.out.print("Warte␣auf␣Eingabe:␣");
    while(sc.hasNext()){
      if(sc.hasNextInt()){
        int n=sc.nextInt();
        System.out.println("Token␣ist␣Integer:␣"+n);
        continue;
      }

      if(sc.hasNextDouble()){
        double x=sc.nextDouble();
        System.out.println("Token␣ist␣Double:␣"+x);
        continue;
      }
      if(sc.hasNext()){
        String s=sc.next();
        System.out.println("Token␣ist␣String:␣"+s);
        if(s.equals("s")) break;
        continue;
      }
    }
  }
}
```

3.4 Abgesicherte Eingabe



- Prüfe die Eingabe auf korrekte Lesbarkeit
- Wenn fehlerhaft, dann verwirf die Eingabe und frage nach neuer Eingabe
- Wenn die Eingabe lesbar ist, dann wird sie als Wert interpretiert und der Variablen zugewiesen.

```
    Scanner sc=new Scanner(System.in);
    System.out.print("Nenner␣n=");
    while(!sc.hasNextInt()){
      System.out.print(
        sc.next()+"␣ist␣keine␣ganze␣Zahl."+
```

```
        "␣Wiederholen:␣");
    }
    n=sc.nextInt();
```

3.5 Abgesicherte Eingabe als Unterprogramm

```
  static int inputInt(String prompt){
    Scanner sc=new Scanner(System.in);
    System.out.print(prompt);
    while(!sc.hasNextInt()){
      sc.next()
      System.out.print(
        "Error␣reading␣Integer.␣Repeat:␣");
    }
    return sc.nextInt();
  }
```

3.6 Gültigkeit von Variablen

java/LokalGlobal1.java

```
public class LokalGlobal1 {

  // globale Variable i


  /* global */static int i = 5; //Kommentar

  public static void main(String args[]) {
    tuwas();
    System.out.println(i);
  }

  static void tuwas() {
    // globale Variable i wird veraendert
    i = 6;
  }
}
```

java/LokalGlobal2.java

```
public class LokalGlobal2 {

  // globale Variable i
  static int i = 5;

  public static void main(String args[]) {
    tuwas();
    System.out.println(i);
  }

  static void tuwas() {
    // lokale Variable i wird gesetzt
    int i = 6;
  }
}
```

Jede Variable ist nur in ihrem umgebenden Block gültig.

Eine in einem inneren Block definierte Variable überlagert (bis zum Blockende) schon vorhandene Variablen gleichen Namens.