Java-Workshop
Sommerpraktikum

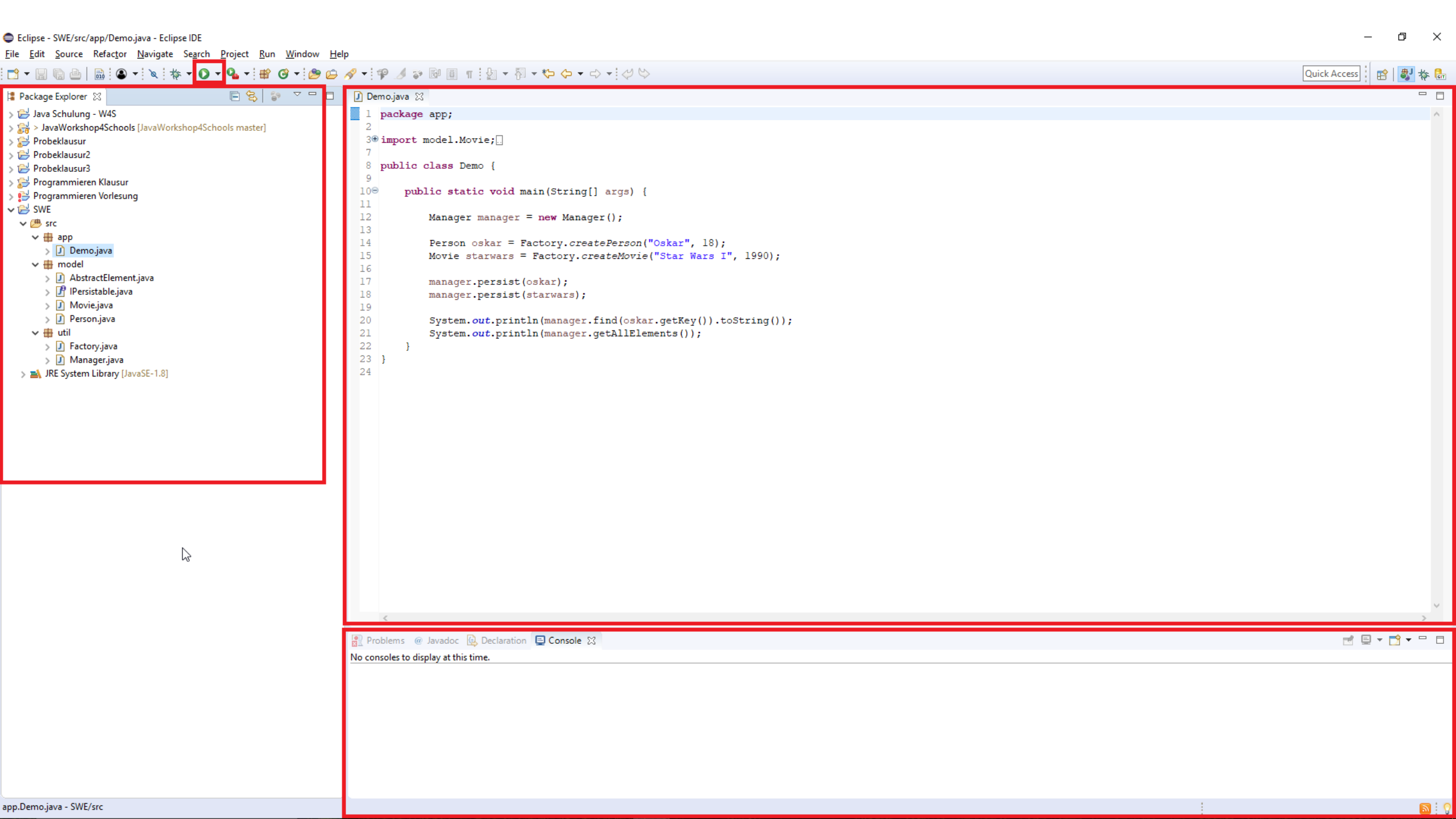
Jonas & Patrick
August 05, 2020

PUBLIC

Agenda

- Einstieg
- Datentypen & Variablen
- Operatoren
- Kontrollstrukturen
- Methoden
- Input & Output

Crash Course Eclipse


Eclipse - SWE/src/app/Demo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
Java Schulung - W4S
JavaWorkshop4Schools [JavaWorkshop4Schools master]
Probeklausur
Probeklausur2
Probeklausur3
Programmieren Klausur
Programmieren Vorlesung
SWE
src
app
Demo.java
model
AbstractElement.java
IPersistable.java
Movie.java
Person.java
util
Factory.java
Manager.java
JRE System Library [JavaSE-1.8]
Demo.java
1 package app;
2
3 import model.Movie;
7
8 public class Demo {
9
10 public static void main(String[] args) {
11
12 Manager manager = new Manager();
13
14 Person oskar = Factory.createPerson("Oskar", 18);
15 Movie starwars = Factory.createMovie("Star Wars I", 1990);
16
17 manager.persist(oskar);
18 manager.persist(starwars);
19
20 System.out.println(manager.find(oskar.getKey()).toString());
21 System.out.println(manager.getAllElements());
22 }
23 }
24
Problems Javadoc Declaration Console
No consoles to display at this time.
app.Demo.java - SWE/src

Grundlegende Java Syntax

- Jede Anweisung endet mit einem Semikolon (;)
- Anweisungsblöcke sind von geschweiften Klammen ({ }) umgeben
- Konsolenausgabe: `System.out.println("Meine Ausgabe");`
  - „syso/sysout“ +Strg + Leertaste
- Programmstart mit der Methode `main(String[] args){};`
- Dateien werden in Paketen organisiert und gesammelt

Übung: Hello World

```
~ › Hello world!
```

Schreibe ein Java-Programm, welches „Hello World!“ in der Konsole ausgibt!

```
~ › Hello Max!
```

Modifiziere das Programm, sodass es deinen Namen oder einen beliebigen Text ausgibt!

Lösung: Hello World

Die Klassendefinition und main-Methode werden bei weiteren Lösungen weggelassen!

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Datentypen & Variablen

Was sind Datentypen?

- Ziele von Datentypen:
  - Zusammenfassung konkreter Wertebereiche und darauf definierten Operationen zu einer Einheit
  - Beispiel: Ganze Zahlen als Datentyp „Integer“
- Typisierung:
  - Einschränkung des Wertebereichs von Variablen
  - Beispiel: „Integer“ (Datentyp für ganze Zahlen), erlaubt sind nur ganze Zahlen, keine Kommazahlen

- Elementare (z.B. Integer) vs. zusammengesetzte Datentypen (z.B. String)
  - Elementardatentypen:
    - festgelegter Wertebereich (z.B. 0 bis 127 bei Zahlen)
    - Bestimmte Operationen sind fest definiert (z.B. elementare Rechenoperationen)
  - Zusammengesetzte Datentypen:
    - Datenkonstrukte, welche aus einfachen Datentypen bestehen
    - Beispiel: „String“ als eine Zeichenkette variabler Länge

Primitive Datentypen in Java

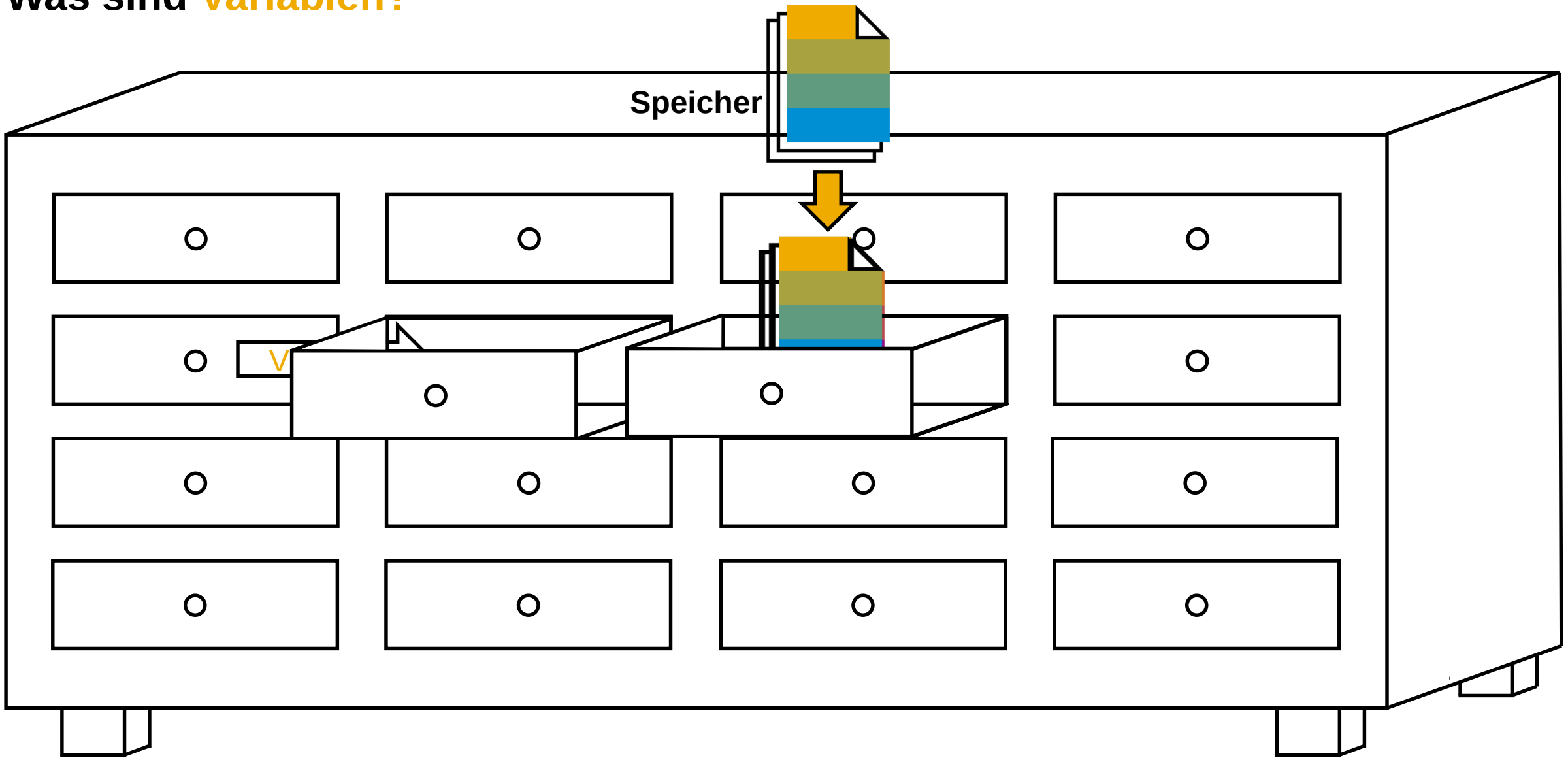
byte	-128 bis 127	8-Bit-Zahl
short	-32.768 bis 32.767	16-Bit-Zahl
int	1.057.295	32-Bit-Zahl (ausreichend für nahezu alle Zahlen)
long	9.472.758.357	64-Bit-Zahl (Maximale Länge einer Zahl)
float	1,568f	32-Bit-Gleitkommazahl
double	47.920,842438	64-Bit-Gleitkommazahl
char	x	16-Bit-Unicodezeichen
boolean	**true** oder **false**	Wahrheitswert

Zusammengesetzte Datentypen in Java

String	„Hallo Welt!“	Zusammengesetzte Zeichen (einfacher "Text"), dynamische Länge
Array	String[ ] tiere	Auflistung mehrerer Elemente des gleichen Datentyps, feste Länge

ArrayList	ArrayList<String> tiere	Dynamische Array Vorteil: Länge muss nicht bei Deklaration angegeben werden

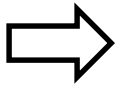
Was sind Variablen?



Variablen in Java

Deklaration

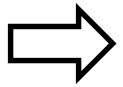
```
int zahl;
float durchschnitt;
boolean wahr;
char kürzel;
String antwort;
```



"Schublade zusammenbauen"

Wertzuweisung

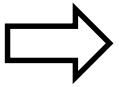
```
zahl = 10;
durchschnitt = 3.15f;
wahr = true;
kürzel = 'C';
antwort = "Hallo Welt!";
```



"Schublade aufmachen und Wert reinlegen"

Aufruf

```
System.out.println(antwort);
int summe = zahl  + 1;
```



"Schublade aufmachen und Wert anschauen"

Übung: Variablen

Erstelle ein Java-Programm, in welchem du folgende Variablen mit dem jeweils passenden Typen deklarierst und mit Werten füllst:

- spielername
- anzahlVersuche
- durchschnittVersuche
- zufallszahl
- nochmalSpielen

Gib danach die Werte der erstellten Variablen in der Konsole aus!

Lösung: Variablen

```
String spielerName;
spielerName = "Peter";

int anzahlVersuche = 5;
double durchschnittVersuche = 4.3;

int zufallsZahl = 20;

boolean nochmalSpielen;
nochmalSpielen = false;

System.out.println("Spieler: " + spielerName);
System.out.println("Anzahl Versuche: " + anzahlVersuche);
System.out.println("Durchschnittliche Anzahl Versuche: " +
durchschnittVersuche);
System.out.println("Zufällige Zahl: " + zufallsZahl);
System.out.println("Nochmal spielen: " + nochmalSpielen);
```

Operatoren

Operatoren

Rechenoperatoren

Addition

5 + 3 = 8

2 + 4 = 6

Multiplikation

7 * 3 = 21

1 * 9 = 9

Modulo (Teilerrest)

5 % 3 = 2

9 % 4 = 1

Subtraktion

8 – 1 = 7

3 – 6 = -3

Division

8 / 2 = 4

1.0 / 2.0 = 0.5

1 / 2 = 0

Operatoren

Kurzschreibweisen

Addition

5 += 3 = 8

2 += 8 = 10

Inkrementieren

6++ = 7

-4++ = -3

Subtraktion

9 -= 3 = 6

2 -= 4 = -2

Dekrementieren

1-- = 0

7-- = 6

Operatoren

Rechnen mit Variablen

- Variablen können in Java, ähnlich wie in der Mathematik, zum Rechnen verwendet werden

```
int a = 5;
int b = 2;
int summe = a + b; // summe = 7
```

```
int a = 11;
a++; // a = 12
```

- Dabei ist immer auf die betroffenen Datentypen zu achten!

```
int ergebnis;
float a = 4.5f;
float b = 3.2f;
ergebnis = a / b; // Fehler: Inkompatible Datentypen!
                  // Lösung: ergebnis vom Typ float
                  //         oder parsen
```

Umwandeln von Datentypen

Parsen

Integer

```
int ergebnis;
float a = 3.537f;
float b = 5.239f;

ergebnis = (int) (a/b);
```

Wichtig:

Klammern nicht vergessen!

String

```
String stringErgebnis = String.valueOf(ergebnis);
```

Von String zu Integer

```
int ergebnis = Integer.parseInt(stringErgebnis);
```

Übung: Operatoren

1. Erstelle zwei Variablen `anzahlVersuche1` (**`int`**) und `anzahlVersuche2` (**`int`**) mit beliebigen positiven Werten.
2. Addiere beide Variablen und speichere das Ergebnis in der Variable `gesamtVersuche` (**`int`**).
3. Erstelle eine Variable `anzahlSpiele` (**`int`**) mit beliebigem positivem Wert.
4. Berechne die durchschnittliche Anzahl an Versuchen pro Spiel und speichere das Ergebnis in der Variable `durchschnittlicheVersuche` (**`double`**).
5. Gib den Inhalt der Variablen in der Konsole aus.
6. Addiere eine beliebige Zahl zu `gesamtVersuche`.
7. Verdopple den Wert der Variable `gesamtVersuche`.
8. Berechne `durchschnittlicheVersuche` neu und gib den neuen Wert aus.

Lösung: Operatoren

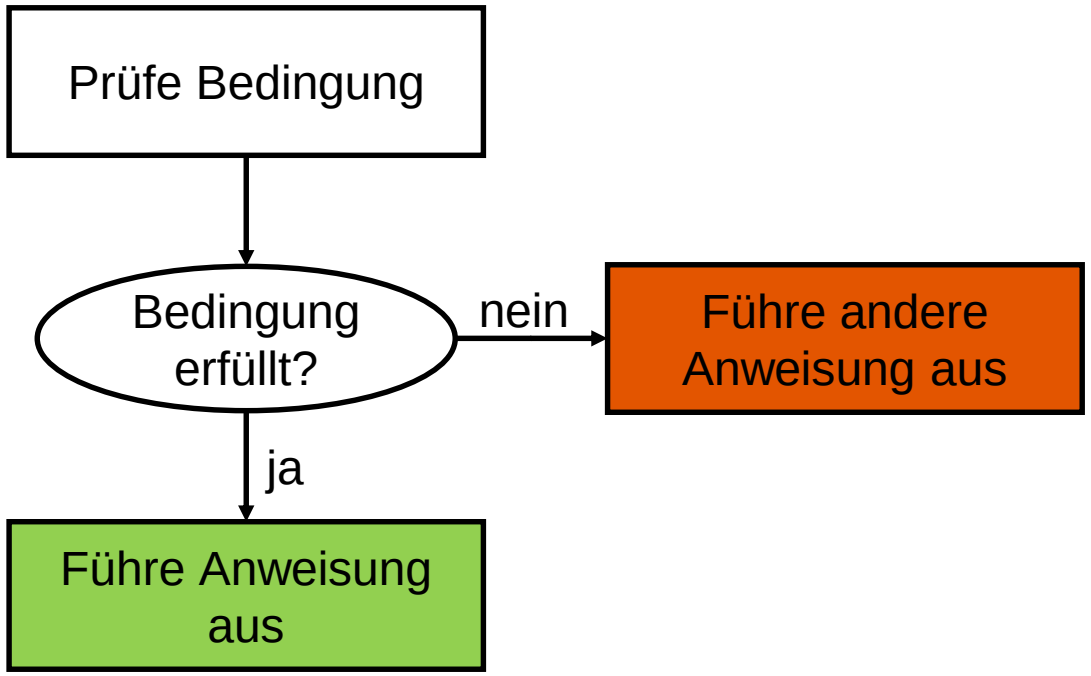
```
int anzahlVersuche1 = 10;
int anzahlVersuche2 = 42;
int gesamtVersuche = anzahlVersuche1 + anzahlVersuche2;
int anzahlSpiele = 2;
double durchschnittlicheVersuche;

System.out.println("anzahlVersuche1 + anzahlVersuche2: " + gesamtVersuche);
System.out.println("anzahlVersuche1 - anzahlVersuche2: " + (anzahlVersuche1 -
anzahlVersuche2));
durchschnittlicheVersuche = (gesamtVersuche / anzahlSpiele);
System.out.println("Durchschnittliche Versuche: " + durchschnittlicheVersuche);

System.out.println("anzahlSpiele++: " + anzahlSpiele++);
System.out.println("++gesamtVersuche: " + ++gesamtVersuche);

gesamtVersuche += 20;
anzahlSpiele *= 2;
durchschnittlicheVersuche = (gesamtVersuche / anzahlSpiele);
System.out.println("Neue durchschnittliche Versuche: " + gesamtVersuche /
anzahlSpiele);
```

Kontrollstrukturen

Bedingte Anweisungen & Schleifen

Kontrollstrukturen

Bedingte Anweisungen




```
boolean erfolg = true;
if (erfolg == true) {
    System.out.println("Erfolg!");
}
```

```
boolean erfolg = false;
if (erfolg == true) {
    System.out.println("Erfolg!");
} else {
    System.out.println("Kein Erfolg!");
}
```

```
String name = "Anna";
if (name.equals("Otto")) {
    System.out.println("Hallo Otto!");
} else if (name.equals("Anna")) {
    System.out.println("Hallo Anna!");
} else {
    System.out.println("Hallo Unbekannte/r!");
}
```

Operatoren

Vergleichsoperatoren

Gleichheit (==)

- ✔ 5 == 5
- ✘ 3 == 5
- ✘ 5 == "5"
- ✔ "Hallo" == "Hallo"

Ungleichheit (!=)

- ✘ 5 != 5
- ✔ 3 != 5
- ✔ 5 != "5"
- ✘ "Hallo" != "Hallo"

Operatoren

Vergleichsoperatoren

Größer (>)

- ✓ 5 > 2
- ✓ 5 > -1
- ✗ 5 > 10
- ✗ 5 > 5

Kleiner (<)

- ✗ 5 < 2
- ✗ 5 < -1
- ✓ 5 < 10
- ✗ 5 < 5

Größer Gleich (>=)

- ✓ 5 >= 2
- ✓ 5 >= -1
- ✗ 5 >= 10
- ✓ 5 >= 5

Kleiner Gleich (<=)

- ✗ 5 <= 2
- ✗ 5 <= -1
- ✓ 5 <= 10
- ✓ 5 <= 5

Operatoren

Vergleichsoperatoren in Java

```
if (5 > 3) {
    System.out.println("5 ist größer als 3!");
}
```

⇨ 5 ist größer als 3!

Keine Ausgabe, da Bedingung nicht erfüllt ist! ⇦

```
if (3 > 5) {
    System.out.println("3 ist größer als 5!");
}
```

```
boolean fehler = false;
// weitere Anweisungen...
if (fehler) {
    System.out.println("Es ist ein Fehler augetreten");
} else {
    System.out.println("Alles hat funktioniert!");
}
```

⇨ Alles hat funktioniert!

Übung: if-Anweisung

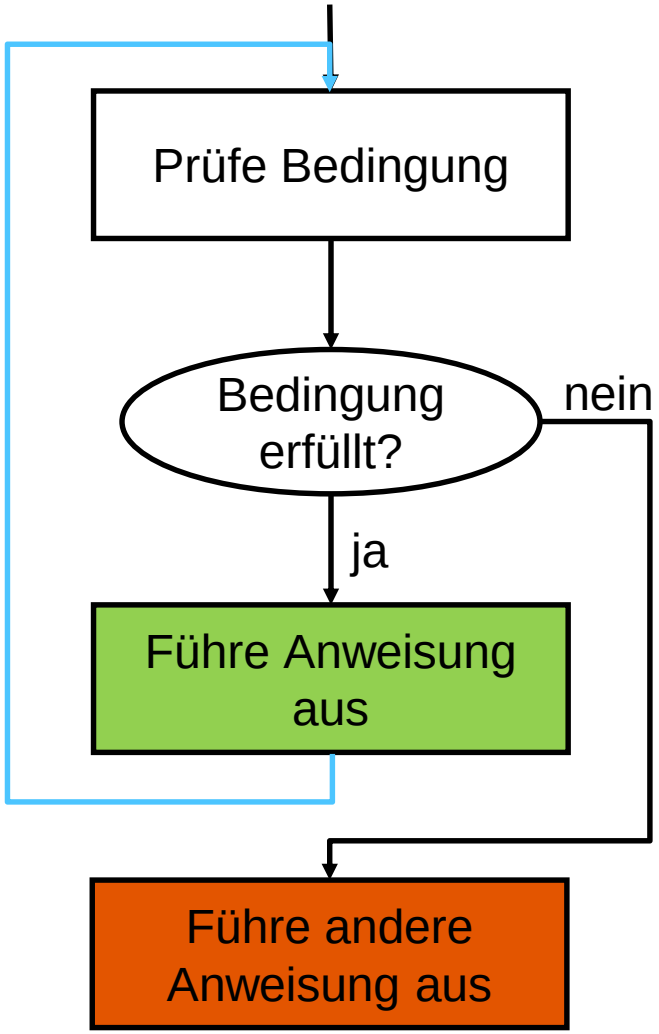
1. Erstelle ein neues Java-Programm, in welchem du folgende Variablen mit beliebigen Werten und passenden Typen erstellst:
   - `name`
   - `vorname`
   - `geschlecht`
   - `alter`
2. Überprüfe ob die Person Erwachsen (über 18 Jahre alt) ist
3. Folgendes soll ausgegeben werden:
   - Für Erwachsene: „Hallo“ + passende Anrede (Herr/Frau) + Nachname
   - Für Kinder: „Hallo“ + Vorname

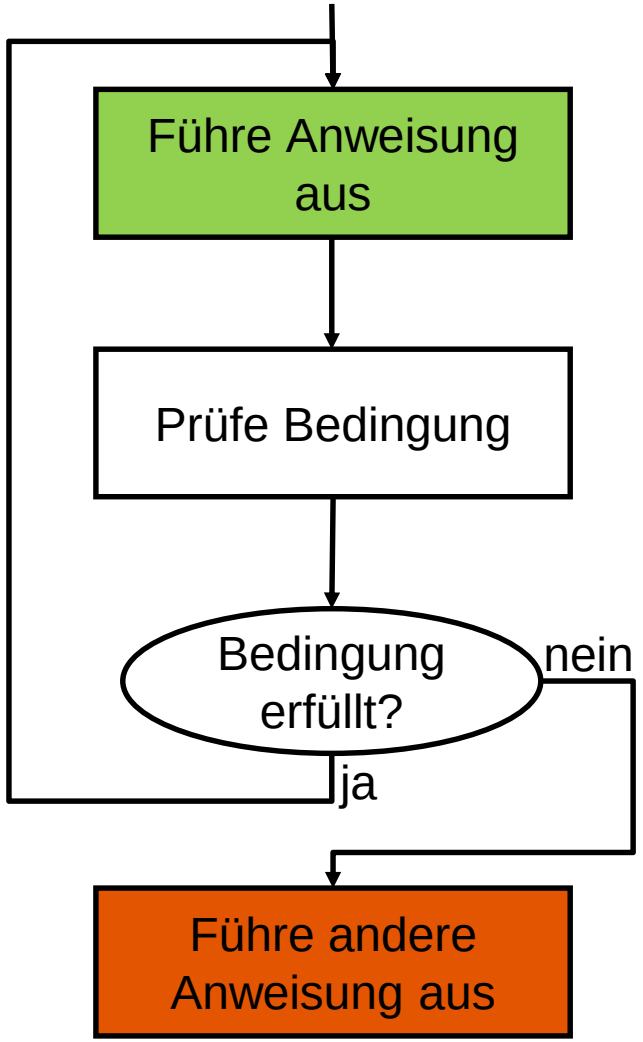
Lösung: Kontrollstrukturen (if)

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

    String name = "Appenmaier", surname = "Daniel";
    char gender = 'm';
    int age = 36;

    if(age >= 18) {
        if(gender == 'm') {
            System.out.println("Hallo Herr " + name +".");
        } else if(gender == 'w') {
            System.out.println("Hallo Frau " + name +".");
        }
    } else {
        System.out.println("Hallo " + surname + ".");
    }

}
```

Kontrollstrukturen

While-Schleife





Kontrollstrukturen

While-Schleife in Java

```
while (true) {
    System.out.println("Dauerschleife");
}
```

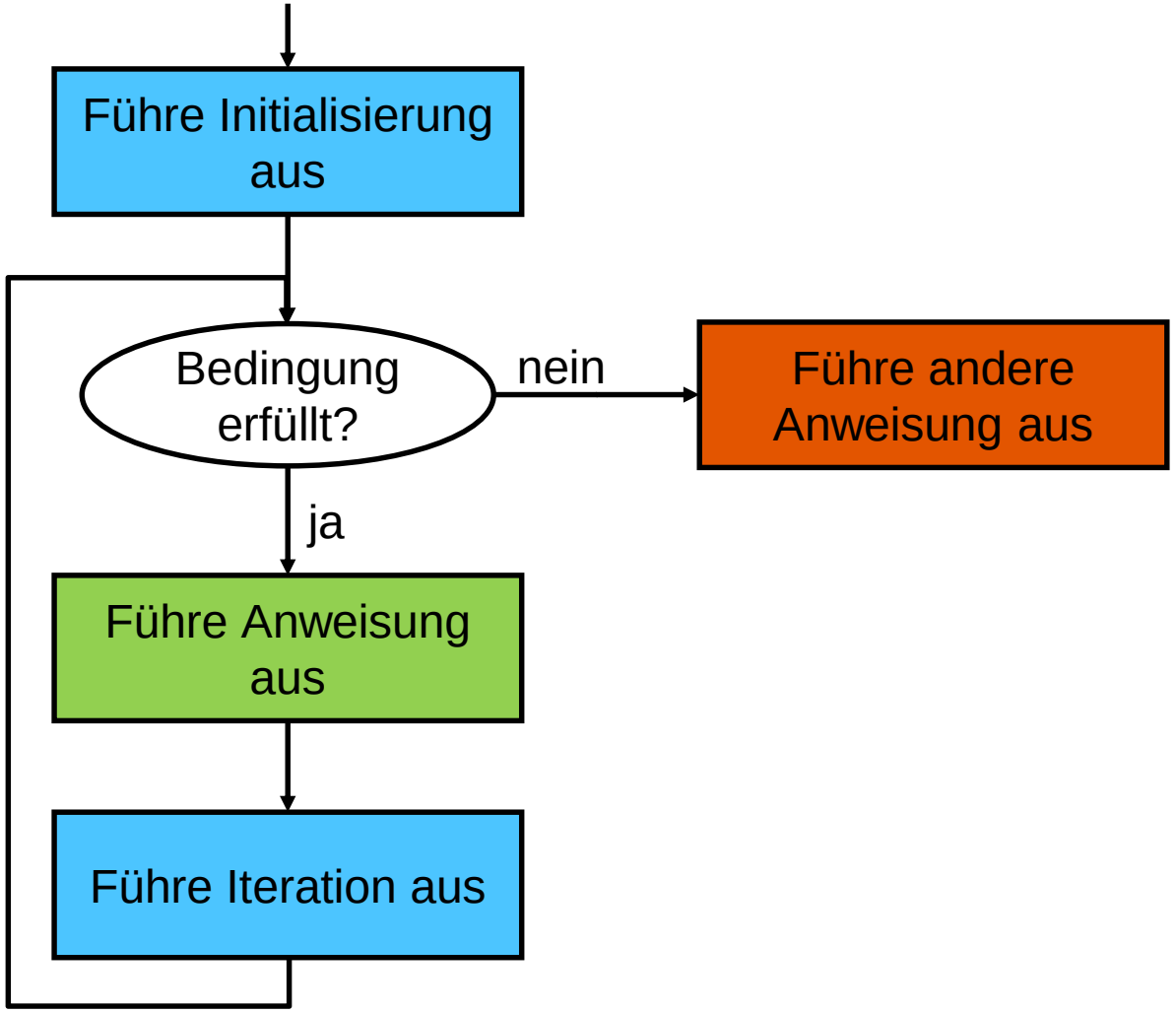
```
do {
    System.out.println("Dauerschleife");
} while (true);
```

```
int i = 5;
while (i > 0) {
    System.out.println("Start in " + i);
    i--;
}
System.out.println("Starte jetzt!");
```

```
String wort = "Hallo Welt!";
int laenge = wort.length();
int i = 0;
while (i < laenge) {
    System.out.println(wort.charAt(i));
    i++;
}
```

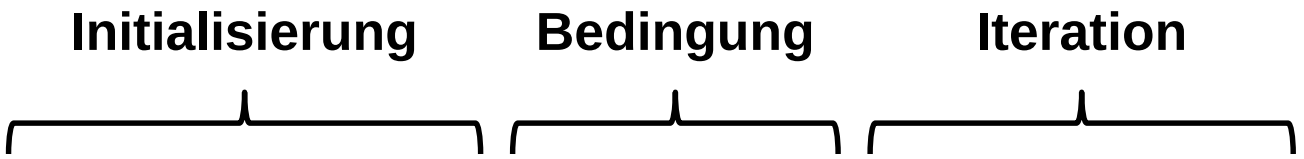
Kontrollstrukturen

For-Schleife



Kontrollstrukturen

For-Schleife in Java




```
for (int i = 5; i > 0; i -= 1) {
    System.out.println("Start in " + i);
}
System.out.println("Starte jetzt!");
```

Anweisung

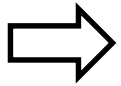
```
for (int i = 0; i < 6; i++) {
    //Logik ;
}
```

Kontrollstrukturen

For-Schleife vs. While-Schleife

While-Schleife

```
int i = 5;
while (i > 0) {
    System.out.println("Start in " + i);
    i--;
}
System.out.println("Starte jetzt!");
```



For-Schleife

```
for (int i = 5; i > 0; i--) {
    System.out.println("Start in " + i);
}
System.out.println("Starte jetzt!");
```

Übung: Schleifen

1. Erstelle ein neues Java-Programm.
2. Erstelle eine while-Schleife die von 10 bis 1 herunterzählt (mit Ausgabe!).
3. Teste das Programm und überprüfe die Ausgabe
4. Verwende anstelle der while-Schleife eine for-Schleife.

Für die Schnellen:

5. Berechne alle Primzahlen zwischen 2 und 100 und gib diese in der Konsole aus.
6. Beschleunige die Berechnung der Primzahlen.

Lösung: Konstrollstrukturen (Schleifen)

While-Schleife:

```
int zaehler = 10;
while (zaehler > 0) {
    System.out.println("Zahl ist: " + zaehler);
    zaehler--;
}
```

For-Schleife:

```
for (int zaehler2 = 10; zaehler2 > 0; zaehler2--) {
        System.out.println("Zahl ist: " + zaehler2);
}
```

Lösung: Konstrollstrukturen (Schleifen)

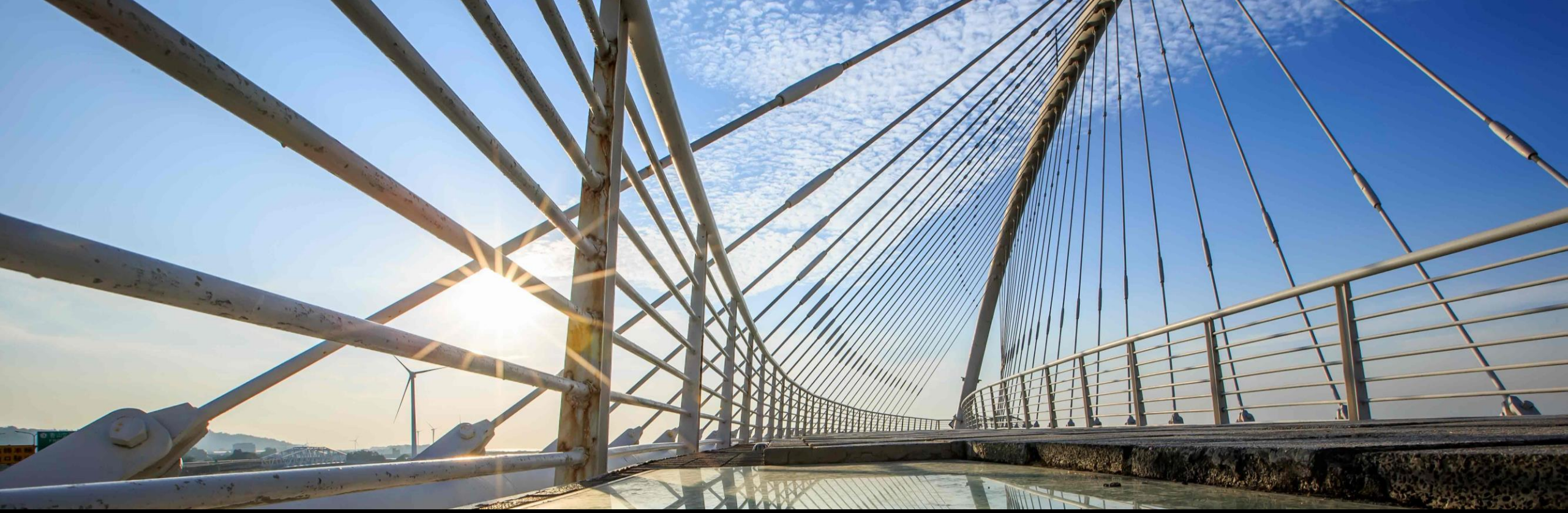
Alle Primzahlen von 2 bis 100:

```
for (int zahl = 2; zahl < 100; zahl++) {
    boolean teilbar = false;
    for (int teiler = 2; teiler < zahl; teiler++) {
        if (zahl % teiler == 0) {
            teilbar = true;
        }
    }
    if (!teilbar) {
        System.out.println(zahl + " ist eine Primzahl");
    }
}
```

Lösung: Konstrollstrukturen (Schleifen)

Alle Primzahlen von 2 bis 100 (beschleunigt):

```
for (int zahl = 2; zahl < 100; zahl++) {
    boolean teilbar = false;
    for (int teiler = 2; teiler < zahl; teiler++) {
        if (zahl % teiler == 0) {
            teilbar = true;
            break;
        }
    }
    if (!teilbar) {
        System.out.println(zahl + " ist eine Primzahl");
    }
}
```

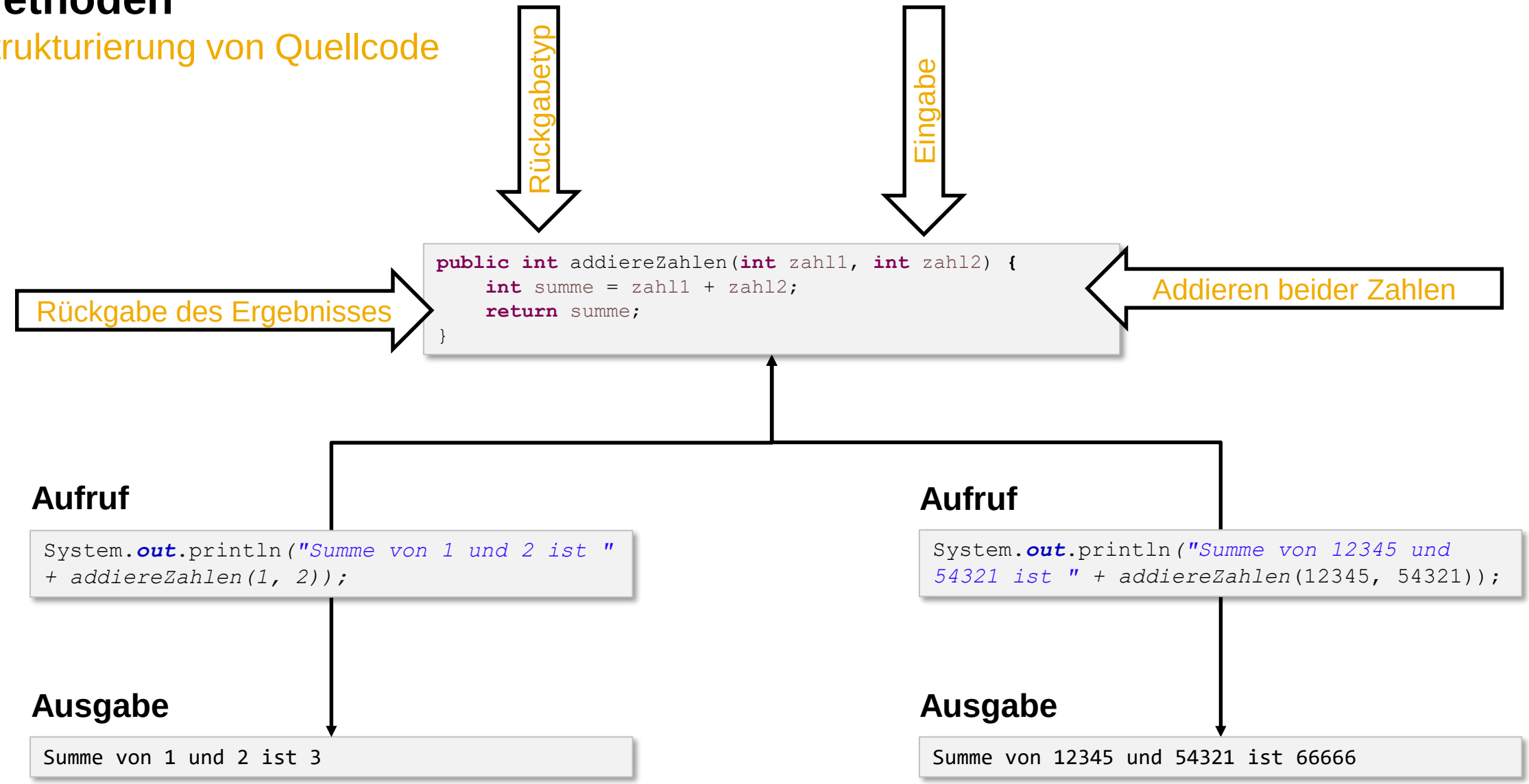


Methoden

Strukturierung von Quellcode

Methoden

Strukturierung von Quellcode



Modifikatoren

- Beschreibung von Methoden, Variablen und Klassen
- Verändern das Verhalten und die Sichtbarkeit des dazugehörigen Codes
- Schutz des Codes vor fremdem Zugriff
- Sichtbarkeit
  - `private`: innerhalb einer Klasse sichtbar
  - `protected`: innerhalb des selben Paketes sichtbar
  - `public`: für jede Klasse sichtbar
- Verhalten:
  - `static`, `abstract`, `final`, `transient`, …

Übung: Methoden

1. Erstelle die statische Methode `eingabePruefen`, die eine Eingabezahl und die zu erratende Zahl entgegen nimmt und einen boolean zurückgibt. Die Methode soll überprüfen, ob die Eingabezahl größer, kleiner oder gleich der zu erratenden Zahl ist. Gib das Ergebnis der Methode in der Konsole aus und gib true zurück, wenn beide Zahlen gleich sind, ansonsten gebe false zurück.

2. Erstelle ein Programm mit einer Variable `zuErratendeZahl` (`int`) und weise dieser einen beliebigen positiven Wert zu und einer Variable `rateVersuch` (`int`) mit dem Wert 1.

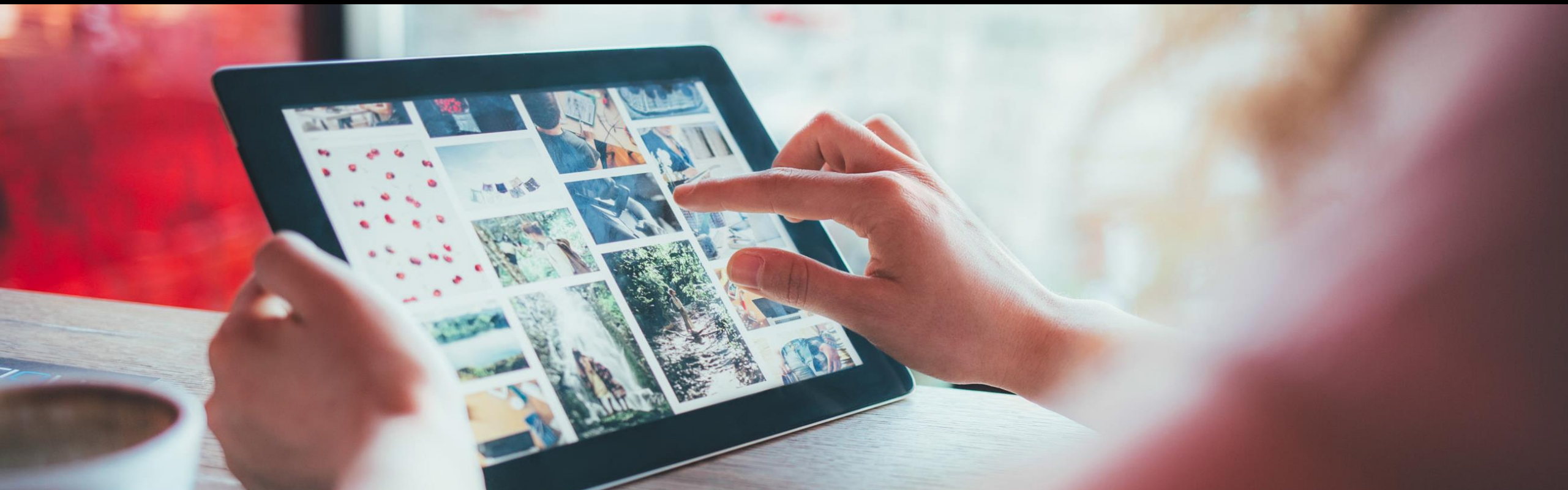
3. Inkrementiere in einer Schleife `rateVersuch` solange, bis `eingabePruefen` true zurück gibt (Methodenaufruf als Schleifenbedingung).

Lösung: Methoden

```
public static void main(String[] args) {
     int zuRatendeZahl = 42;
     int rateVersuch = 1;
     while (true) {
          if (eingabeUeberpruefen(rateVersuch, zuRatendeZahl)) {
               break;
          }
          rateVersuch++;
     }
}
```

```
public static boolean eingabeUeberpruefen(int eingabe, int zuRatendeZahl) {
     if (eingabe > zuRatendeZahl) {
          System.out.println(eingabe + " ist zu hoch!");
          return false;
     } else if (eingabe < zuRatendeZahl) {
          System.out.println(eingabe + " ist zu niedrig!");
          return false;
     } else {
          System.out.println(eingabe + " ist richtig!");
          return true;
     }
}
```

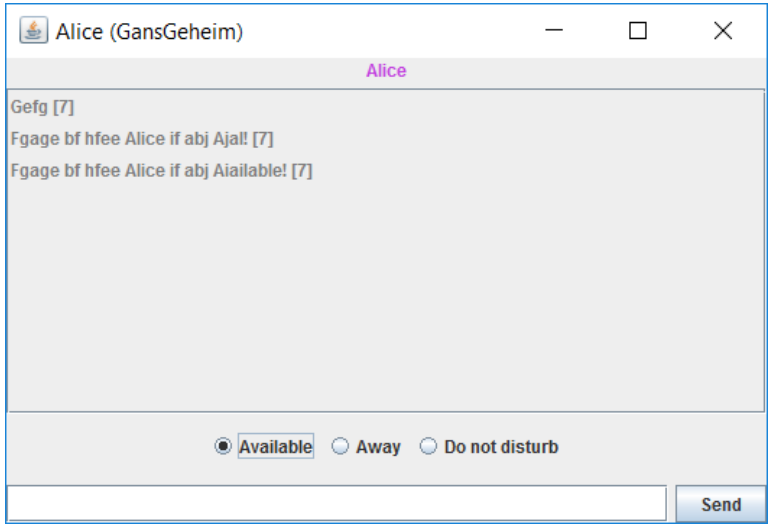
Input & Output



Input & Output
User Interface (UI)

User Interface (UI)

- Ein-/Ausgabe mit einem Benutzerinterface
- Erleichtert die Bedienung
- Erlaubt die Bedienung auch ohne Kenntnisse über das Programm



Vorteile

- Viele verschiedene Möglichkeiten der Programmierung (Swing, JavaFX, …)
- Individuelle Gestaltung
- Erweiterte Ausgabe (Bilder, Ton, …)

Nachteile

- Aufwändig in der Entwicklung
- Fehleranfällig
- Unterschiedlich je nach Betriebssystem

Input & Output

Konsole

```
Player: Michael van Gerwen, 501 points remaining
Enter visit: 10
Scored: 10
==============================
Player: Phil Taylor, 501 points remaining
Enter visit: 20
Scored: 20
==============================
Player: Michael van Gerwen, 491 points remaining
Enter visit:
```

Konsole

- Ein-/Ausgabe über die Konsole
- Erlaubt die Bedienung oftmals nur mit Kenntnissen über das Programm
- Praktisch für Prototypen eines größeren Programms

Vorteile

- Einfach zu Entwickeln
- UI nicht zwingend erforderlich
  - Programm kann auch vollautomatisch ablaufen

Nachteile

- Wenig Vielfalt in der Ausgabe
- Schwierig in der Bedienung

Input & Output

Konsole - Beispiel

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

System.out.print("Gib deinen Namen ein: ");
String name = scanner.next();

System.out.print("Gib eine Ganzzahl ein: ");
int ganzzahl = scanner.nextInt();
```

Übung: Input

- Schreibe ein Java-Programm, welches einen Text, eine Gleitkommazahl und eine Ganzzahl einliest und danach wieder ausgibt.

```
Gib eine Ganzzahl ein: 5
Gib eine Gleitkommazahl ein: 3,1415
Gib einen Text ein: Hello

Deine Ganzzahl: 5
Deine Gleitkommazahl: 3.1415
Dein Text: Hello
```

Lösung: Input

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

System.out.print("Gib eine Ganzzahl ein: ");
int ganzzahl = scanner.nextInt();
System.out.print("Gib eine Gleitkommazahl ein: ");
double gleitkommazahl = scanner.nextDouble();
System.out.print("Gib einen Text ein: ");
String zeichenkette = scanner.next();

scanner.close();

System.out.println("Deine Ganzzahl: " + ganzzahl);
System.out.println("Deine Gleitkommazahl: " + gleitkommazahl);
System.out.println("Dein Text: " + zeichenkette);
```

Abschlussübung: Zahlenraten

Versuche aus dem heute gelernten das Spiel Zahlenraten zu programmieren:

1. Zuerst wird der Spieler nach seinem Namen gefragt. Danach wird zufällig eine geheime Zahl generiert.
2. Der Spieler wird nach einer Schätzung für die geheime Zahl gefragt.
3. Je nachdem, ob die geschätzte Zahl größer oder kleiner ist, wird eine Meldung ausgegeben und der Spieler hat einen neuen Versuch.
4. Ist die Schätzung richtig, so wird die benötigte Anzahl an Versuchen ausgegeben und die Frage, ob der Spieler erneut spielen will.

Wenn du dies geschafft hast, kannst du dein Spiel nach deinen Wünschen und Vorstellungen modifizieren.

Lösung: Zahlenraten

```
Scanner scanner = new Scanner(System.in);
System.out.print("Was ist dein Name? ");
String name = scanner.next();
int eingabe = 0;
while (eingabe == 0) {
    int zufaelligeZahl = (int)(Math.random() * 100) + 1;
    int durchlaeufe = 1;
    while (true) {
          System.out.printf("%s, schätze eine Zahl (1 - 100): ", name);
          eingabe = scanner.nextInt();
          if (eingabe == zufaelligeZahl) {
              System.out.printf("Versuch #%d: %d ist richtig!\n", durchlaeufe, eingabe);
              break;
          } else if (eingabe < zufaelligeZahl) {
              System.out.printf("Versuch #%d: %d ist zu niedrig!\n", durchlaeufe, eingabe);
          } else {
              System.out.printf("Versuch #%d: %d ist zu hoch!\n", durchlaeufe, eingabe);
          }
          durchlaeufe++;
    }
    System.out.println("Du hast die richtige Zahl erraten! Du hast " + durchlaeufe + " gebraucht!");
    System.out.print("Schreibe eine 0 um das Spiel neuzustarten oder etwas anderes zum beenden: ");
    eingabe = scanner.nextInt();
}
```

Thank You.

Contact Information:

Jonas Schilling
jonas.schilling@sap.com
Student of Business Information Technology

Patrick Berlet
patrick.berlet@sap.com
Student of Business Information Technology

SAP folgen auf

www.sap.com/germany/contactsap

THE BEST RUN SAP®