

Strukturierte Programmierung

Inhalt:

- [1. Das erste Programm](#)
- [2. Variablen \(einfache Datentypen\)](#)
- [3. Rechnen](#)
- [4. Eingabe über die Tastatur](#)
- [5. IF-Anweisung](#)
- [6. IF-Else-Anweisung](#)
- [7. IF-Anweisung \(UND, ODER, NICHT\)](#)
- [8. Mehrfachverzweigung](#)
- [9. Switch-case-Anweisung](#)
- [10. Schleifen](#)
- [11. for - Schleifen](#)
- [12. Arrays \(Felder\)](#)
- [13. foreach-Schleifen und Arrays](#)
- [14. Datei-EA](#)
- [15. Methoden ohne Parameter](#)
- [16. Methoden mit Parameter](#)
- [17. Methoden mit Rückgabewerten](#)
- [18. Arrays als Parameter](#)
- [19. Stringverarbeitung](#)
- [20. ArrayListen](#)
- [21. Objekte als eigene Datentypen](#)
- [22. Zeichenfunktionen](#)



Strukturierte Programmierung

Das erste Programm



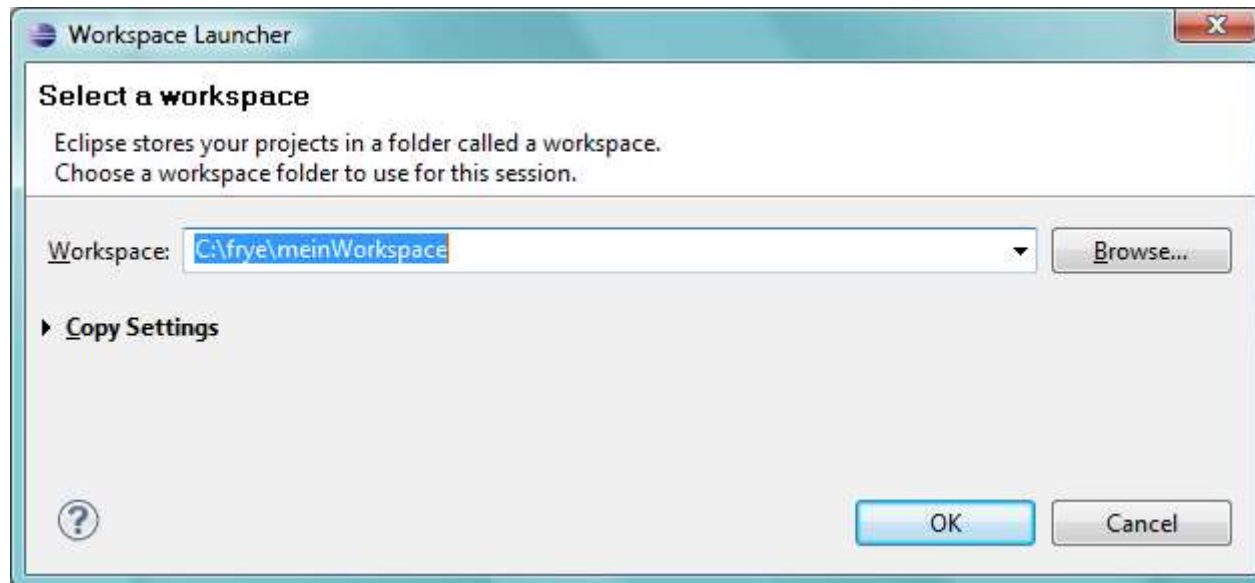
Arbeiten mit JAVA und Eclipse

- Hier lernen Sie ein erstes Programm in JAVA unter der Entwicklungsumgebung „Eclipse“ zu programmieren und auszuführen.
- Die aktuelle Version von Eclipse gibt es unter <http://www.eclipse.org/downloads/>
- Außerdem benötigen Sie (mindestens) die [JAVA Runtime Engine \(JRE\)](#).
- Eine Anleitung, wie Sie direkt mit JAVA (Compiler, Runtime) und einen beliebigen Editor programmieren können, finden Sie im folgendem [Tutorial](#).



Eclipse starten

- Nach dem Start von Eclipse müssen Sie zunächst einen „Workspace“ wählen. Darin werden der Programm Quellcode und die kompilierten Dateien gespeichert.



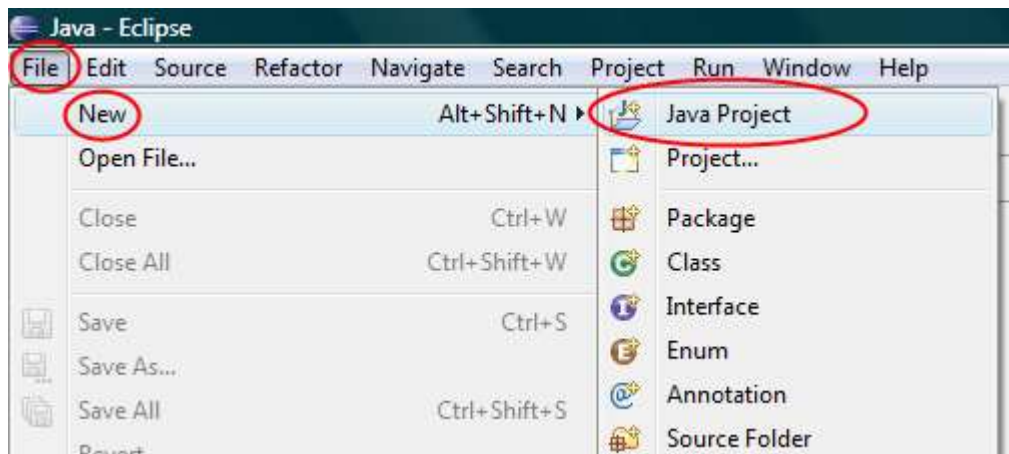
Eclipse starten

- Es öffnen der Startbildschirm. Schließen Sie diesen:

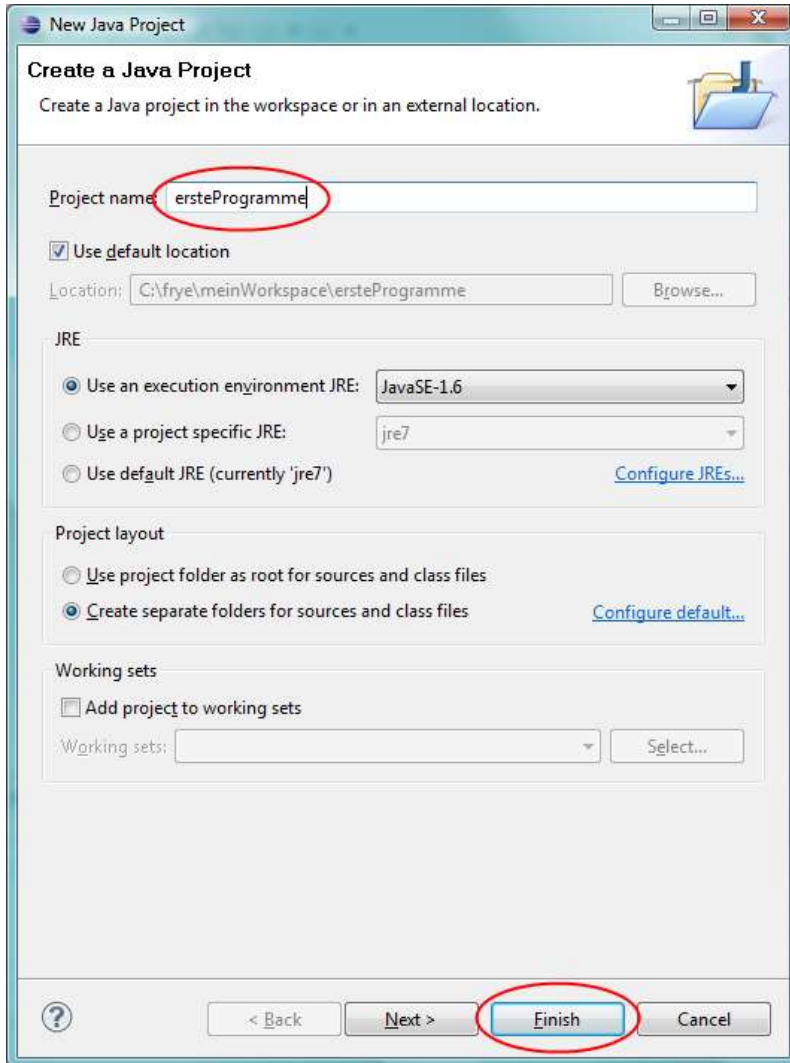


JAVA Projekt

- Zunächst müssen Sie ein neues JAVA-Projekt anlegen:



JAVA Projekt



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: ersteProgramme

☒ Use default location

Location: C:\frye\meinWorkspace\ersteProgramme [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-1.6

☐ Use a project specific JRE: jre7

☐ Use default JRE (currently 'jre7') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

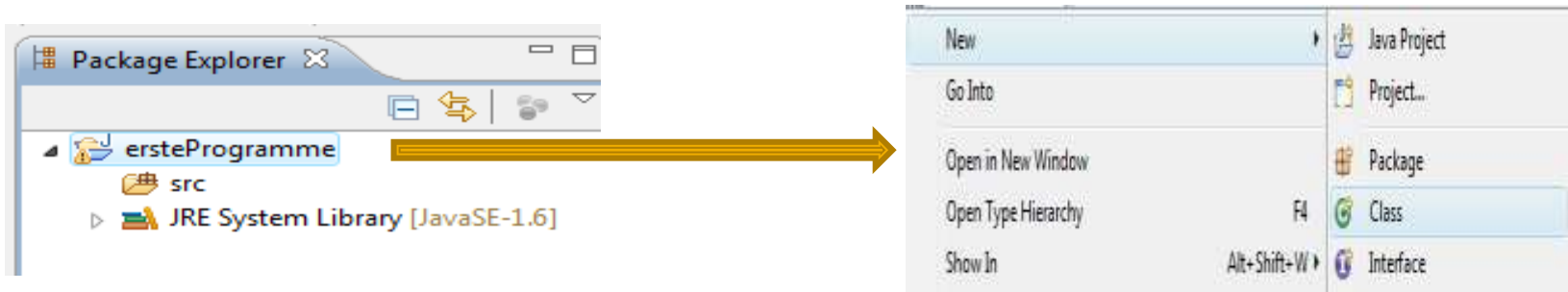
☐ Add project to working sets

Working sets: [Select...](#)

[? < Back](#) [Next >](#) **Finish** [Cancel](#)

- Vergeben Sie einen Namen für Ihr Projekt.
- Bestätigen Sie mit „Finish“

Klasse erstellen



- Ihr Projekt erscheint dann im „Package Explorer“ (siehe links).
- Als nächstes müssen Sie im Projekt ein Programm (in JAVA „Klassen“) erstellen.
- Führen Sie dazu einen Rechtsklick auf den Projektnamen durch und wählen sie „New->Class“.

Klasse erstelleb

New Java Class

Java Class

The use of the default package is discouraged.

Source folder: ersteProgramme/src Browse...

Package: (default) Browse...

Enclosing type: Browse...

Name: helloWorld

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

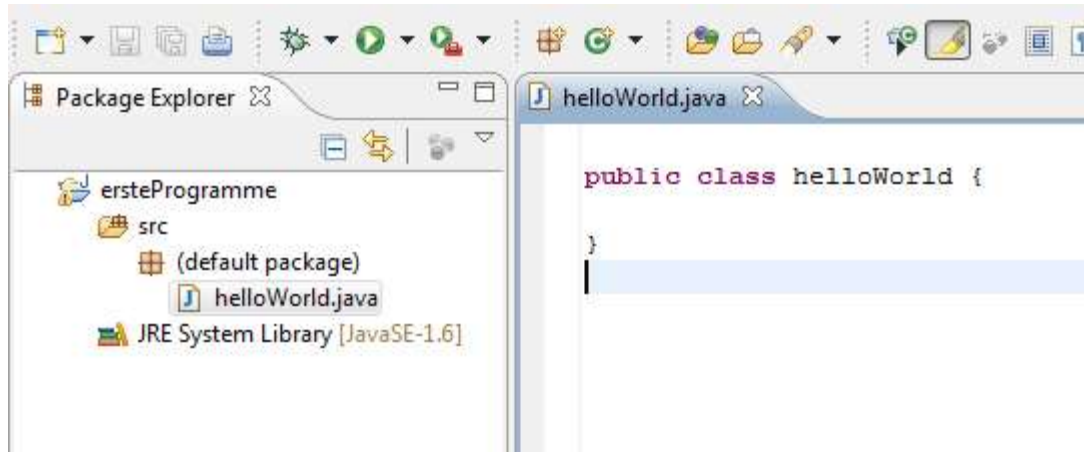
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Finish Cancel

- Vergeben Sie einen Klassennamen.
- Drücken Sie auf „Finish“

Programmgerüst



- Eclipse hat die Klasse automatisch im Package Explorer eingefügt.
- Außerdem hat Eclipse das Programmgerüst generiert (mittleres Fenster). Dort können Sie anfangen zu programmieren.



Main – Methode

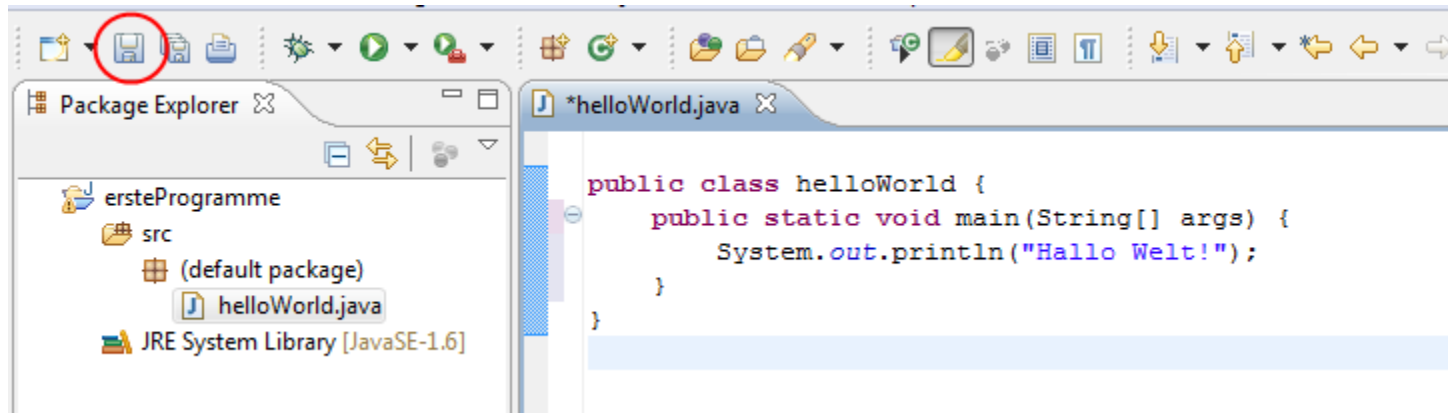
- Damit eine Klasse direkt ausgeführt werden kann, benötigt diese als „Einsprungspunkt“ eine „Main-Methode. Diese lautet in JAVA:

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

- Innerhalb dieser Methode schreiben Sie (zunächst) Ihre Programmanweisungen.

Beispiel

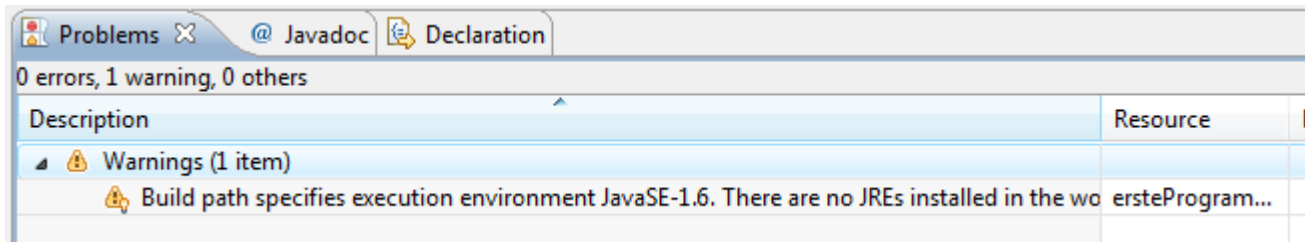
- Es soll ein Programm geschrieben werden, welches „Hallo Welt!“ auf dem Bildschirm ausgibt:



- Drücken Sie dann auf das speichern Symbol.

Programm speichern

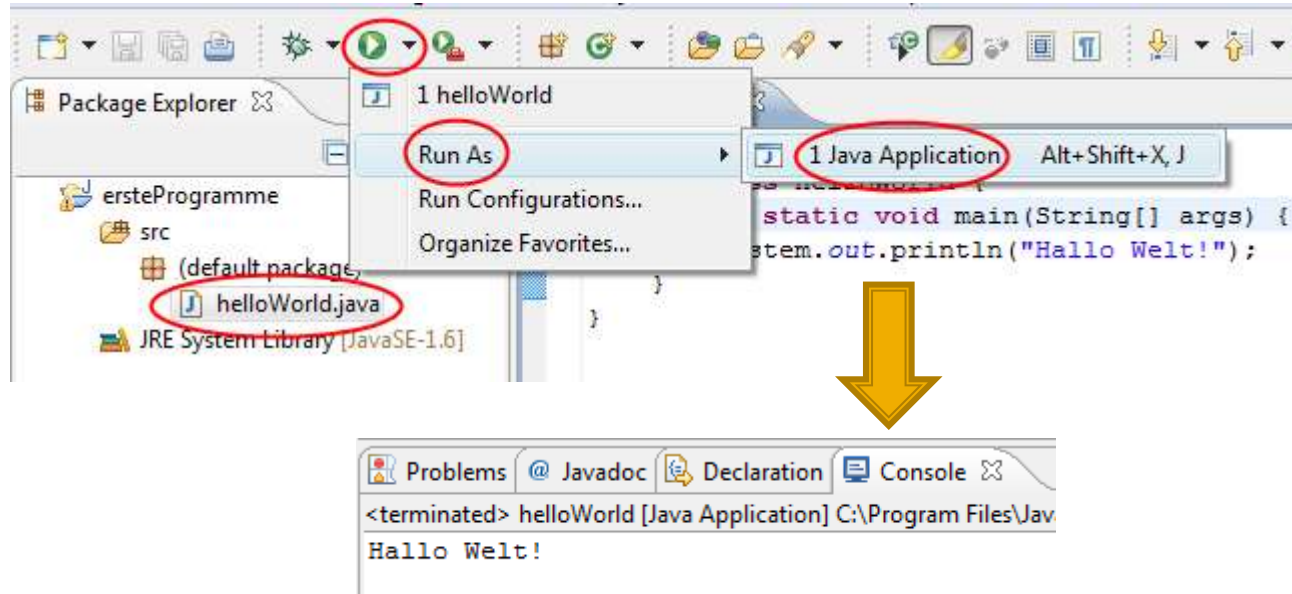
- Mit dem Abspeichern wird das Programm auf Fehler überprüft. Fehler werden im Fenster „Problems“ angezeigt:



- Aufgabe: Probieren Sie aus, wie sich die Anzeige ändert, wenn Sie Fehler einbauen.

Programm ausführen

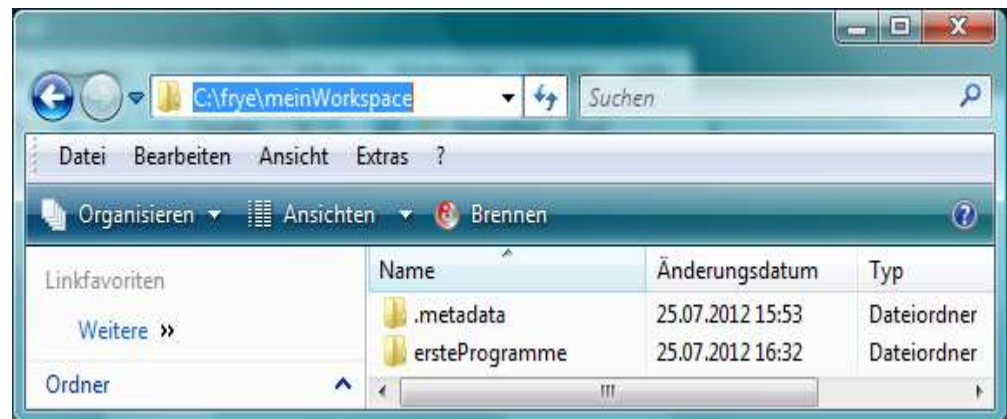
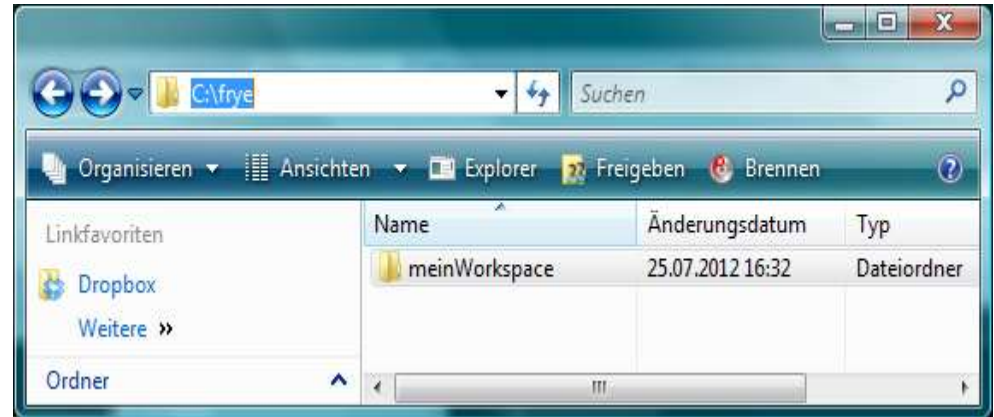
- Ist das Programm fehlerfrei, kann es ausgeführt werden:



- Wichtig: Es muss dabei eine Klasse im Package Explorer ausgewählt sein, die eine main-Methode besitzt.

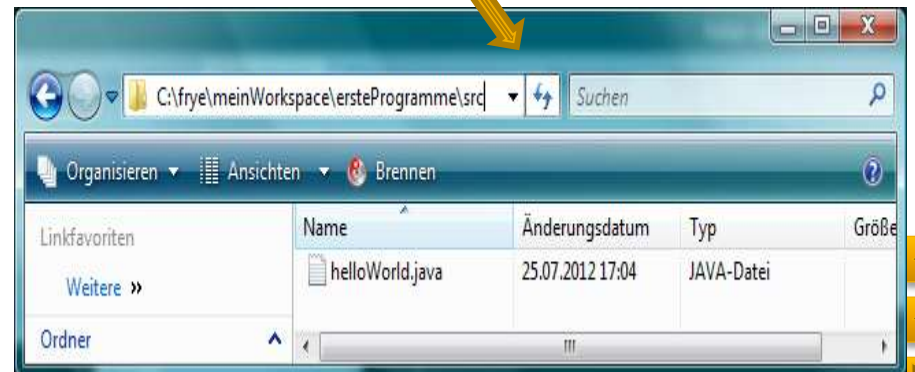
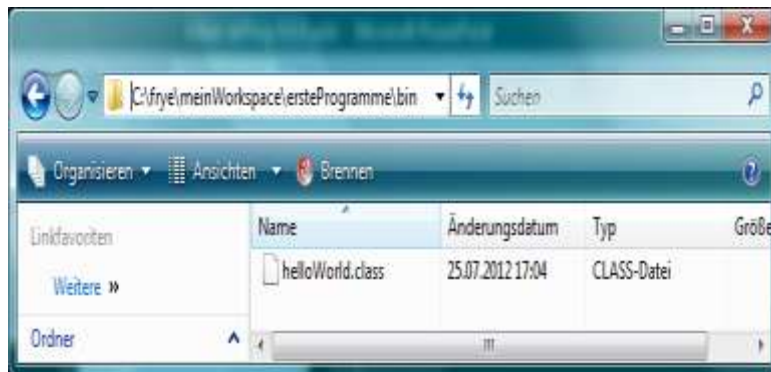
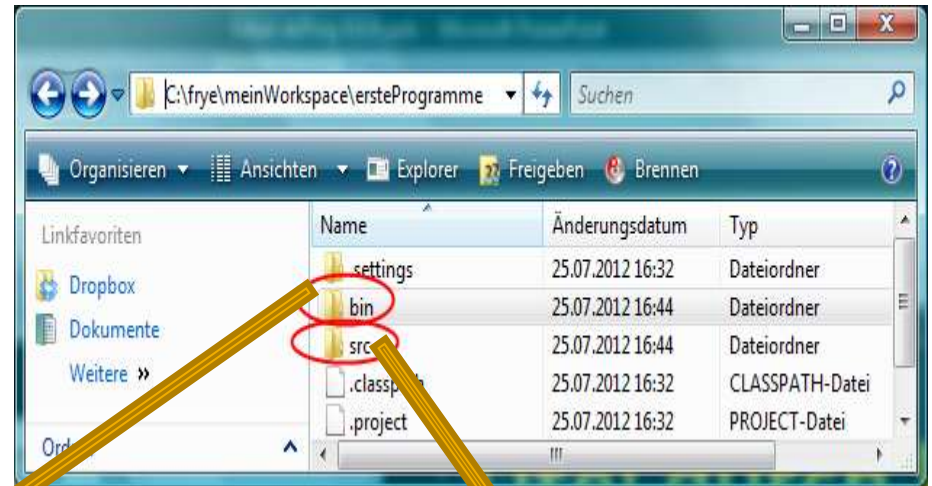
Angelegte Ordnerstruktur

- Ein Workspace ist ein Ordner (Verzeichnis) auf dem gewählten Laufwerk.
- Im Workspace-Ordner ist für jedes Java-Projekt ein Projektordner angelegt.



Ordnerstruktur

- Der Projektordner enthält einen Ordner für den kompilierten Code (bin) und einen für den Quellcode (src).

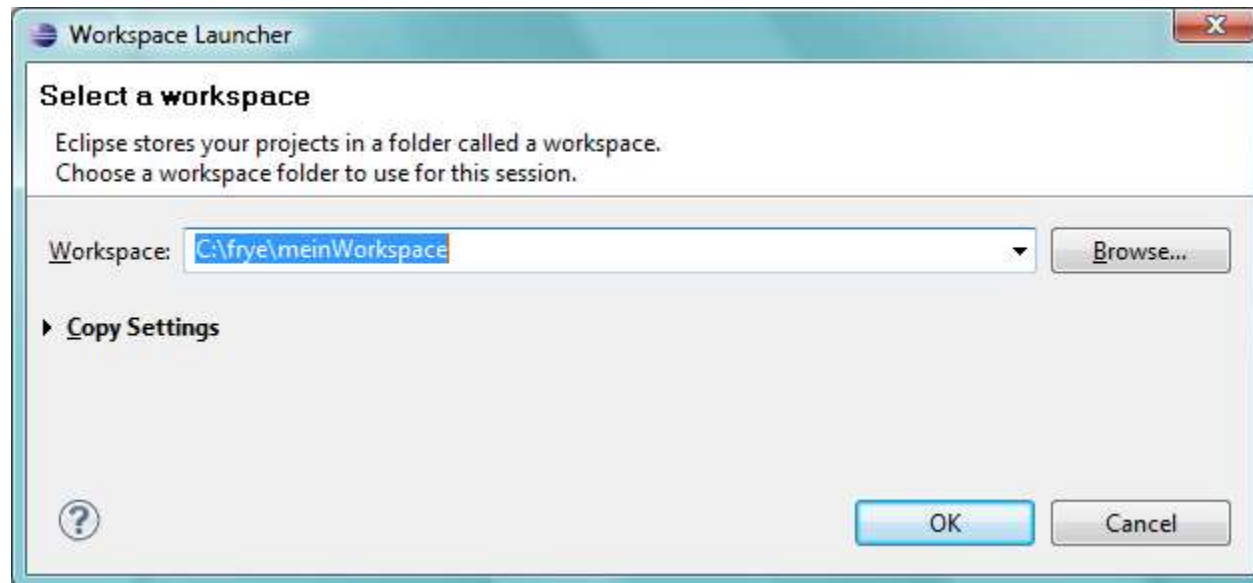


Programme sichern

- Um ein fertiges Programm zu sichern, gibt es mehrere Möglichkeiten:
 - Speichern des Workspace-Ordners (Vorteil: Einfach, Nachteil: Viel „Overhead“).
 - Man speichert nur den Quellcode (.java-Datei). (Vorteil: einfache Textdatei, wenig Speicher, Nachteil: es muss bei Wiederverwendung in Eclipse ein neues Projekt angelegt werden).

Programme wieder öffnen

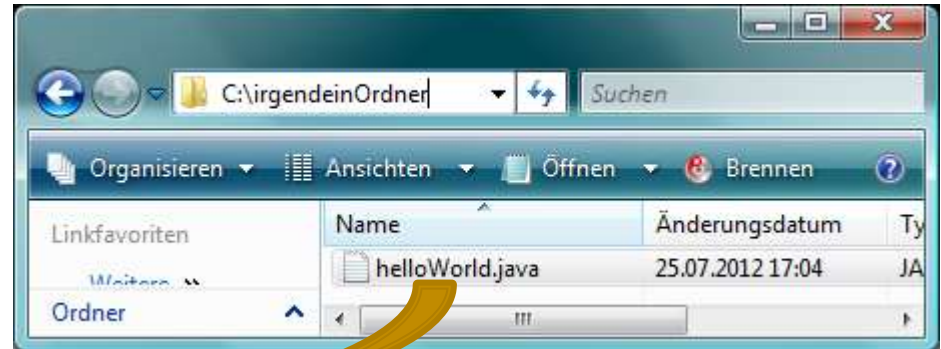
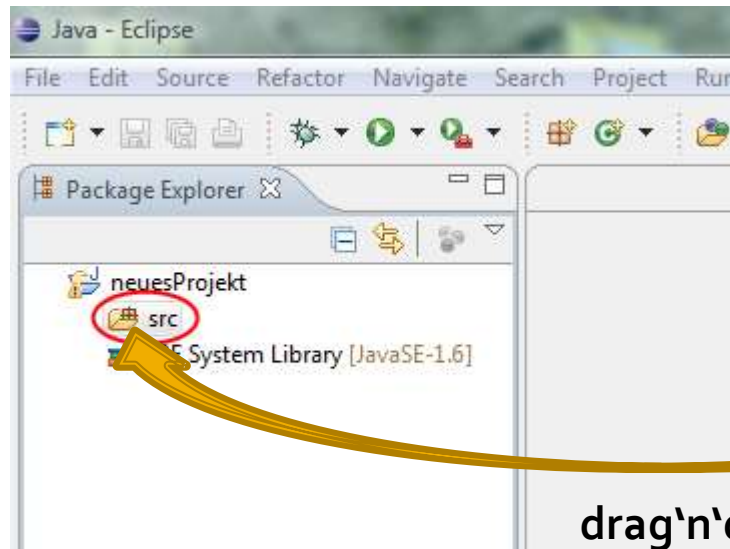
- Wenn der Workspace-Ordner gesichert wurde gibt man den Pfad einfach bei Programmstart an.



Programme öffnen

- Wenn man nur den Quellcode hat, legt man zuerst ein neues Projekt an und zieht die Datei per Drag'n'Drop in den Package-Explorer (in „src“).
- Hinweis: Verwenden Sie nicht den Öffnen-Befehl. Dieser öffnet zwar die Datei. Bindet diese aber nicht in das Projekt ein!

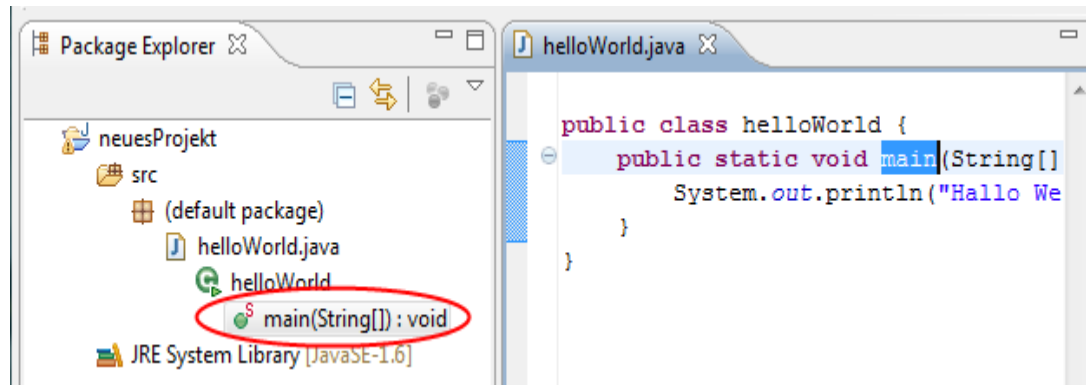
Quellcode in Projekt einbinden



drag'n'drop

Wählen Sie dann „Dateien kopieren“

Ergebnis:



Klicken Sie dann doppelt auf die main-Methode. Sie können jetzt das Programm bearbeiten.



Strukturierte Programmierung

Variablen



Variablen

- Variablen sind kleine Speicherplätze, in denen Werte gespeichert werden.
- Variablen müssen einen Namen haben, über den Sie im Programmcode angesprochen werden: z.B. `x`, `y`, `zahl`, `betrag1`, `hans4`
- Den Variablen können Werte mit dem `=` - Operator zugewiesen werden: z.B. `x = 5`

Datentypen

- Die Variablen benötigen einen Datentyp. Dieser gibt an, welche Werte in der Variablen gespeichert werden können.
- Im Programmcode muss dieser Datentyp einmalig „mitgeteilt“ werden.
- Variablen werden mit dem Datentyp gefolgt von dem Variablennamen deklariert.

Beispiele

```
public class VariablenTest {  
    public static void main(String[] args) {  
  
        // eine Variable für eine Ganzzahl  
        int zahl1;  
        // die Variable initialisieren (erstmal einen Wert zuweisen)  
        zahl1 = 12;  
        // den Wert der Variablen auf dem Bildschirm ausgeben  
        System.out.println("Die Variable zahl1 hat den Wert: " + zahl1);  
        // der Variablen einen neuen Wert zuweisen  
        zahl1 = -1467;  
        // den Wert wieder auf dem Bildschirm ausgeben  
        System.out.println("Die Variable zahl1 hat den Wert: " + zahl1);  
    }  
}
```

A screenshot of a Java IDE console window. The window has tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, showing the output of the program. The output consists of two lines: 'Die Variable zahl1 hat den Wert: 12' and 'Die Variable zahl1 hat den Wert: -1467'. The window title bar shows '<terminated> VariablenTest [Java Application] /System/Library/Fr'.

Beispiel

```
public class VariablenTest2 {  
    public static void main(String[] args) {  
        /* eine Variable für eine Kommzahl anlegen  
        * und diese gleichzeitig initialisieren  
        * als Komma muss ein Punkt genommen werden  
        */  
        double kommazahl1 = 24.6567;  
        // den Wert der Variablen auf dem Bildschirm ausgeben  
        System.out.println("Die Variable kommazahl1 hat den Wert: " + kommazahl1);  
        // eine Variable für einen Text anlegen  
        String txt = "Hallo 11 FOI 0";  
        // den Wert wieder auf dem Bildschirm ausgeben  
        System.out.println("Der Text der Variablen txt lautet: " + txt);  
    }  
}
```



The screenshot shows an IDE window with a tab labeled 'Console'. The output text is as follows:

```
<terminated> VariablenTest2 [Java Application] /System/Library/Fr  
Die Variable kommazahl1 hat den Wert: 24.6567  
Der Text der Variablen txt lautet: Hallo 11 FOI 0
```



Datentypen Zusammenfassung

Datentyp	Art	Wertebereich/mögliche Werte	Größe	Beispiel
byte	Ganzzahl	-128 bis +127	8 Bit	byte x = 105;
short	Ganzzahl	-32.768 bis +32.767	16 Bit	short y = -64;
int	Ganzzahl	(ca.) -2 Milliarden bis +2 Milliarden	32 Bit	Int z = 45637;
long	Ganzzahl	(ca.) -10E18 bis +10E18	64 Bit	long l = 123343333;
float	Kommazahl	-3.4E+38 bis +3.4E+38	32 Bit	float f = 3.232;
double	Kommazahl	-1.7E+308 bis 1.7E+308	64 Bit	double d = -344.2322;
boolean	Wahrheitswert	true; false	1 Bit	boolean b = false;
char	Zeichen	alle Zeichen	8 Bit	char c = 'g';
String	Zeichenkette	Texte	unterschiedlich	String txt = "hallo";

Benennen von Variablen

- Erlaubt sind die Zeichen 'a' bis 'z', 'A' bis 'Z', 'o' bis 'g', der Unterstrich '_' und das Dollarzeichen '\$'.
- Ein Name darf keine Leerzeichen enthalten.
- Das erste Zeichen darf keine Zahl sein.
- Ein Name kann eine beliebige Länge haben.
- Groß- und Kleinbuchstaben zählen als *verschiedene* Zeichen.



Benennen von Variablen

- Ein Name kann kein reserviertes Wort sein.
- Ein Name darf nicht doppelt im selben Programmkontext vergeben werden.
- Benennen Sie Variablen so, dass klar ist wofür diese im Programm verwendet wird (z.B. `summe`; `produkt`).
- Java Variablen sollten mit einem kleinen Buchstaben beginnen klein.
- camelCase-Schreibweise: `diesIstEineVariable`



Strukturierte Programmierung

Rechnen



Rechenoperatoren

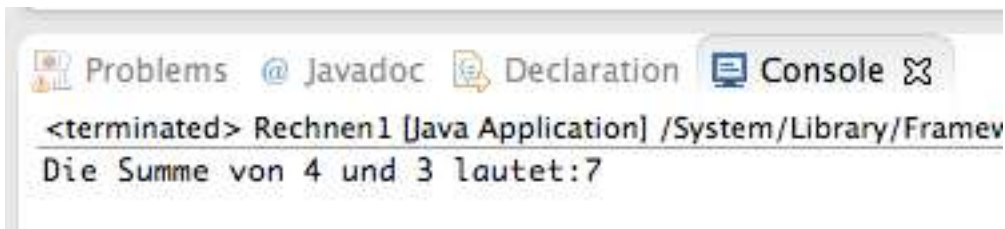
- Um in Java rechnen zu können gibt es u.a. die folgenden Operatoren:

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

- Zahlen und Variablen können in Rechnungen beliebig kombiniert werden.
- Das Rechenergebnis kann dann einer Variablen zugeordnet werden

Beispiel

```
public class Rechnen1 {  
    public static void main(String[] args) {  
  
        int zahl1 = 4;  
        int zahl2 = 3;  
        int ergebnis;  
  
        ergebnis = zahl1 + zahl2;  
  
        System.out.println("Die Summe von " + zahl1 + " und " + zahl2  
                            + " lautet:" + ergebnis);  
  
    }  
}
```



Beispiel

- Es soll der Wert an der Stelle $x = 7$ von folgender Funktion berechnet werden:

$$f(x) = \frac{x^2}{x+2}$$

```
public class Rechnen2 {  
    public static void main(String[] args) {  
        double x = 7.2;  
        double y;  
  
        y = (x * x) / (x + 2);  
  
        System.out.println("Das Ergebnis lautet:" + y);  
    }  
}
```



Strukturierte Programmierung

Eingabe über die Tastatur



Eingabe über die Tastatur

- Bislang müssen Werte für Variablen oder in Berechnungen fest im Programmcode angegeben werden.
- Nachteil: Soll z.B. eine Berechnung mit anderen Werten erfolgen, muss der Programmcode abgeändert werden.
- Sinnvoller ist es daher die Werte während dem Programmablauf über die Tastatur einzulesen.



Problem

- Es soll ein Programm mit folgenden Eigenschaften entwickelt werden:
 - Der Benutzer wird nach zwei Zahlen gefragt.
 - Das Programm rechnet die Differenz der beiden Zahlen aus und gibt das Ergebnis auf dem Bildschirm aus.

Beispiel 1 – Zahlen einlesen

```
import java.util.Scanner;

public class Eingabel {
    public static void main(String[] args) {

        int zahl1;
        int zahl2;

        Scanner eingabe = new Scanner( System.in );

        System.out.println("Geben Sie die erste Zahl ein:");
        zahl1 = eingabe.nextInt();

        System.out.println("Geben Sie die zweite Zahl ein:");
        zahl2 = eingabe.nextInt();

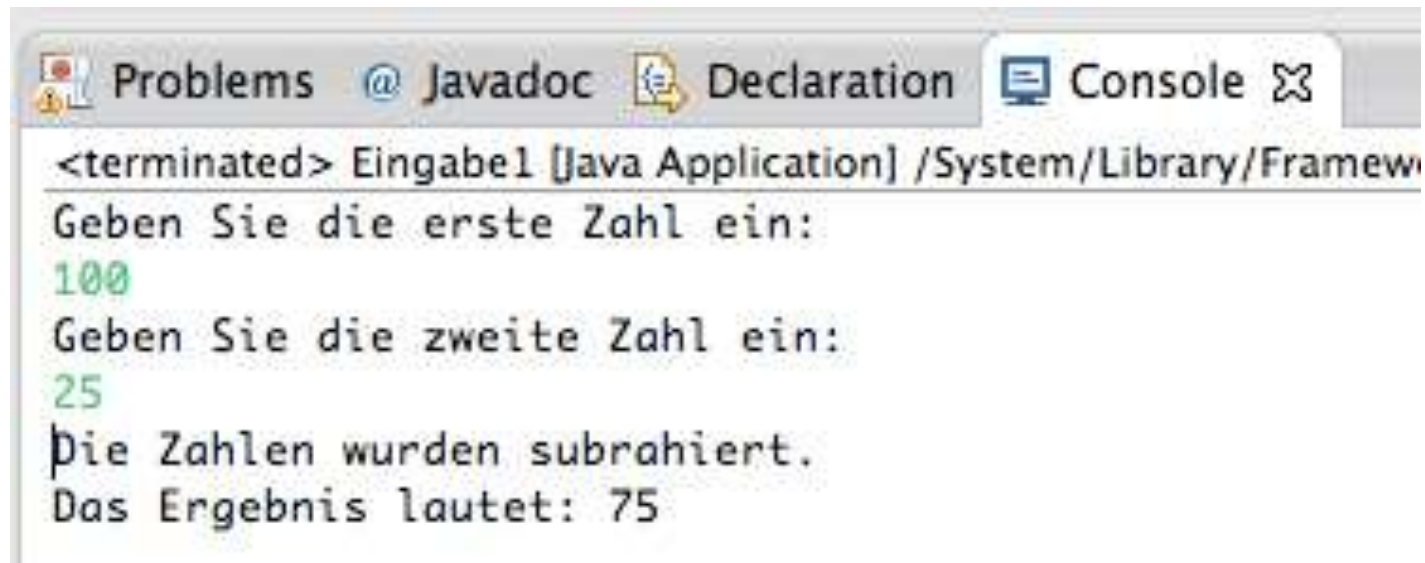
        int ergebnis = zahl1 - zahl2;

        System.out.println("Die Zahlen wurden subtrahiert.");
        System.out.println("Das Ergebnis lautet: "+ergebnis);

    }
}
```



Ausgabe



The screenshot shows a console window from an IDE. The title bar contains tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console output is as follows:

```
<terminated> Eingabe1 [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/CurrentJDK/Classes/
Geben Sie die erste Zahl ein:
100
Geben Sie die zweite Zahl ein:
25
Die Zahlen wurden subtrahiert.
Das Ergebnis lautet: 75
```

Beispiel 2 – Text einlesen

```
import java.util.Scanner;

class Eingabe2{
    public static void main (String[] args)  {
        String vorname;
        String nachname;

        Scanner eingabe = new Scanner( System.in );

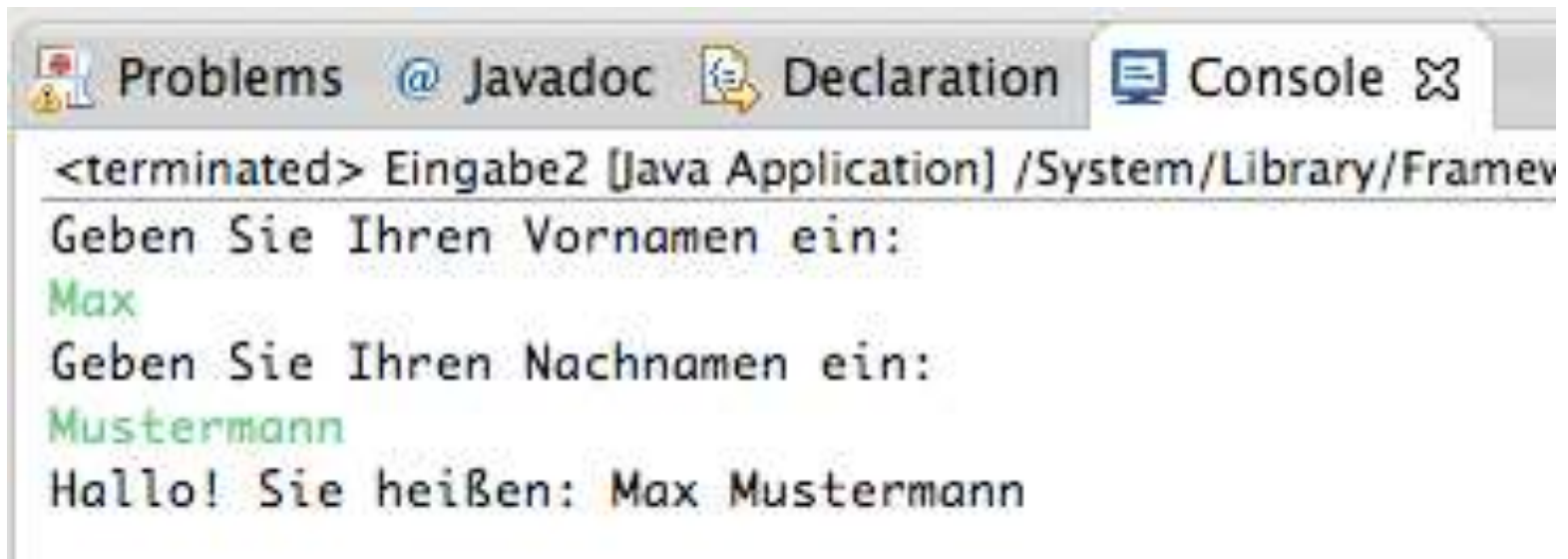
        System.out.println("Geben Sie Ihren Vornamen ein:");
        vorname = eingabe.nextLine();

        System.out.println("Geben Sie Ihren Nachnamen ein:");
        nachname = eingabe.nextLine();

        System.out.println("Hallo! Sie heißen: " + vorname + " "+nachname);
    }
}
```



Ausgabe



Strukturierte Programmierung

IF-Anweisung

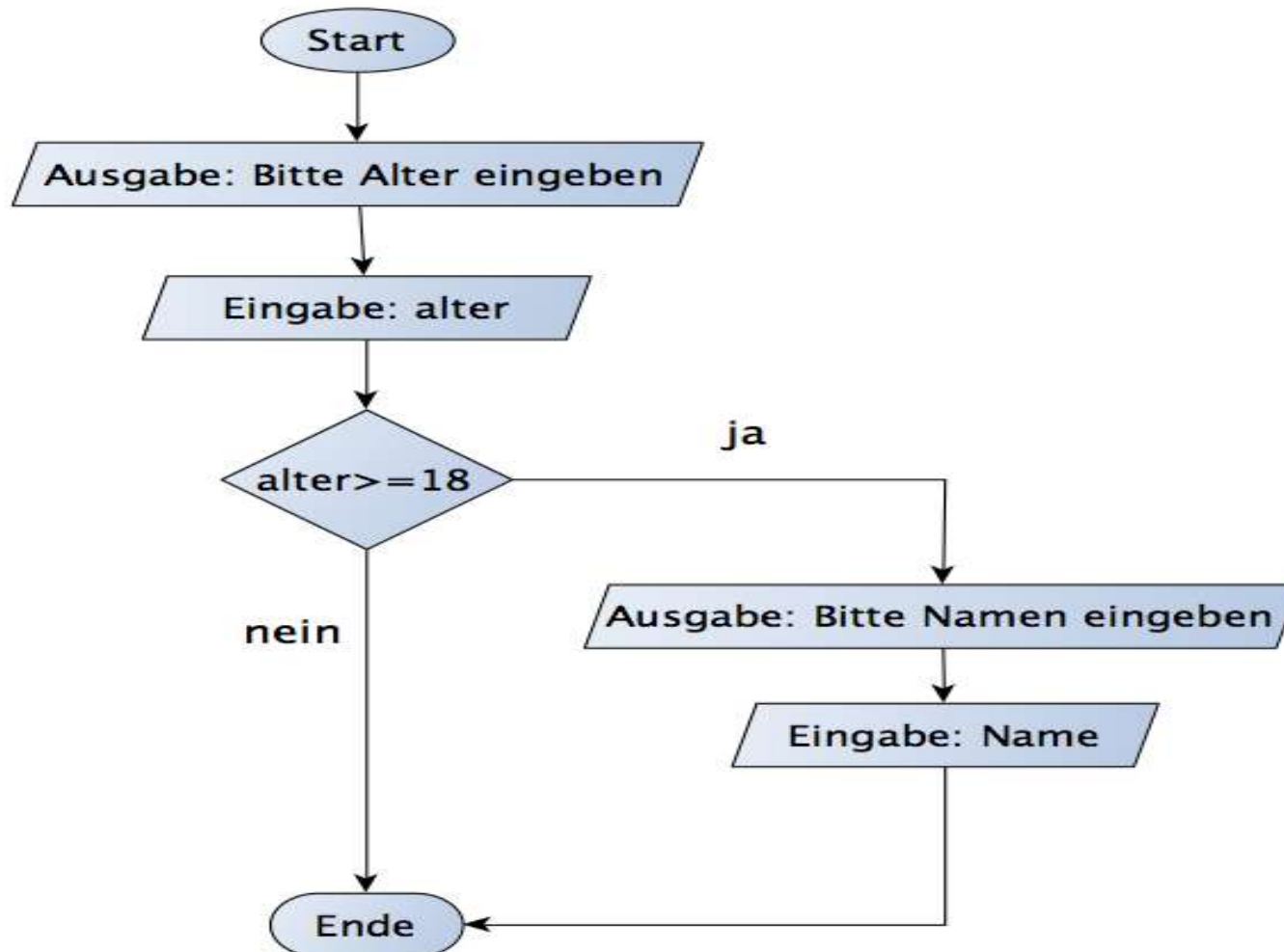


Beispiel

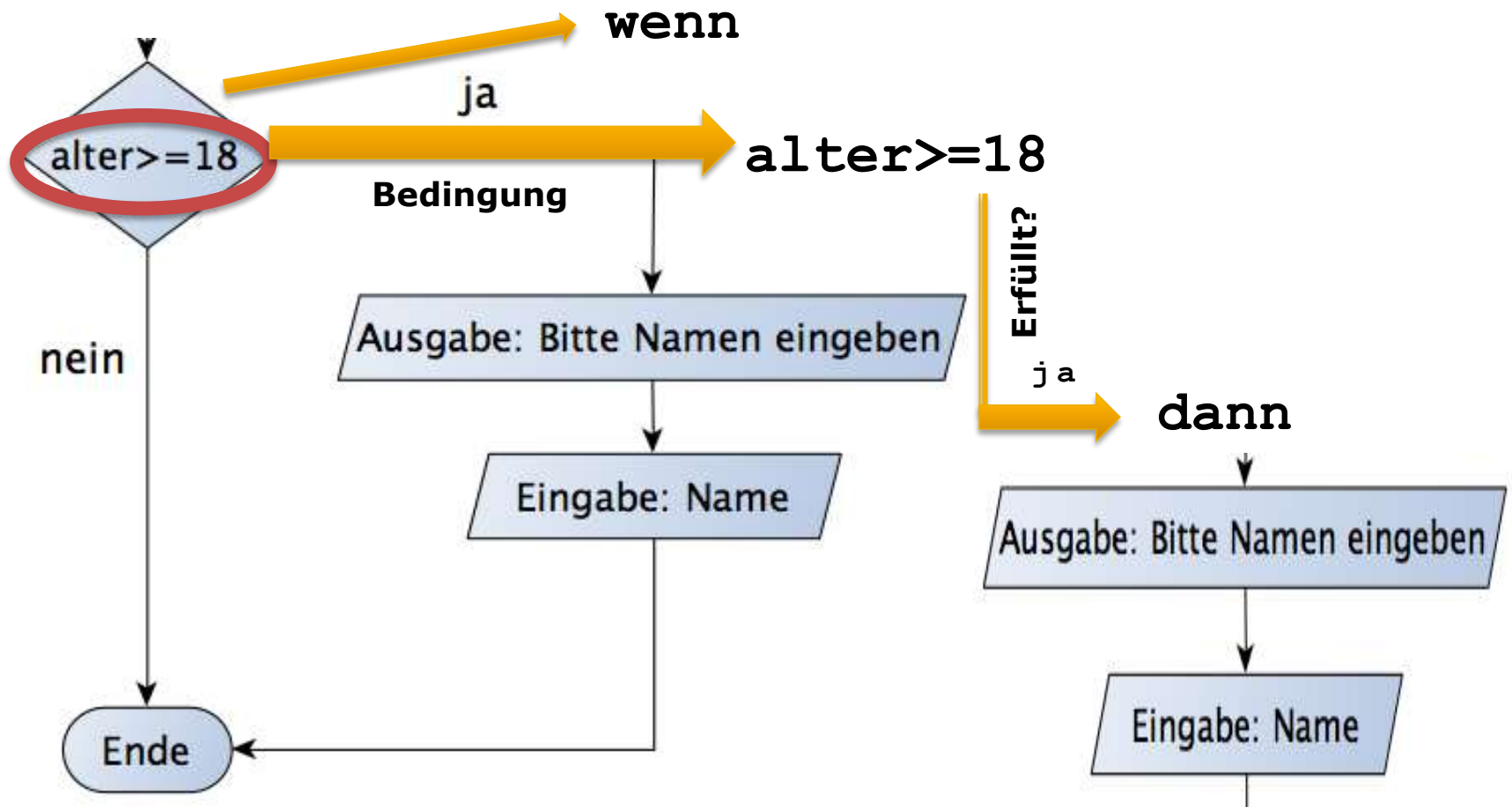
- Es soll folgendes Programm entwickelt werden:
 - Das Programm fragt nach dem Alter des Benutzers.
 - Ist der Benutzer volljährig ist, fragt das Programm auch nach seinen Namen. Ist er es nicht, geschieht nichts weiter.
- Problem: Das Ausführen eines Programmabschnitts (Eingabe Name) ist von einer Bedingung (Alter) abhängig.



Programmablaufplan



Umsetzung (Pseudocode)



Umsetzung (Pseudocode)

wenn

`alter >= 18`

Erfüllt?

dann

ja

Ausgabe: Bitte Namen eingeben

Eingabe: Name

```
if (alter >= 18) {
```

```
    /* Benutzer  
    * nach Namen  
    * fragen  
    */
```

```
}
```

Programmcode

```
Scanner eingabe = new Scanner(System.in);
int alter;
String name;

// Alter von Tastatur einlesen
System.out.println("Bitte geben Sie das Alter ein:");
alter=eingabe.nextInt();

if (alter>=18){
    System.out.println("Geben Sie Ihren Namen ein:");
    name=eingabe.nextLine();
}
```



Strukturierte Programmierung

IF – ELSE - Anweisung

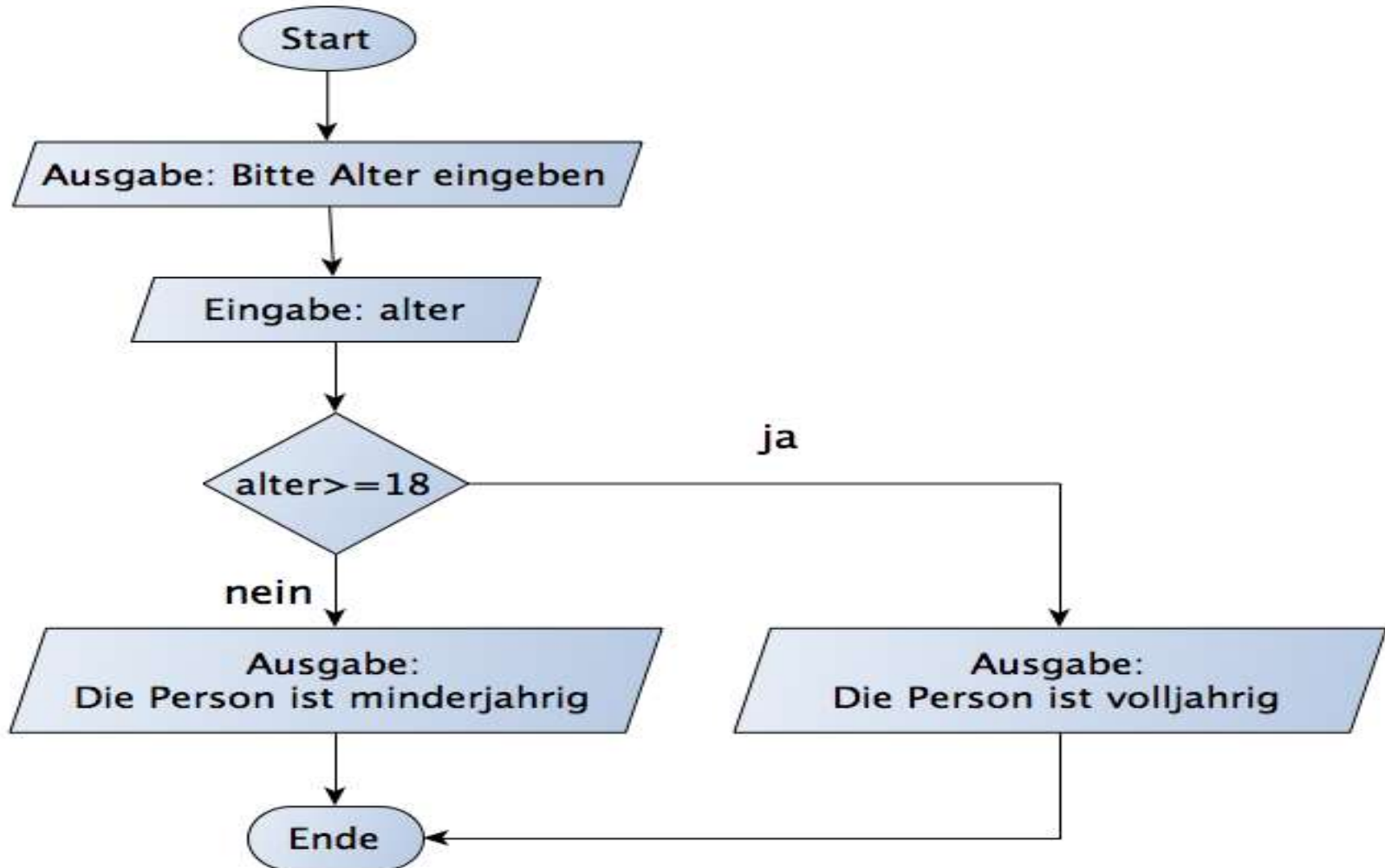


Beispiel

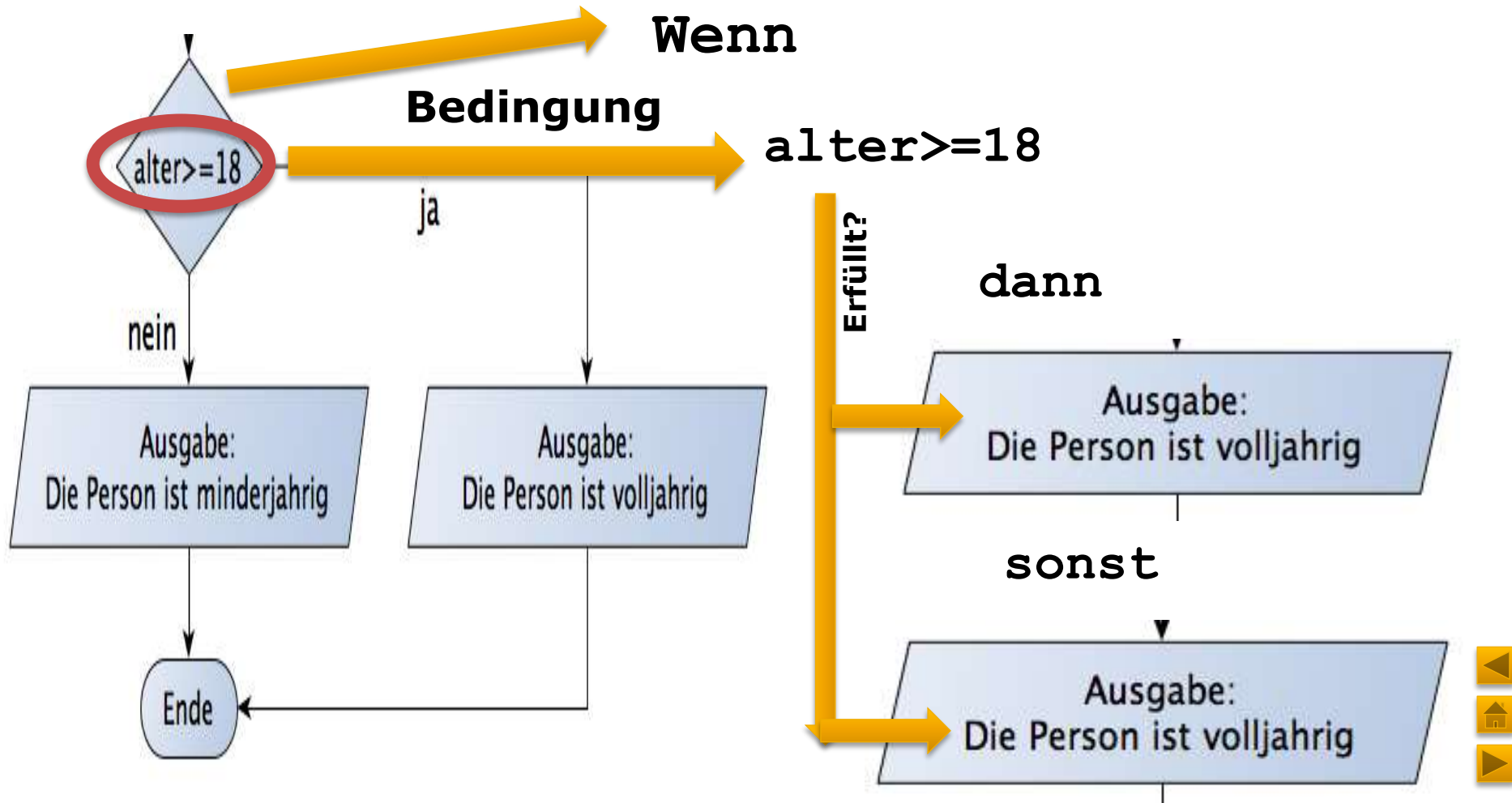
- Es soll eine Software mit folgenden Eigenschaften entwickelt werden:
 - Es soll das Alter einer Person von der Tastatur eingelesen werden.
 - Das Programm gibt auf dem Bildschirm aus, ob die Person volljährig oder minderjährig ist.
- Problem: Das Programm reagiert je nach eingegebenem Alter unterschiedlich.



Programmablaufplan



Umsetzung (Pseudocode)



Umsetzung (Pseudocode)

Wenn

`alter >= 18`

```
if (alter >= 18) {
```

Erfüllt?

dann

Ausgabe:
Die Person ist volljährig

```
/* Ausgabe:  
Schüler ist  
volljährig*/
```

sonst

Ausgabe:
Die Person ist minderjährig

```
} else {
```

```
/* Ausgabe:  
Schüler ist  
minderjährig*/
```

```
}
```

Umsetzung

```
public static void main(String[] args) {  
  
    Scanner eingabe = new Scanner(System.in);  
    int alter;  
  
    // Alter von Tastatur einlesen  
    System.out.println("Bitte geben Sie das Alter ein:");  
    alter=eingabe.nextInt();  
  
    if (alter>=18){  
        System.out.println("Schüler ist volljährig");  
    }else{  
        System.out.println("Schüler ist minderjährig");  
    }  
}
```



Strukturierte Programmierung

IF - Mehrfachverzweigung

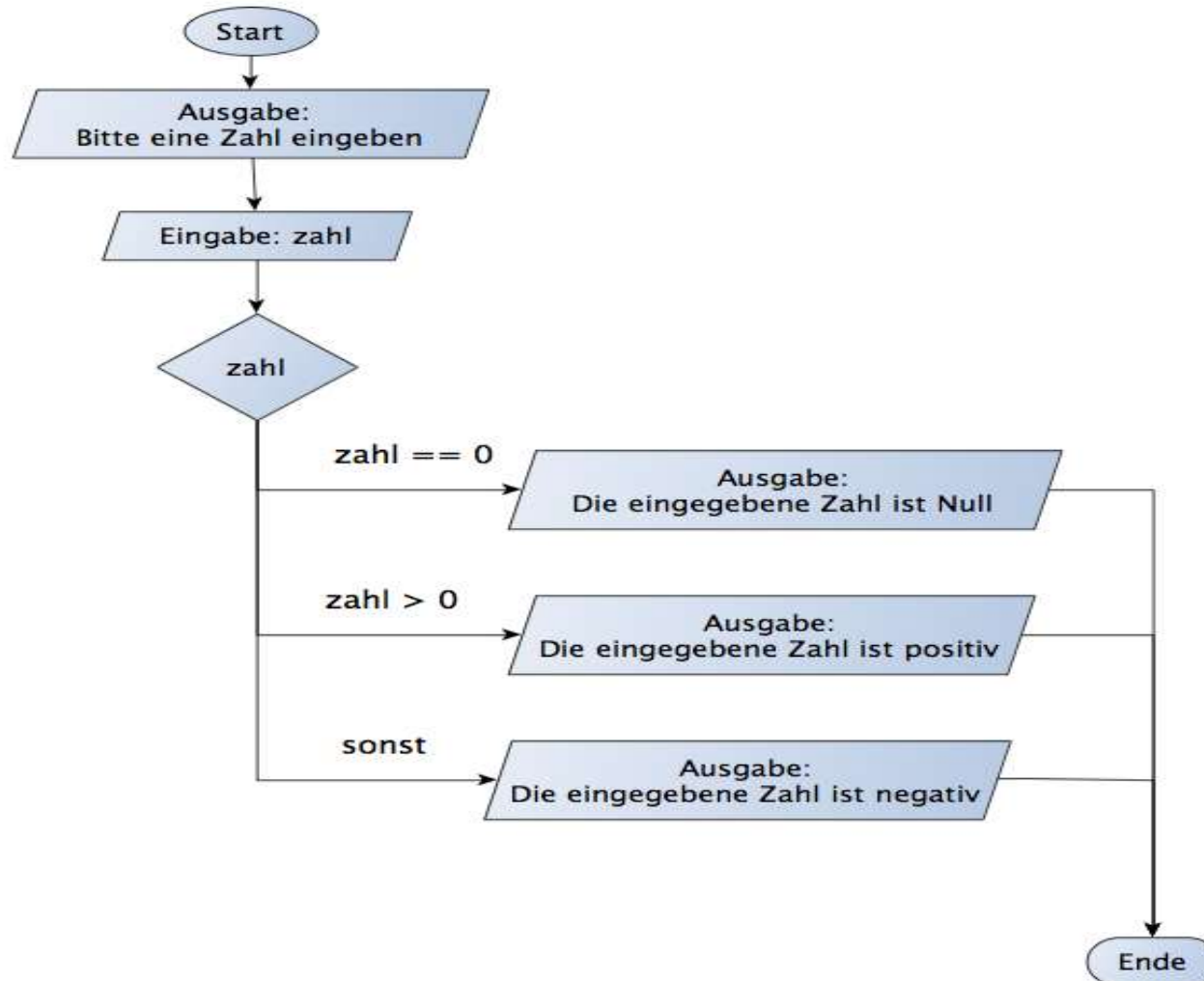


Beispiel

- Es soll ein Programm entwickelt werden, welches den Benutzer nach einer Zahl fragt.
- Das Programm testet die Zahl und gibt auf dem Bildschirm aus, ob die Zahl „positiv“, „negativ“ oder „Null“ ist.
- Problem: Es müssen mehrere Bedingungen geprüft werden ($\text{zahl} > 0$, $\text{zahl} < 0$, $\text{zahl} \text{ gleich } 0$).
- Lösung: Mehrfachverzweigung



Programmablaufplan



Programmcode

```
int zahl = eingabe.nextInt();

if (zahl==0){

    System.out.println(„Die eingegebene zahl ist Null.“);

}else if (zahl>0){

    System.out.println(„Die eingegebene zahl ist positiv.“);

}else{

    System.out.println(„Die eingegebene Zahl ist negativ.“);

}
```



Strukturierte Programmierung

IF – Anweisung: UND, ODER, NICHT

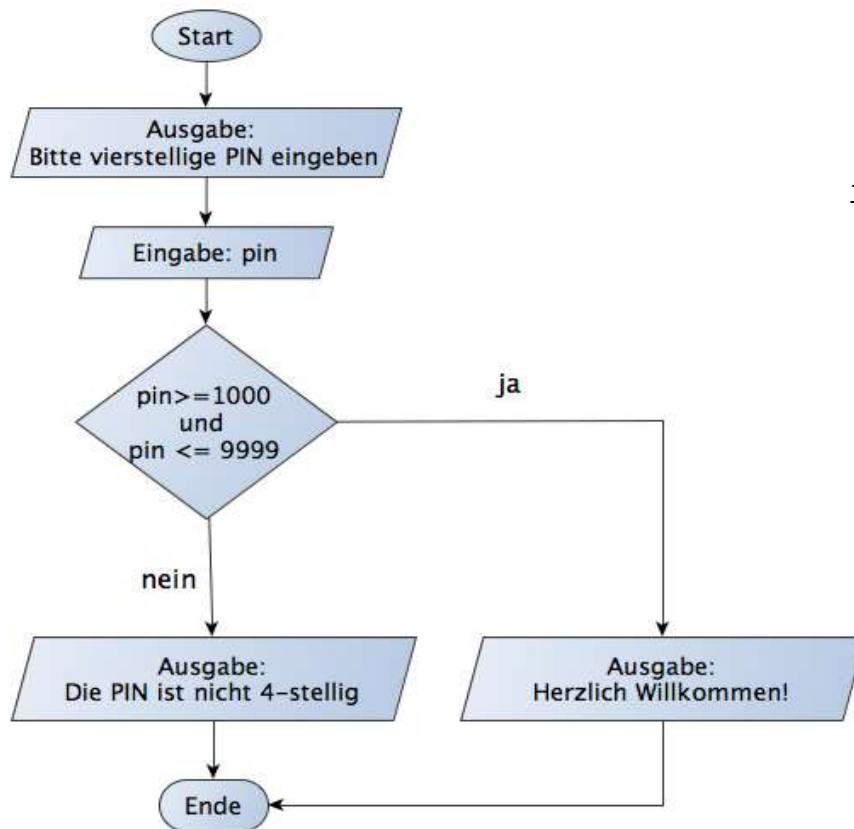


Beispiel

- Für einen Bankautomat soll eine 4-Stellige PIN eingegeben werden.
- Sie erhalten den Auftrag ein Programm zu entwickeln, welches nach dieser PIN fragt. Nur wenn die PIN 4 – stellig ist, soll eine Willkommensmeldung ausgegeben werden; sonst eine Fehlermeldung.
- Lösungssatz: Der PIN ist 4-stellig, wenn er größer gleich 1000 und kleiner gleich 9999 ist.



PAP



```
if (PIN>=1000 && PIN<=9999) {  
    System.out.println(  
        „Herzlich Willkommen!“  
    )  
}else{  
    System.out.println(  
        „Fehler: Die PIN ist  
        nicht 4-Stellig“);  
}
```

Weitere Beispiele

IF-Bedingung	Bedeutung
<code>if (zahl1 > 0 zahl2 <10) ...</code>	Wenn zahl1 größer 0 oder zahl 2 kleiner 10 ist, dann ...
<code>if(!zahl1>0) ...</code>	Wenn zahl1 nicht größer Null ist, dann ...
<code>if(zahl1==4)...</code>	Wenn zahl1 gleich 4 ist, dann ...
<code>if (zahl1 > 500 && zahl1 <520 && zahl1 != 515)...</code>	Wenn zahl1 zwischen 500 und 520 liegt, aber nicht 515 ist, dann ...

Strukturierte Programmierung

switch-case-Anweisung



Beispiel

- Es soll ein Programm geschrieben werden, welches den Benutzer nach einer Zahl zwischen 0 und 3 fragt.
- Das Programm gibt anschließend die eingegebene Zahl als Wort (Null, Eins, Zwei, Drei) aus.
- Wird eine Zahl eingegeben, die nicht zwischen 0 und 3 liegt, erscheint eine Fehlermeldung.



Lösung mit IF-ELSE

```
System.out.println("Geben Sie eine Zahl zwischen 0 und 3 ein:");
zahl=eingabe.nextInt();

if (zahl==0){
    System.out.println("Sie haben die Null eingegeben.");
}else if (zahl == 1){
    System.out.println("Sie haben die Eins eingegeben.");
}else if (zahl == 2){
    System.out.println("Sie haben die Zwei eingegeben.");
}else if (zahl == 3){
    System.out.println("Sie haben die Drei eingegeben.");
}else{
    System.out.println("Die Zahl liegt nicht zwischen 0 und 3");
}
```



switch-case

```
System.out.println("Geben Sie eine Zahl zwischen 0 und 3 ein:");  
zahl=eingabe.nextInt();
```

```
switch(zahl) {  
case 0:  
    System.out.println("Sie haben die Null eingegeben.");  
    break;  
case 1:  
    System.out.println("Sie haben die Eins eingegeben.");  
    break;  
case 2:  
    System.out.println("Sie haben die Zwei eingegeben.");  
    break;  
case 3:  
    System.out.println("Sie haben die Drei eingegeben.");  
    break;  
default:  
    System.out.println("Die Zahl liegt nicht zwischen 0 und 3.");  
}
```



Strukturierte Programmierung

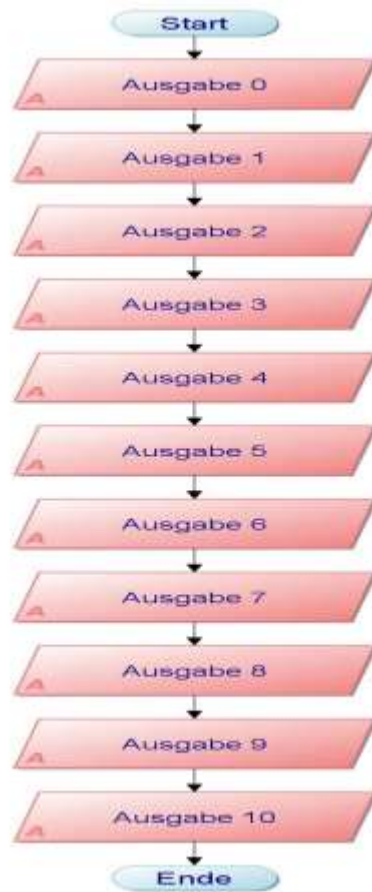
Schleifen



Beispiel 1

- Es soll ein Programm geschrieben werden, dass von 0 bis 10 zählt und die Zahlen auf dem Bildschirm ausgibt.

Ansatz 1

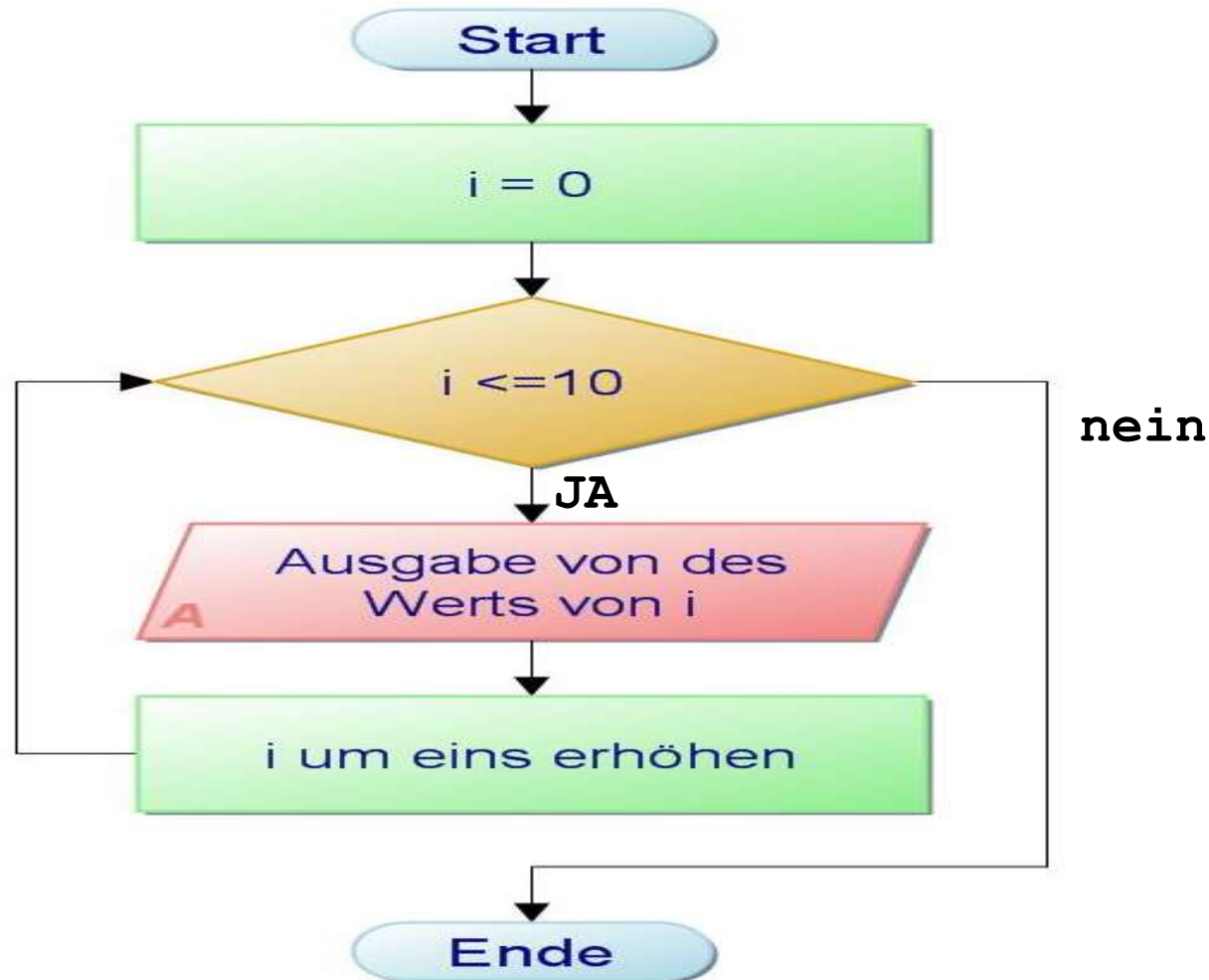


```
System.out.println("0");  
System.out.println("1");  
System.out.println("2");  
System.out.println("3");  
System.out.println("4");  
System.out.println("5");  
System.out.println("6");  
System.out.println("7");  
System.out.println("8");  
System.out.println("9");  
System.out.println("10");
```

Bewertung

- Umständlich
- Code doppelt sich
- aufwendig anzupassen (es soll bis 100, 1000, 100000 gezählt werden)
- Unflexibel
- Idee: wenig Programmcode doppeln, dafür den Programmcode mehrfach ausführen

Darstellung als Verzweigung



Darstellung mit Schleifensymbol

S c h l e i f e

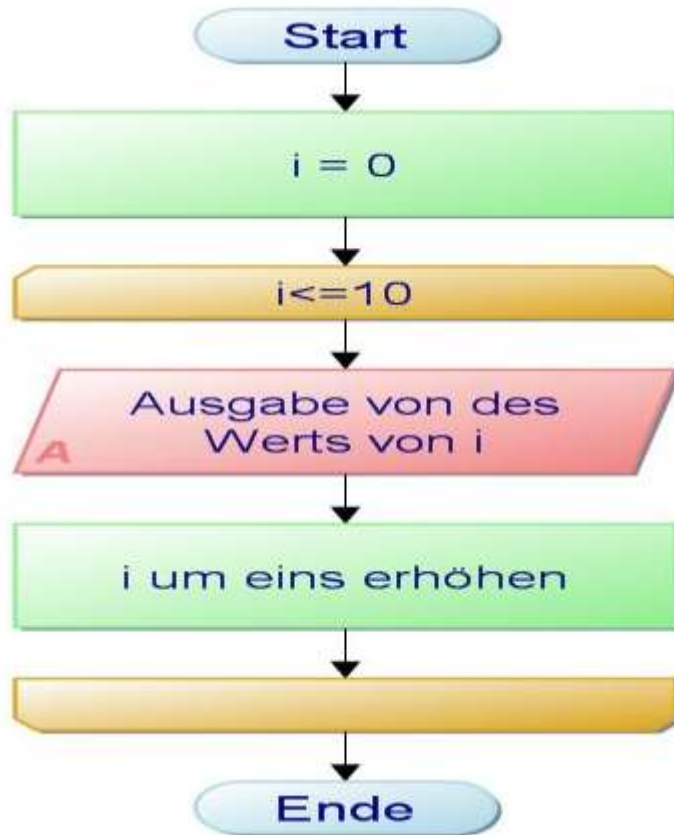


Zählvariable

Schleifen-Bedingung

Die Schleife wird solange wiederholt, wie die Schleifenbedingung erfüllt ist. Diese Überprüfung findet hier am Anfang der Schleife statt. Innerhalb der Schleife wird die Zählvariable i auf dem Bildschirm ausgegeben und deren Wert anschließend um eins erhöht.

Umsetzung



```
int i = 0;
```

```
while (i <= 10) {
```

```
    System.out.println(i);
```

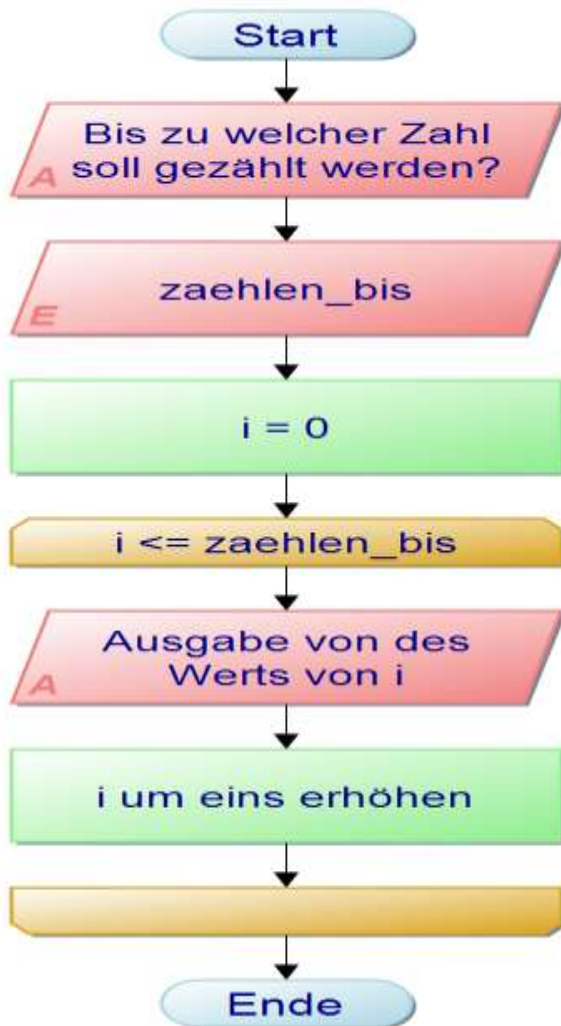
```
    // i um eins erhöhen:  
    i = i + 1;
```

```
}
```

Beispiel

- Es soll ein Programm entwickelt werden, dass den Benutzer fragt, bis zu welcher Zahl gezählt werden soll.
- Das Programm gibt dann alle Zahlen von 0 bis zur eingegebenen Zahl auf dem Bildschirm aus.

Umsetzung



```
System.out.println(«Bis zu welcher  
Zahl soll gezählt  
werden?»);
```

```
int zaehlen_bis=eingabe.nextInt();
```

```
int i = 0;
```

```
while (i <= zaehlen_bis) {
```

```
    System.out.println(i);
```

```
    // i um eins erhöhen:
```

```
    i = i + 1;
```

```
}
```


Weitere Beispiele

- Ein Programm summiert Zahlen der Reihe nach auf.
- Beispiel: Der Benutzer gibt 5 ein, das Programm berechnet:
 $1+2+3+4+5 = 25$

```
System.out.println("Bis zu welcher  
Zahl soll wollen Sie die Summe  
bilden?");
```

```
int summe_bis = eingabe.nextInt();
```

```
int i = 1;  
int summe = 0;
```

```
while (i <= summe_bis) {  
    // i auf die bisherige  
    // Summe addieren:  
    summe = summe + i;  
  
    // i um eins erhöhen:  
    i = i + 1;  
}
```

```
System.out.println("Die Summe  
ist:"+summe);
```



Weiterer Beispiele

- Ein Programm wird solange wiederholt, wie es der Benutzer möchte:

```
int nochmal = 1;

while (nochmal==1) {

    System.out.println("Hallo!");
    System.out.println("nochmal?");
    System.out.println("1 für ja");

    nochmal = eingabe.nextInt();

}
```

Strukturierte Programmierung

for - Schleifen



For - Schleifen

- Die meisten Schleifen
 - brauchen eine Zählvariable,
 - benötigen eine Schleifenbedingung, die abhängig ist von der Zählvariable,
 - verändern die Zählvariable in jedem Schleifendurchlauf um einen festen Wert (z.B. plus 1).

Beispiel „typische“ Schleife

```
int i = 0;
while (i <= zaehlen_bis) {
    System.out.println(i);

    // i um eins erhöhen:
    i = i + 1;
}
```

Zählvariable

Schleifenbedingung

Zählvariable ändern



Idee:

Die drei „Bestandteile“ werden im Schleifenkopf zusammen gefasst.



For-Schleife

```
int i = 0;
```

```
while (i <= zaehlen_bis) {
```

```
    System.out.println(i);
```

```
    // i um eins erhöhen:  
    i = i + 1;
```

```
}
```

```
for (int i=0; i<=zaehlen_bis; i=i+1){
```

```
    System.out.println(i);
```

```
}
```

Anmerkung: statt `i = i + 1` können Sie auch `i++` schreiben

Strukturierte Programmierung

Arrays



Beispiel

- Es soll ein Programm entwickelt werden, welches 10 ganzzahlige Messwerte speichern soll.
- Die Messwerte sollen auf dem Bildschirm ausgegeben werden.

1. Anzahl

```
// 10 Variablen anlegen
// und Werte zuweisen
int zahl1 = 2;
int zahl2 = 34;
int zahl3 = -23;
int zahl4 = 9;
int zahl5 = -8;
int zahl6 = -45;
int zahl7 = 233;
int zahl8 = 56;
int zahl9 = 765;
int zahl10 = -34;
```

```
// Ausgabe der Werte
System.out.println(zahl1);
System.out.println(zahl2);
System.out.println(zahl3);
System.out.println(zahl4);
System.out.println(zahl5);
System.out.println(zahl6);
System.out.println(zahl7);
System.out.println(zahl8);
System.out.println(zahl9);
System.out.println(zahl10);
```

Array

zahlenliste:

0	1	2	3	4	5	6	7	8	9
2	34	-23	9	-8	-45	233	55	765	-34

```
int zahlenliste[]={2, 34, -23, 9, -8, -45, 233, 55 , 765, -34}
```

Zugriff auf Array

Wert aus Array auf dem Bildschirm ausgeben:

zahlenliste:	0	1	2	3	4	5	6	7	8	9
	2	34	-23	9	-8	-45	233	55	765	-34

```
// Die Programmzeile  
System.out.println(zahlenliste[6]);  
// gibt den Wert 233 auf dem Bildschirm aus
```

Wert im Array ändern

vorher

zahlenliste:	0	1	2	3	4	5	6	7	8	9
	2	34	-23	9	-8	-45	233	55	765	-34

```
zahlenliste[4] = 1111;
```

nachher

zahlenliste:	0	1	2	3	4	5	6	7	8	9
	2	34	-23	9	1111	-45	233	55	765	-34

Alle Werte im Array ausgeben

```
int zahlenliste[]=  
{2, 34, -23, 9, -8, -45, 233, 55 , 765, -34}  
  
for (int i = 0; i<10; i++){  
  
    System.out.println(zahlenliste[i]);  
  
}
```



Leeres Array füllen (Tastatur)

```
// leeres Array mit 10 Einträgen erzeugen
int zahlenliste[]=new int[10];

// alternativ
int n = 10;
int zahlenliste[]=new int[n];

// Zehn Werte von Tastatur einlesen
// und im Array speichern
for (int i = 0; i<10; i++){

    zahlenliste[i]=eingabe.nextInt();

}
```



Strukturierte Programmierung

foreach-Schleife



Schleifen über Arrays

- Schleifen „über Arrays“, in welchen jedes Element aus einem Array der Reihe nach herausgenommen und verarbeitet wird sind sehr häufig, z.B.:

```
// zahlenliste ist ein bereits  
// mit int - Werten gefülltes Array  
  
for (int i = 0; i<zahlenliste.length; i++){  
    int wert = zahlenliste[i];  
    System.out.println();  
}
```

- Für dieses Schema gibt es die foreach-Schleife.
- Die Schleife verwendet in JAVA ebenfalls das Schlüsselwort „for“.
- Der Schleifenkopf hat aber einen anderen Aufbau.



Foreach-Schleife

```
// „normale“ Schreibweise  
// mit int - Werten gefülltes Array  
  
for (int i = 0; i<zahlenliste.length; i++){  
    int wert = zahlenliste[i];  
    System.out.println(wert);  
}
```



```
// „for-each“ Schreibweise  
  
for (int wert:zahlenliste){  
    System.out.println(wert);  
}
```

Hole der Reihe nach
die Werte aus dem int-Array.

Und speichere die Werte
in der int Variablen „wert“

Wäre zahlenliste vom Typ double:
double wert:zahlenliste



Strukturierte Programmierung

Datei Ein- und Ausgabe



Beispiel 1

- Eine Textdatei soll bis zum Dateiende zeilenweise eingelesen werden.
- Der Inhalt wird dabei auf dem Bildschirm ausgegeben.

Datei einlesen

```
import java.io.*;

class ReadFile2
{
    public static void main(String[] args) throws IOException
    {
        FileReader fr = new FileReader("test.txt");
        BufferedReader br = new BufferedReader(fr);

        String zeile = "";

        do
        {
            zeile = br.readLine();
            System.out.println(zeile);
        }
        while (zeile != null);

        br.close();
    }
}
```



Beispiel 2

- In eine Datei soll ein Text zeilenweise geschrieben werden.

Schreiben in eine Datei

```
import java.io.*;

class WriteFile
{
    public static void main(String[] args) throws IOException
    {
        FileWriter fw = new FileWriter("ausgabe.txt");
        BufferedWriter bw = new BufferedWriter(fw);

        bw.write("test test test");
        bw.newLine();
        bw.write("tset tset tset");

        bw.close();
    }
}
```



Beispiel

- Es sollen Zahlen aus einer Datei gelesen und verarbeitet werden.
- Die verarbeiteten Zahlen werden dann in eine Datei geschrieben.
- Problem: Die Zahlen können nur als Text (String) gelesen werden. Eine Verarbeitung als Zahl ist so nicht möglich. Ebenso können Sie Zahlen nur als Text schreiben.
- Lösung: Die Zahl als String muss in einen Integer (oder double) umgewandelt werden.
- Für das Schreiben muss die Zahl in einen String konvertiert werden.



Konvertierungen

```
// Text in ganze Zahl umwandeln

String txt1 = „1234“;

int zahl1 = Integer.parseInt(txt1);

// Zahl in Text umwandeln

int zahl2 = 5678;

String txt2;

txt2 = Integer.toString(zahl2);
```

Anmerkung: Sie müssen dazu zunächst die Klasse Integer importieren (import java.lang.Integer).



Strukturierte Programmierung

Methoden



Beispiel 1

- Es soll ein Programm geschrieben werden, welches die Fläche eines Kreises ausrechnet. Der Radius des Kreises soll dabei von der Tastatur eingelesen werden.

Umsetzung (Struktogramm)

Ausgabe: Dieses Programm berechnet die Fläche eines Kreises
Ausgabe: Bitte geben den Radius ein
Eingabe: Radius r
Fläche = $3,14 * r * r$
Ausgabe: Fläche

Umsetzung (JAVA Code)

```
import java.util.*;

public class kreisflaeche1 {
    public static void main(String[] args) {

        Scanner eingabe = new Scanner (System.in);

        System.out.println("Dieses Programm berechnet die Flaeche eines Kreises.");
        System.out.println("Geben Sie den Radius r des Kreises ein:");
        double r = eingabe.nextDouble();

        double flaeche = 3.14*r*r; //

        System.out.println("Die Flaeche betraegt:"+flaeche);

    }
}
```



Beispiel

- Das Programm soll die Fläche von zwei Kreisen berechnen.

Umsetzung (Struktogramm)

Ausgabe: Dieses Programm berechnet die Fläche eines Kreises
Ausgabe: Bitte geben den Radius ein
Eingabe: Radius r
$\text{Fläche} = 3,14 * r * r$
Ausgabe: Fläche
Ausgabe: Bitte geben den Radius ein
Eingabe: Radius r
$\text{Fläche} = 3,14 * r * r$
Ausgabe: Fläche



Umsetzung (JAVA Code)

```
import java.util.Scanner;

public class kreisflaeche2 {

    public static void main(String[] args) {

        Scanner eingabe = new Scanner (System.in);
        System.out.println("Dieses Programm berechnet die Flaeche eines Kreises.");
        System.out.println("Geben Sie den Radius r des Kreises ein:");
        double r = eingabe.nextDouble();
        double flaeche = 3.14*r*r; //

        System.out.println("Die Flaeche betraegt:"+flaeche);

        System.out.println("Dieses Programm berechnet die Flaeche eines Kreises.");
        System.out.println("Geben Sie den Radius r des Kreises ein:");
        r = eingabe.nextDouble();
        flaeche = 3.14*r*r; //
        System.out.println("Die Flaeche betraegt:"+flaeche);

    }
}
```



Bewertung der Lösung

- Code-Dopplung
- Unübersichtlich
- Nicht wieder verwendbar (Kreisberechnung)
 - Idee: Kreisberechnung auslagern in eigenen Programmteil und immer wieder aufrufen.
 - Ein wiederaufrufbarer Programmteil heißt Methode.
 - Für eine Methode muss ein Name definiert werden, unter dem sie aufgerufen werden kann.



Umsetzung (Struktogramm)

Hauptprogramm:

Ausgabe: Dieses Programm berechnet die Fläche von 2 Kreisen	
	kreisflächeBerechnen()
	kreisflächeBerechnen()

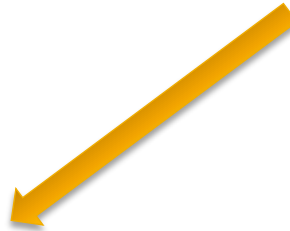
Methode

kreisflaecheBerechnen() :

Ausgabe: Bitte geben den Radius ein
Eingabe: Radius r
Fläche = $3,14 * r * r$
Ausgabe: Fläche

Methode ohne Parameter

Methodenname: Unter diesem Namen
Wird die Methode im Programmcode
aufgerufen



```
static void kreisflaecheBerechnen() {  
  
    Scanner eingabe = new Scanner (System.in);  
    System.out.println("Geben Sie den Radius r ein:");  
    double r = eingabe.nextDouble();  
    double flaeche = 3.14*r*r; //  
    System.out.println("Die Flaeche betraegt:"+flaeche+"\n");  
  
}
```



Gesamtcode

```
public class kreisflaeche3 {  
  
    static void kreisflaecheBerechnen(){  
        Scanner eingabe = new Scanner (System.in);  
        System.out.println("Geben Sie den Radius r des Kreises ein:");  
        double r = eingabe.nextDouble();  
        double flaeche = 3.14*r*r; //  
        System.out.println("Die Flaeche betraegt:"+flaeche+"\n");  
    }  
  
    public static void main(String[] args) {  
  
        System.out.println("Dieses Programm berechnet  
                            die Flaeche eines Kreises.");  
  
        kreisflaeche3.kreisflaecheBerechnen();  
        kreisflaeche3.kreisflaecheBerechnen();  
  
    }  
}
```



Strukturierte Programmierung

Methoden mit Parameter



Beispiel

- Bei dem Programm zur Kreisberechnung soll folgendes geändert werden:
 - Die Methode soll nicht selbst nach einem Wert des Radius fragen. Der Wert für den Radius soll der Methode im Main-Teil übergeben werden.
- Werte können in Methoden mit Parametern übergeben werden.
- Parameter brauchen einen Datentyp und einen Namen



Methode mit Parameter

Parameter

Datentyp des Parameters

Parametername

```
static void kreisflaecheBerechnen(double r) {  
  
    double flaeche = 3.14*r*r;   
    System.out.println("Die Flaeche ist:"+flaeche);  
}
```

Zugriff auf Wert des Parameters

Gesamtcode

```
public class kreisflaeche4 {  
  
    static void kreisflaecheBerechnen(double r){  
        double flaeche = 3.14*r*r; //  
        System.out.println("Die Flaeche betraegt:"+flaeche+"\n");  
    }  
  
    public static void main(String[] args) {  
  
        // Methode mit festem Wert aufrufen  
        kreisflaeche4.kreisflaecheBerechnen(5);  
  
        // Methode mit einem von der Tastatur eingegeben Wert aufrufen  
        Scanner eingabe = new Scanner (System.in);  
        System.out.println("Geben Sie den Radius r des Kreises ein:");  
        double radius=eingabe.nextDouble();  
        kreisflaeche4.kreisflaecheBerechnen(radius);  
  
    }  
}
```



Beispiel 2

- Es soll eine Methode bereitgestellt werden, welche die Fläche eines Rechtecks berechnet.
- Die beiden Seitenlängen sollen als Parameter der Methode übergeben werden.
- Es werden zwei Parameter benötigt (Seite A und Seite B)

Programmcode

erster Parameter

zweiter Parameter

```
public class Rechteck {  
  
    static void fleacheBerechnen(double a, double b) {  
        double flaeche = a * b;  
        System.out.println("Die Flaeche ist: " + flaeche);  
    }  
  
    public static void main(String[] args) {  
        // Flaeche berechnen  
        Rechteck.fleacheBerechnen(2, 3);  
    }  
}
```



Strukturierte Programmierung

Methoden mit Rückgabewert

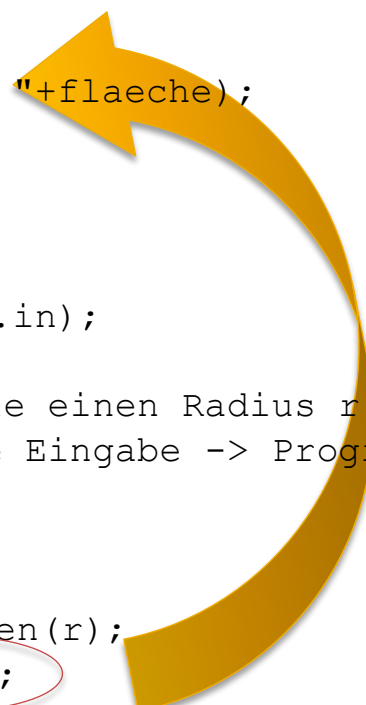


Beispiel

- Es soll ein Programm entwickelt werden, welches die Flächen von beliebig vielen Kreisen berechnet.
- Die Kreisfläche soll innerhalb einer Methode berechnet werden.
- Die Kreisflächen sollen aufsummeriert werden.

Ansatz

```
static void kreisflaecheBerechnen(double r) {  
    double flaeche = 2 * 3.14 * r;  
    System.out.println("Die Flaeche ist: "+flaeche);  
}  
  
public static void main(String[] args) {  
    double r = 0; double summe = 0;  
    Scanner eingabe = new Scanner(System.in);  
    do {  
        System.out.println("Geben Sie einen Radius r ein:");  
        System.out.println("negative Eingabe -> Programmende");  
        r = eingabe.nextDouble();  
        // Flaeche berechnen  
        if (r >= 0) {  
            kreisflaecheBerechnen(r);  
            summe = summe + ???;  
        }  
    } while (r >= 0);  
}
```



Problem: Wie kommt das
Hauptprogramm an die berechnete
Fläche?

Rückgabewerte

- Auf Variablen innerhalb einer Methode kann nicht direkt zugegriffen werden.
- Um Werte nach außen zu geben, muss die Methode „antworten“.
- Die Methode braucht einen Rückgabewert.
- Statt void wird der Typ des Rückgabewertes angegeben.
- Die eigentliche Rückgabe erfolgt mit „return“.



Methode mit Rückgabewert



Datentyp des Rückgabewertes

```
static double kreisflaecheBerechnen(double r) {  
    double flaeche = 2 * 3.14 * r;  
    return flaeche;  
}
```



Rückgabe des Wertes in der Variablen „flaeche“

Gesamtcode

```
Static double kreisflaecheBerechnen(double r) {  
    double flaeche = 2 * 3.14 * r;  
    return flaeche;  
}  
  
public static void main(String[] args) {  
    double r = 0; double summe = 0;  
    Scanner eingabe = new Scanner(System.in);  
    do {  
        System.out.println("Geben Sie einen Radius r ein:");  
        System.out.println("negative Eingabe -> Programmende");  
        r = eingabe.nextDouble();  
        // Flaeche berechnen  
        if (r >= 0) {  
            double teilflaeche = kreisflaecheBerechnen(r);  
            summe = summe + teilflaeche;  
        }  
    } while (r >= 0);  
}
```

„Auffangen“ des Rückgabewertes



Strukturierte Programmierung

Arrays als Parameter



Beispiel 1

- Im Hauptprogramm soll ein Array mit Werten angelegt werden.
- Es soll eine Methode „wertImArrayAusgeben“ bereitgestellt werden, welche ein Array als Parameter erhält.
- Die Methode gibt den Inhalt des Arrays auf dem Bildschirm aus



Methode wertImArrayAusgeben

```
static void wertImArrayAusgeben(int zahlen[]) {  
    System.out.println("Werte im Array:");  
    for (int i = 0; i<zahlen.length;i++){  
        System.out.println(zahlen[i]);  
    }  
}
```

Übergabe eines Arrays mit ganzen Zahlen. Wird innerhalb der Methode mit dem Namen „zahlen“ angesprochen.

Array fragen, wie „lang“, d.h. wie groß es ist.

Zahl im Array an Position i ausgeben.



Komplettes Programm

```
public class array1 {  
  
    static void werteImArrayAusgeben(int zahlen[]){  
  
        System.out.println("Werte im Array:");  
        for (int i = 0; i<zahlen.length;i++){  
            System.out.println(zahlen[i]);  
        }  
  
    }  
  
    public static void main(String[] args) {  
        int zahlenliste[]={1,-234,5,-4,23}; // Array anlegen  
        werteImArrayAusgeben(zahlenliste);  
  
    }  
  
}
```



Beispiel 2

- Im Hauptprogramm soll ein leeres Array mit z.B. 10 Einträgen angelegt werden.
- Es soll eine Methode „arrayMitWerteFuellen“ bereitgestellt werden. Der Methode wird ein Array übergeben.
- Die Methode füllt dann das Array mit Werten, die von der Tastatur eingelesen werden.



Methode

```
static void arrayMitWerteFuellen(int zahlenl[]){  
  
    Scanner eingabe = new Scanner (System.in);  
  
    // Es wird vorausgesetzt, dass das Array  
    // vorher erzeugt wurde  
    for (int i = 0; i<zahlen.length;i++){  
        System.out.println("Geben Sie einen Wert ein:");  
        zahlen[i]=eingabe.nextInt();  
    }  
  
}
```



Komplettes Programm

```
public class arrays {  
  
    static void arrayMitWerteFuellen(int zahlen[]){  
        Scanner eingabe = new Scanner (System.in);  
        for (int i = 0; i<zahlen.length;i++){  
            System.out.println("Geben Sie einen Wert ein:");  
            zahlen[i]=eingabe.nextInt();  
        }  
    }  
  
    public static void main(String[] args) {  
        int zahlenliste[]=new int[5]; // Array mit 5 Eintraegen einlesen  
        arrayMitWerteFuellen(zahlenliste);  
        //Beispielwert ausgeben  
        System.out.println(zahlenliste[4]);  
    }  
}
```



Strukturierte Programmierung

Stringverarbeitung



Strings vergleichen

- Zwei Strings sollen miteinander verglichen werden, ob sie denselben Wert besitzen.

Ansatz 1

```
public class string_vergleich {  
  
    public static void main(String[] args) {  
  
        String str1=new String("hallo");  
        String str2=new String("hallo");  
  
        if (str1 == str2){  
            System.out.println("gleich");  
        }else{  
            System.out.println("nicht gleich");  
        }  
  
    }  
}
```



Auswertung

- Problem: Ansatz 1 erzeugt immer die Ausgabe "nicht gleich".
- Strings sind keine elementaren Datentypen, sondern Objekte im Speicher.
- str1 und str2 haben zwar **denselben Inhalt** "hallo", sind aber **verschiedene Objekte** im Speicher.
- str1 == str2 prüft, ob es **dieselben Objekte** sind (was nicht der Fall ist).
- Es wird eine Möglichkeit benötigt, die **Inhalte** zu vergleichen.



Ansatz 2

```
public class strvergleich {  
    public static void main(String[] args) {  
  
        String str1=new String("hallo");  
        String str2=new String("hallo");  
  
        if (str1.equals(str2)){  
            System.out.println("gleich");  
        } else{  
            System.out.println("nicht gleich");  
        }  
    }  
}
```



Liefert das gewünschte Ergebnis.



Strings lexikographisch vergleichen

- Zwei Strings sollen miteinander verglichen werden, um festzustellen, welcher String im Alphabet vorher kommt.

Lexikographisch vergleichen

```
public class strvergleich {  
    public static void main(String[] args) {  
  
        String str1=new String("Berta");  
        String str2=new String("Sommer");  
  
        // funktioniert nicht:  
        if (str1 < str2) {  
            System.out.println(str1 + "steht vor " +str2);  
        }  
  
        // Vergleich, ob str1 vor str 2 steht  
        if (str1.compareTo(str2)<0) {  
            System.out.println(str1 + "steht vor " +str2);  
        }  
  
        // Vergleich, ob str1 hinter str 2 steht  
        if (str1.compareTo(str2)>0) {  
            System.out.println(str1+"steht hinter"+str2);  
        }  
  
    }  
}
```



Strings verketten

- Mehrere Strings sollen zu einem String zusammengebaut werden.
- Dies geschieht mit dem „+“-Operator.
- Er wird im Prinzip wie die Addition bei Zahlen verwendet, nur dass die Strings verkettet werden.

Strings verketten

```
public class stringsverketten {  
    public static void main(String[] args) {  
  
        String str1="Hallo";  
        String str2="Dies";  
        String str3="ist";  
        String str4="ein";  
        String str5="Test";  
  
        String satz = str1+"! "+str2+" "+str3+" "+str4+" "+str5+".";  
  
        System.out.println(satz);  
  
    }  
}
```



Problems



@ Javadoc



Declaration



Console



```
<terminated> stringsverketten [Java Application] /Library/Java/JavaVir  
Hallo! Dies ist ein Test.
```



Strukturierte Programmierung

ArrayListen



Wozu?

- In ArrayListen können wie in „normalen“ Arrays viele Werte vom selbem Datentyp gespeichert werden.
- Unterschiede:
 - normale Arrays sind **statisch**: d.h. sie werden mit einer festen Größe angelegt. Die Größe kann dann nicht mehr geändert werden.
 - ArrayListen sind **dynamisch**: d.h. sie haben keine feste Größe. Die Listen wachsen automatisch mit.
 - ArrayListen bieten weitere Methoden an (z.B. zum Löschen von Einträgen).



Vergleich: Erzeugen eines Arrays / einer ArrayList

„NORMALES“ ARRAY

```
// Array mit ganzen Zahlen  
erzeugen:
```

```
int zahlenListe[] =  
new int[10];
```

ARRAYLIST

```
// ArrayList erzeugen:
```

```
ArrayList<Integer> zahlenListe  
= new ArrayList<Integer>() ;
```

```
// Anmerkung statt int muss  
Integer verwendet werden  
// es wird keine Größe  
Angaben (dynamisch)
```

Vergleich: Mit Werten füllen

ARRAY

```
/* Array mit zehn Werten  
füllen */  
  
for (int i = 0; i < 10; i++) {  
  
    // einzelne Zahl eingeben  
    System.out.println("Zahl  
    eingeben:");  
  
    int zahl = eingabe.nextInt();  
  
    // Zahl im Array speichern  
    zahlenListe[i] = zahl;  
  
}
```

ARRAYLIST

```
/* ArrayListe mit zehn Werten  
füllen */  
  
for (int i = 0; i < 10; i++) {  
  
    // einzelne Zahl eingeben  
    System.out.println("Zahl  
    eingeben:");  
  
    int zahl = eingabe.nextInt();  
  
    // Zahl im Array speichern  
    zahlenListe.add(zahl);  
  
}
```

Vergleich: Alle Werte ausgeben

ARRAY

```
// Array ausgeben

for (int i = 0; i < 10; i++) {

    // einzelne Zahl an Position i
    herausholen

    int zahl_i = zahlenListe[i];

    // Zahl ausgeben:
    System.out.println("Zahl an
    Position " + i + " " +
    zahl_i);

}
```

ARRAYLIST

```
// ArrayList ausgeben

for (int i = 0; i < 10; i++) {

    // einzelne Zahl an Position i
    herausholen

    int zahl_i =
zahlenListe.get(i);

    // Zahl ausgeben:
    System.out.println("Zahl an
    Position " + i + " " +
    zahl_i);

}
```

Weitere Beispiele für ArrayListen

```
// Kommazahlen  
ArrayList<Double> zahlenListe =  
new ArrayList<Double>();
```

```
zahlenListe.add(4.5);
```

```
// Texte  
ArrayList<String> textListe =  
new ArrayList<String>();
```

```
textListe.add("Hallo");
```

```
/* es können auch „eigene Datentypen“ bzw. Objekte  
verwaltet werden -> siehe nächstes Kapitel */
```



Elemente in ArrayListen löschen

`remove(int index)`

Entfernt das im Array gespeicherte Objekt an übergebenen Position aus der Liste.

Beispiel:

```
meineListe.remove(5);
```

Entfernt das 5. Objekt in der Liste. Die nachfolgenden Elemente der Liste rücken dann nach oben.



Wie viele Einträge hat eine ArrayList?

```
int size()
```

Gibt die Anzahl der Elemente in der ArrayList als int-Wert zurück.

Beispiel:

```
System.out.println("Die ArrayList enthält "  
+ meineListe.size() + " Elemente.")
```



Strukturierte Programmierung

Eigene Datentypen: Objekte



Beispiel

- Es soll eine Software zur Verwaltung von Telefonnummern entwickelt werden.
- Ein Eintrag für eine Telefonnummer besteht aus der Telefonnummer und dem Vor- und Nachnamen der zugehörigen Person.
- Die Daten sollen von der Tastatur eingelesen werden.

Bisheriger Ansatz

// **Bisheriger Ansatz:** Für jede Eigenschaft wird eine Variable angelegt

```
public class telefonbuch1 {  
    public static void main(String[] args) {  
  
        Scanner eingabe = new Scanner(System.in);  
  
        String name;  
        String vorname;  
        int nummer;  
  
        System.out.println("Geben Sie einen Namen ein:");  
        name = eingabestr.nextLine();  
        System.out.println("Geben Sie einen Vorname ein:");  
        vorname = eingabestr.nextLine();  
        System.out.println("Geben Sie eine Telefonnummer ein:");  
        nummer = eingabezahlen.nextInt();  
  
    }  
}
```



Nachteile

- Die einzelnen Variablen, welche einen Telefonbucheintrag beschreiben, haben nicht „direkt“ miteinander „etwas zu tun“.
- Die Variablen könnten im gesamten Programmcode „verstreut“ sein - > unübersichtlich/unstrukturiert
- Will man viele Telefonbucheinträge speichern, braucht man viele Arrays (für jede eines).



Idee

```
public class telefonbuch1 {  
    public static void main(String[] args)
```

```
        Scanner eingabe = new Scanner(System.in);
```

```
        String name;  
        String vorname;  
        int nummer;
```

```
        System.out.println("Geben Sie den Namen ein");  
        name= eingabestr.nextLine();  
        System.out.println("Geben Sie das Vorname ein");  
        vorname = eingabestr.nextLine();  
        System.out.println("Geben Sie die Nummer ein");  
        nummer = eingabezahlen.nextInt();
```

```
    }  
}
```

Die Variablen „name“, „vorname“ und „nummer“ gehören zusammen und bilden einen Telefonbucheintrag.
Idee: Zusammengehörige Variablen zu einem neuen Datentyp (Objekt) „zusammenpacken“.



Telefonbucheintrag als eigener Datentyp

**Zusammenfassung der Daten als neuer eigener Datentyp.
Der Datentyp wird als Klasse in einer eigenen JAVA Datei definiert:**

```
public class Telefonbucheintrag {  
    String name;  
    String vorname;  
    int nummer;  
}
```

Aus dieser Klasse werden dann Objekte erzeugt, welche dann verwendet werden können.



Telefonbucheintrag als eigener Datentyp

Name des neuen Datentyps / der Klasse

```
public class Telefonbucheintrag {  
    String name;  
    String vorname;  
    int nummer;  
}
```

Attribute oder Eigenschaften des neuen Datentyps (beliebige Variablen).

Wichtig: der Datentyp als neue Klasse wird in einer eigenen JAVA-Datei definiert (hier: Telefonbucheintrag.java)



Umsetzung

```
public class telefonbuch2 {  
    public static void main(String[] args) {  
        Scanner eingabe = new Scanner(System.in);  
  
        // einen neuen einzelnen Telefonbucheintrag erzeugen  
        // Ein konkretes Objekt mit Namen tb_ein aus der Klasse  
        // Telefonbucheintrag erzeugen:  
        Telefonbucheintrag tb_ein = new Telefonbucheintrag();  
  
        System.out.println("Geben Sie einen Namen ein:");  
        // Auf das Attribut name im Objekt tb_ein zugreifen:  
        tb_ein.name= eingabe.nextLine();  
        System.out.println("Geben Sie einen Vorname ein:");  
        // auf das Attribute vorname zugreifen:  
        tb_ein.vorname = eingabe.nextLine();  
        System.out.println("Geben Sie eine Telefonnummer ein:");  
        // und die nummer:  
        tb_ein.nummer = eingabe.nextInt();  
    }  
}
```

Wichtig: Telefonbucheintrag.java und telefonbuch2.java müssen sich im selben Projekt befinden.



Vorteile

- Logisch zusammenhängende Variablen bilden eine Einheit.
- Benutzerdefinierte Datentypen können in Bibliotheken gespeichert und damit wieder verwendet werden.
- Das „Handling“ im Zusammenhang mit Array wird vereinfacht (siehe nächste Folien).



Beispiel 2

- Es sollen beliebig viele Telefonbucheinträge verwaltet werden.
- Zu Beginn gibt der Benutzer ein, wie viele Einträge eingegeben werden sollen.
- Anschließend werden exakt so viele Einträge eingegeben.
- Die Telefonbucheinträge sollen in einem Array mit Namen Telefonbuch gespeichert werden:

```
Telefonbucheintrag telefonbuch[] =  
new Telefonbucheintrag[anzahl]
```



Ansatz mit Objekt und Array

```
Scanner eingabestr = new Scanner(System.in);
Scanner eingabezahlen = new Scanner(System.in);

System.out.println("Wie viele Adressen sollten eingegeben werden?");
int anzahl = eingabezahlen.nextInt();

// Array anlegen mit Telefonbucheinträgen anlegen:
Telefonbucheintrag telefonbuch[]=new Telefonbucheintrag[anzahl];

for (int i = 0; i < anzahl; i++) {
    // einzelnen Telefonbucheintrag erzeugen und einlesen
    Telefonbucheintrag tb_ein = new Telefonbucheintrag();
    System.out.println("Geben Sie einen Namen ein:");
    tb_ein.name = eingabestr.nextLine();
    System.out.println("Geben Sie einen Vorname ein:");
    tb_ein.vorname = eingabestr.nextLine();
    System.out.println("Geben Sie eine Telefonnummer ein:");
    tb_ein.nummer = eingabezahlen.nextInt();
    // Telefonbucheintrag in das Array "stecken"
    telefonbuch[i]=tb_ein;
}
```



Ansatz mit Objekt und ArrayList

```
Scanner eingabestr = new Scanner(System.in);
Scanner eingabezahlen = new Scanner(System.in);

System.out.println("Wie viele Adressen sollten eingegeben werden?");
int anzahl = eingabezahlen.nextInt();

// ArrayListe anlegen - nicht vergessen die ArrayListe zu importieren
ArrayList<Telefonbucheintrag> telefonbuch = new ArrayList<Telefonbucheintrag>();

for (int i = 0; i < anzahl; i++) {
    // einzelnen Telefonbucheintrag erzeugen und einlesen
    Telefonbucheintrag tb_ein = new Telefonbucheintrag();
    System.out.println("Geben Sie einen Namen ein:");
    tb_ein.name = eingabestr.nextLine();
    System.out.println("Geben Sie einen Vorname ein:");
    tb_ein.vorname = eingabestr.nextLine();
    System.out.println("Geben Sie eine Telefonnummer ein:");
    tb_ein.nummer = eingabezahlen.nextInt();
    // Telefonbucheintrag in das Array "stecken"
    telefonbuch.add(tb_ein);
}
```



Strukturierte Programmierung

Zeichenfunktionen

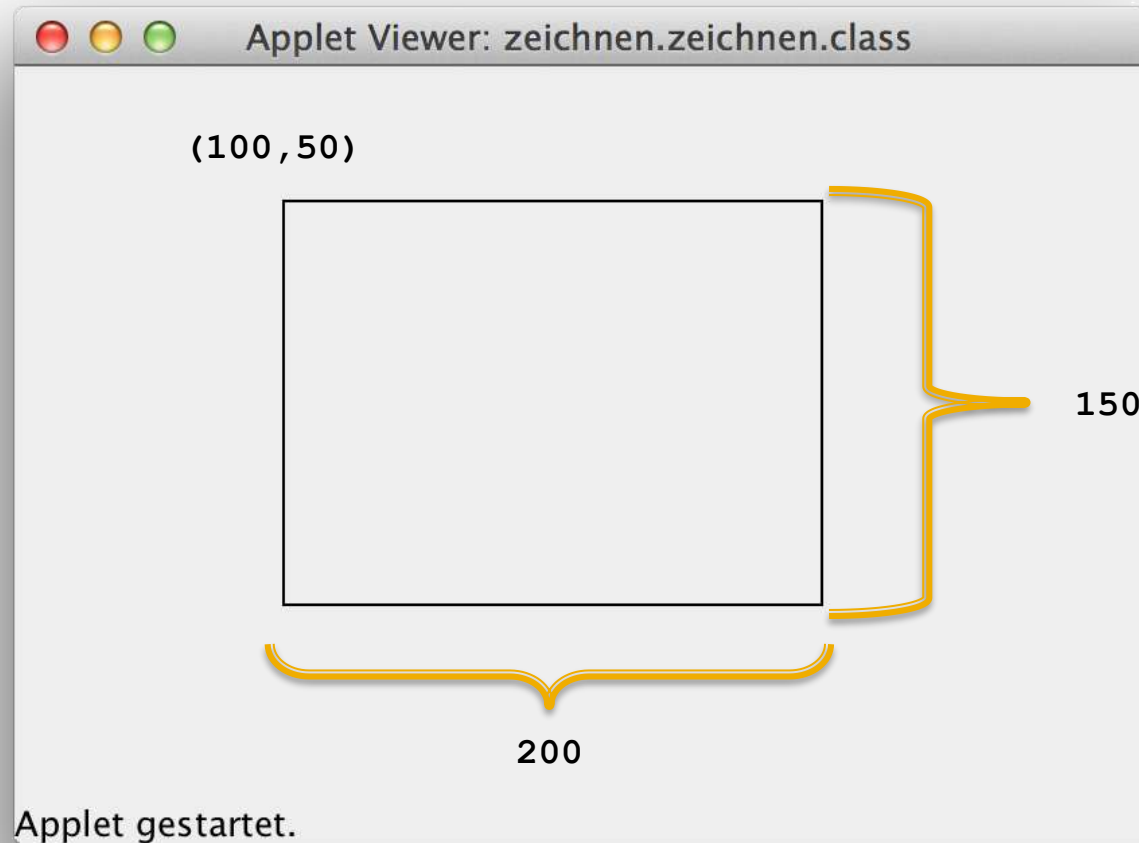


Zeichnen

```
import java.applet.Applet;  
import java.awt.Color;  
import java.awt.Graphics;  
  
public class zeichnen extends Applet {  
    public static void main(String[] args) {  
  
    }  
  
    public void paint(Graphics g) {  
        // drawRect (x,y,breite,hoehe)  
        g.drawRect(100, 50, 200, 150);  
  
    }  
  
}
```



Ausgabe



Farbe

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Graphics;

public class zeichnen2 extends Applet {
    public static void main(String[] args) {

    }

    public void paint(Graphics g) {
        // TODO Auto-generated method stub
        g.setColor(Color.GREEN);
        g.fillRect(10, 20, 200, 150);

    }

}
```



Fenstergröße und Eingabe

```
public class zeichnen3 extends Applet {
    int x;
    int y;
    public static void main(String[] args) {

    }
    public void init() {
        Scanner ein = new Scanner(System.in);
        this.setSize(800,600);
        System.out.println("x:");
        x=ein.nextInt();
        System.out.println("y:");
        y=ein.nextInt();
        ein.close();
        super.init();
    }

    public void paint(Graphics g) {
        // TODO Auto-generated method stub
        g.drawRect(x, y, 200, 150);
    }

}
```



Weitere Zeichenfunktionen

Weitere Zeichenfunktion: siehe <http://docs.oracle.com/javase/7/docs/api/> Klasse Graphics

java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
Graphics
Graphics2D
GraphicsConfigTemplate
GraphicsConfiguration
GraphicsDevice
GraphicsDevice.WindowTr
GraphicsEnvironment
GrayFilter
GregorianCalendar
GridBagConstraints
GridBagLayout
GridBagLayoutInfo
GridLayout
Group
GroupLayout
GroupLayout.Alignment

Component, clipRect(int, int, int, int), setColor(java.awt.Color),
setPaintMode(), setXORMode(java.awt.Color), setFont(java.awt.Font)

Constructor Summary

Constructors

Modifier	Constructor and Description
protected	Graphics() Constructs a new Graphics object.

Method Summary

Methods

Modifier and Type	Method and Description
abstract void	clearRect (int x, int y, int width, int height) Clears the specified rectangle by filling it with the background color of the current drawing surface.
abstract void	clipRect (int x, int y, int width, int height) Intersects the current clip with the specified rectangle.
abstract void	copyArea (int x, int y, int width, int height,



**Autor:**

Christian Frye

Version:

1.8.1

Datum:

20.08.2020

Web:

<http://moodle2.erasmus-kittler-schule.de>

<http://www.eks-darmstadt.de>

Mail:

christian.frye@erasmus-kittler-schule.de

