Verzweigungen

Einführung in die Programmierung 1

Sommersemester 21

TU Wien

Überblick

- if/else-Anweisungen
- switch-Anweisung

if/else-Anweisungen

Motivation

- Bisher wurden die einzelnen Zeilen eines Programms sequentiell abgearbeitet
- Sehr oft sollen aber bestimmte Anweisungen situationsabhängig ausgewählt werden
- Mit Verzweigungen kann der Ablauf des Programms situationsabhängig gesteuert werden

Elementare Anweisungen (1)

- Zuweisung
- Ausdrucksanweisung

```
Ausdruck;
```

  - Nur sinnvoll, wenn der Ausdruck Seiteneffekte hat (z. B. i++)

Elementare Anweisungen (2)

- Block
  - Zusammenfassung von aufeinander folgenden Anweisungen

```
{
  Anweisung1;
  Anweisung2;
  …
}
```

- Leere Anweisung

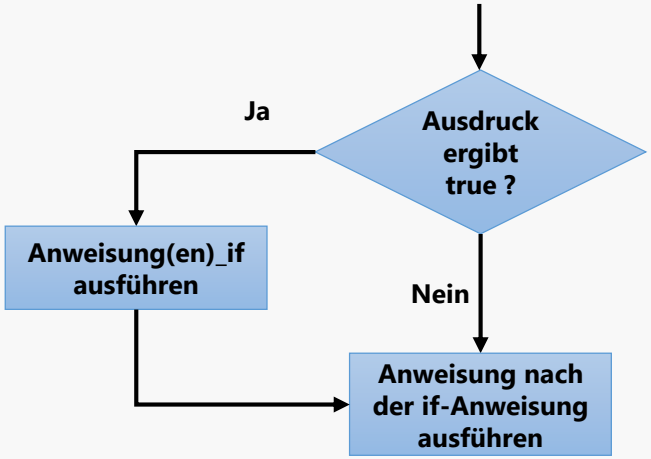
```
;
```

if-Anweisung

- Allgemeine Form

```
if (Ausdruck)
  Anweisung(en)_if
```

- Ablauf



Beispiel (if)

Die Methode random der Klasse Math liefert einen Wert (Zufallszahl) größer oder gleich 0 aber kleiner als 1.0 (vom Typ double) zurück

```
public class IfTest1 {

    public static void main(String[] args) {
        int first = (int) (Math.random() * 100);
        int second = (int) (Math.random() * 100);
        if (first > second) {
            System.out.println("first > second");
        }
        System.out.println("first = " + first);
        System.out.println("second = " + second);
    }

}
```

```
first = 39
second = 75
```

```
first > second
first = 64
second = 49
```

if – Hinweise

- Bei mehreren Anweisungen Block verwenden
  - Beispiel für Unterschied im Ergebnis

```
int x = 5;
int y = 3;
if (x < 10) {
    System.out.println(x + y);
    System.out.println(x - y);
}
if (x < 10)
    System.out.println(x + y);
    System.out.println(x - y);
```

```
8
2
8
2
```

Absichtlich falsche Formatierung

```
int x = 15;
int y = 3;
if (x < 10) {
    System.out.println(x + y);
    System.out.println(x - y);
}
if (x < 10)
    System.out.println(x + y);
    System.out.println(x - y);
```

```
12
```

Hinweis – Klasse Math

- Hilfsklasse für mathematische Operationen
  - `abs`, `min`, `max`, `sin`, `cos`, `log`, `pow`, `sqrt` …
  - Enthält auch Konstanten `E` (Eulersche Zahl) und `PI`
- API
  - https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/lang/Math.html

Hinweis zu Einrückungen

- Erhöhen die Lesbarkeit
  - Programmstruktur sichtbar machen
- Typische Einrückungstiefe
  - 1 Tabulator
- Einfache if-Anweisungen werden manchmal einzeilig geschrieben

```
if (n != 0) x = x/n;
```

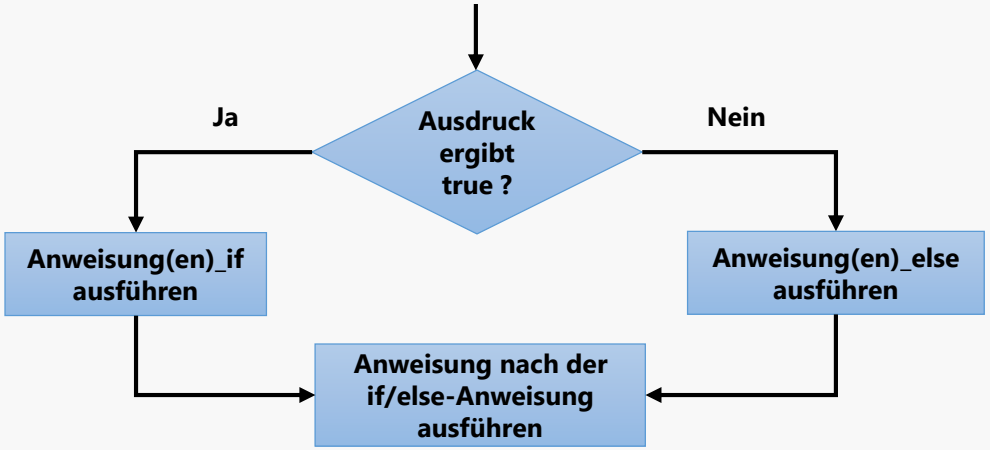
  - Das sollte, wenn möglich, vermieden werden

if-else

- Allgemeine Form

```
if (Ausdruck)
  Anweisung(en)_if
else
  Anweisung(en)_else
```

- Ablauf



Beispiel (if-else)

```
public class IfTest2 {

    public static void main(String[] args) {
        int first = (int) (Math.random() * 100);
        int second = (int) (Math.random() * 100);
        if (first > second) {
            System.out.println("first = " + first);
            System.out.println("second = " + second);
            System.out.println("first > second");
        } else {
            System.out.println("first = " + first);
            System.out.println("second = " + second);
            System.out.println("first <= second");
        }
    }

}
```

```
first = 59
second = 53
first > second
```

```
first = 49
second = 71
first <= second
```

Beispiel (Maximum dreier Zahlen)

- 2 verschiedene Versionen
  - Eine Version mit mehreren verschachtelten if-Anweisungen (`MaximumOfThree.java`)
  - Eine Version mit weniger Vergleichen aber mehr Zuweisungen (`MaximumOfThree2.java`)
- Hinweis
  - Längere Programme werden ab jetzt in IntelliJ gezeigt
  - Alle Programme werden in TUWEL angeboten
    - Einzelne zip-Dateien zu den jeweiligen Kapiteln

Vergleich der zwei Versionen (1)

```
…
if (firstNumber > secondNumber) {
    if (firstNumber > thirdNumber) {
        maximum = firstNumber;
    } else {
        maximum = thirdNumber;
    }
} else {
    if (secondNumber > thirdNumber) {
        maximum = secondNumber;
    } else {
        maximum = thirdNumber;
    }
}
…
```

Beispiel: Diese Zuweisung wird nur erreicht, falls firstNumber > secondNumber **und** firstnumber > thirdnumber gilt

Vergleich der zwei Versionen (2)

```
…
maximum = firstNumber;
if (secondNumber > maximum) {
    maximum = secondNumber;
}
if (thirdNumber > maximum) {
    maximum = thirdNumber;
}
…
```

Diese if-Anweisungen sind unabhängig voneinander. Es kann eine beliebige Kombination davon ausgeführt werden.

Vergleich der zwei Versionen (3)

```
…
if (firstNumber > secondNumber) {
    if (firstNumber > thirdNumber) {
        maximum = firstNumber;
    } else {
        maximum = thirdNumber;
    }
} else {
    if (secondNumber > thirdNumber) {
        maximum = secondNumber;
    } else {
        maximum = thirdNumber;
    }
}
…
```

```
…
maximum = firstNumber;
if (secondNumber > maximum) {
    maximum = secondNumber;
}
if (thirdNumber > maximum) {
    maximum = thirdNumber;
}
…
```

- Vergleiche
  - Die 2. Version (rechts) ist textuell kürzer und leichter lesbar
  - 1. Version braucht immer 2 Vergleiche und 1 Zuweisung
  - 2. Version braucht immer 2 Vergleiche und im Durchschnitt 2 Zuweisungen

Dangling else (1)

- Ein `else`-Zweig bei zwei `if`-Anweisungen
  - Wohin gehört der `else`-Zweig?
- Beispiel (irreführend formatiert)

```
if (counter < 5)
  if (counter % 2 == 0)
    System.out.println("HERE1");
else
  System.out.println("HERE2");
```

Dangling else (2)

- Beispiel (irreführend formatiert)

```
if (counter < 5)
  if (counter % 2 == 0)
    System.out.println("HERE1");
else
  System.out.println("HERE2");
```

- Auflösung durch Compiler
  - `else` gehört zum textuell letzten freien `if` im selben Block
  - `counter` mit Wert 7 führt zu keiner Ausgabe
  - `counter` mit Wert 3 führt zur Ausgabe `"HERE2"`

Bedingungsoperator

- Ternärer Operator und rechtsassoziativ
- Form

```
Bedingung ? Ausdruck1 : Ausdruck2;
```

- Beispiel

```
int max, a, b;
…
max = (a > b) ? a : b;
```

```
if (a > b) {
    max = a;
} else {
    max = b;
}
```

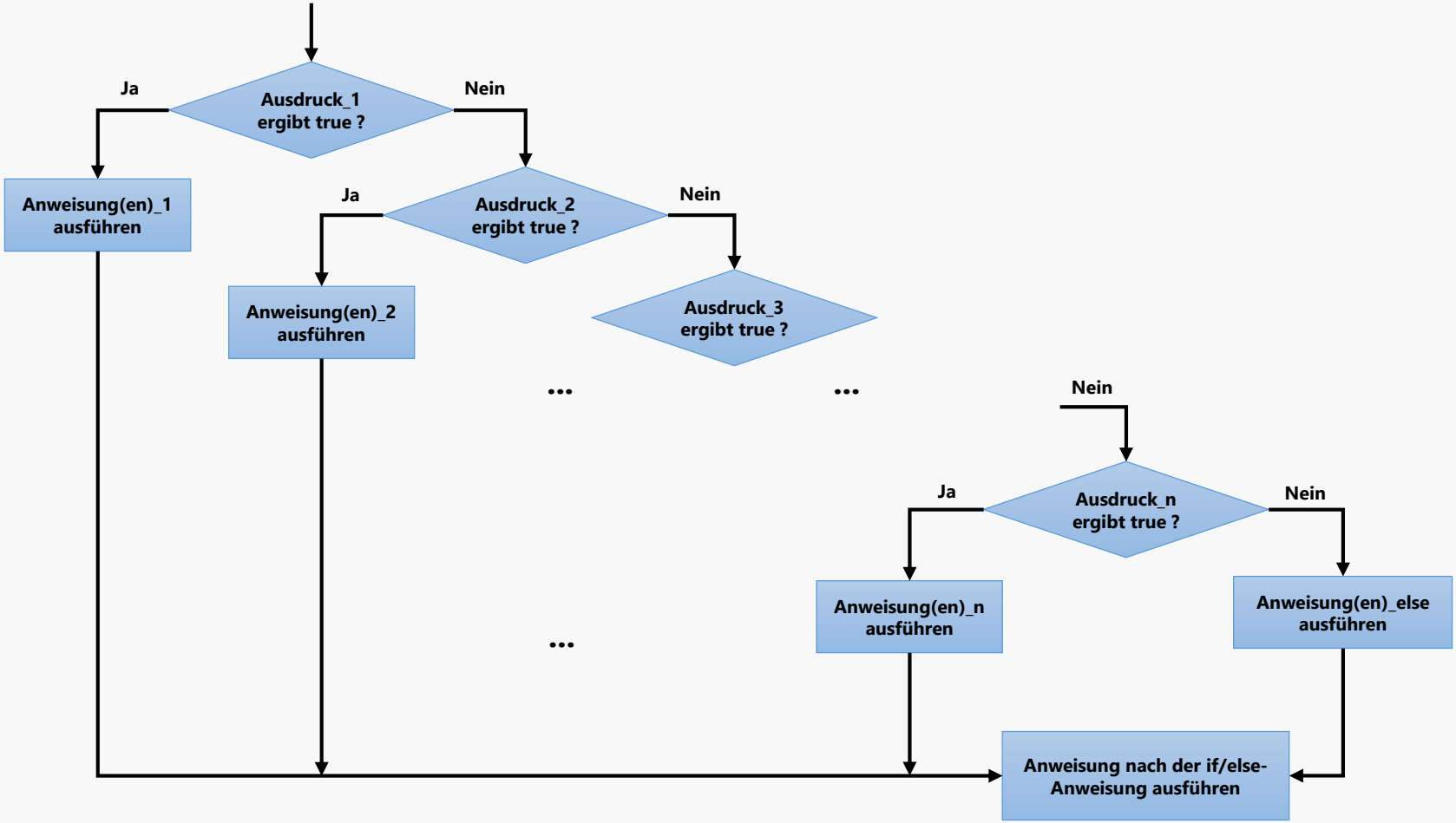
Mehrfachverzweigung bei if

- Allgemeine Form

```
if (Ausdruck_1)
  Anweisung(en)_1
else if (Ausdruck_2)
  Anweisung(en)_2
…
else if (Ausdruck_n)
  Anweisung(en)_n
else
  Anweisung(en)_else
```

- Ein Vergleich wird nach dem anderen durchgeführt (Seiteneffekte beachten!)
- Bei der **ersten Bedingung**, die wahr ist, wird die Anweisung (Anweisungsfolge) abgearbeitet und dann abgebrochen

Mehrfachverzweigung – Ablauf



Beispiel (Mehrfachverzweigung)

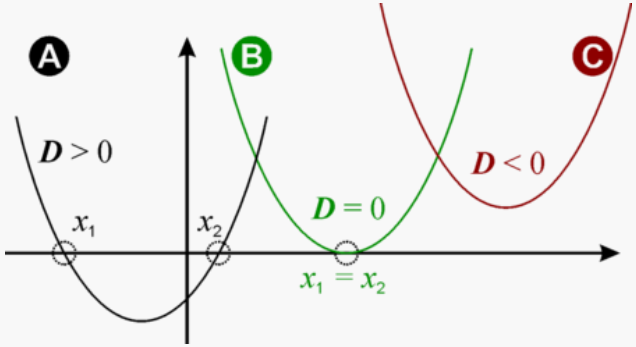
- Lösen quadratischer Gleichungen (`QuadraticEquation.java`)

$$ax^2 + bx + c = 0$$

- Es werden drei Koeffizienten eingelesen
- Danach wird die Diskriminante berechnet

$$D = b^2 - 4ac$$

- Drei Fälle unterscheiden



https://de.wikipedia.org/wiki/Quadratische_Gleichung

Beispiel (Mehrfachverzweigung)

```java
import java.util.Scanner;

public class QuadraticEquation {

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

        System.out.print("First coefficient: ");
        double a = inputScanner.nextDouble();
        System.out.print("Second coefficient: ");
        double b = inputScanner.nextDouble();
        System.out.print("Third coefficient: ");
        double c = inputScanner.nextDouble();

        System.out.println("Solving quadratic equation ...");
        double d = Math.pow(b, 2) - 4 * a * c;
        if (d < 0) {
            System.out.println("No Solution");
        } else if (Double.compare(d, 0) == 0) {
            System.out.println("1 solution: " + -b / (2 * a));
        } else {
            d = Math.sqrt(d);
            System.out.print("2 solutions: ");
            System.out.print((-b + d) / (2 * a) + " and ");
            System.out.println((-b - d) / (2 * a));
        }

    }

}
```

- Verwendet weitere Math-Methoden
- Methode compare aus der Klasse Double ermöglicht einen genauen numerischen Vergleich

```
First coefficient: 1,5
Second coefficient: 7,2
Third coefficient: 2,3
Solving quadratic equation ...
2 solutions: -0.3441141406520967 and -4.455885859347903
```

Weiteres Beispiel für Mehrfachverzweigung

```java
import java.util.Scanner;

public class GradingExample1 {

    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Please enter number: ");
        int grade = inputScanner.nextInt();

        String text;
        if (grade == 1) {
            text = " excellent ";
        } else if (grade == 2) {
            text = " good ";
        } else if (grade == 3) {
            text = " satisfactory ";
        } else if (grade == 4) {
            text = " sufficient ";
        } else if (grade == 5) {
            text = " insufficient ";
        } else {
            text = " no grade ";
        }

        System.out.println("Your grade:" + text);
    }

}
```

```
Please enter number: 2
Your grade: good
```

```java
import java.util.Scanner;

public class GradingExample1 {

    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Please enter number: ");
        int grade = inputScanner.nextInt();

        String text = " no grade ";
        if (grade == 1) {
            text = " excellent ";
        } else if (grade == 2) {
            text = " good ";
        } else if (grade == 3) {
            text = " satisfactory ";
        } else if (grade == 4) {
            text = " sufficient ";
        } else if (grade == 5) {
            text = " insufficient ";
        }

        System.out.println("Your grade:" + text);
    }

}
```

- Zweite Version rechts ohne else-Zweig
- text muss dabei korrekt initialisiert werden

Weiteres Beispiel – schlechte Alternative

- Mit verschachtelten Bedingungsoperatoren
- Zeigt was möglich ist
  - Repräsentiert schlechten Stil!

```
import java.util.Scanner;

public class GradingExample2 {

    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Please enter number: ");
        int grade = inputScanner.nextInt();

        String text;
        text = grade == 1 ? " excellent " :
                grade == 2 ? " good " :
                        grade == 3 ? " satisfactory " :
                                grade == 4 ? " sufficient " :
                                        grade == 5 ? " insufficient " :
                                                " no grade ";

        System.out.println("Your grade:" + text);
    }

}
```

```
Please enter number: 2
Your grade: good
```

if/else – Vergleich

Situation	Konstrukt	Beispiel für Form

Beliebige Kombination von Anweisungen mit Überprüfung ausführen	Sequentielle if-Anweisungen	`if (Ausdruck_1)`
		`  Anweisung(en)_1`
		`if (Ausdruck_2)`
		`  Anweisung(en)_2`
		`if (Ausdruck_3)`
		`  Anweisung(en)_3`

Keine oder eine Anweisung mit Überprüfung ausführen	Verschachtelte if-Anweisungen ohne else am Ende	`if (Ausdruck_1)`
		`  Anweisung(en)_1`
		`else if (Ausdruck_2)`
		`  Anweisung(en)_2`
		`else if (Ausdruck_3)`
		`  Anweisung(en)_3`

Genau eine Anweisung mit Überprüfung ausführen	Verschachtelte if-Anweisungen mit else am Ende	`if (Ausdruck_1)`
		`  Anweisung(en)_1`
		`else if (Ausdruck_2)`
		`  Anweisung(en)_2`
		`else`
		`  Anweisung(en)_3`

switch-Anweisung

switch

- Allgemeine Form

```
switch (Ausdruck) {
  case Wert_1:
        Anweisung(en)_1
        break;
  case Wert_2:
        Anweisung(en)_2
        break;
  …
  case Wert_n:
        Anweisung(en)_n
        break;
  default:
        Anweisung(en)_default
        break;              /* Nicht notwendig, aber hilfreich */
}
```

switch – allgemeine Beschreibung

- Ausdruck wird ausgewertet
  - Erlaubte Datentypen (von den bisher besprochenen): `short`, `byte`, `int`, `char` oder `String`
- Auswahl einer Alternative
  - Wert des Ausdrucks entspricht dem **konstanten Wert** hinter einer `case`-Marke

```
switch (Ausdruck) {
     case Wert_1:
          Anweisung(en)_1
          break;
     case Wert_2:
          Anweisung(en)_2
          break;
     …
     case Wert_n:
          Anweisung(en)_n
          break;
     default:
          Anweisung(en)_default
          break;
}
```

switch – break

- Sorgt dafür, dass **nicht** zur folgenden `case`-Marke weitergesprungen wird
  - `switch`-Anweisung wird verlassen
- Ohne `break` können zwei oder mehrere `case`-Marken verbunden werden
  - Alle nachfolgenden Anweisungen werden bis zum nächsten `break` oder dem Ende der `switch`-Anweisung abgearbeitet
  - „Fall-through“

switch – default

- `default`-Marke
  - ist optional
  - wird ausgeführt, wenn keine `case`-Marke passt
  - ohne `default` ist es möglich, dass keine Anweisungen ausgeführt werden
- Hinweis
  - Jeder Wert darf nur einmal bei einer `case`-Marke vorkommen

Beispiele (switch)

- Beispiel mit switch statt Mehrfachverzweigung
  - Variante von GradingExample1.java

```
Please enter number: 2
Your grade: good
```

```java
import java.util.Scanner;

public class GradingExample3 {

    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Please enter number: ");
        int grade = inputScanner.nextInt();

        String text;
        switch (grade) {
            case 1:
                text = " excellent ";
                break;
            case 2:
                text = " good ";
                break;
            case 3:
                text = " satisfactory ";
                break;
            case 4:
                text = " sufficient ";
                break;
            case 5:
                text = " insufficient ";
                break;
            default:
                text = " no grade ";
        }

        System.out.println("Your grade: " + text);
    }

}
```

Beispiele (switch)

- Alternatives Beispiel mit switch
  - Benutzt „Fall-through“

```
Please enter number: 2
Your grade: passed
```

```
Please enter number: 3
Your grade: passed
```

```java
import java.util.Scanner;

public class GradingExample4 {

    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(System.in);
        System.out.print("Please enter number: ");
        int grade = inputScanner.nextInt();

        String text;
        switch (grade) {
            case 1:
                text = " with distinction ";
                break;
            case 2:
            case 3:
            case 4:
                text = " passed ";
                break;
            case 5:
                text = " not passed ";
                break;
            default:
                text = " no grade ";
        }

        System.out.println("Your grade:" + text);
    }

}
```

switch-Ausdruck (Java 14)

- switch kann einen Wert erzeugen
- Beispiel

Hinweis: Beispiel verwendet die Enumeration (ähnlich zu Klasse) Day. Darin sind die Wochentage als Kostanten enthalten.

```
public class SwitchTest {
    public enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY; }
    public static void main(String[] args) {
        int numLetters = 0;
        Day day = Day.WEDNESDAY;
        switch (day) {
            case MONDAY:
            case FRIDAY:
            case SUNDAY:
                numLetters = 6;
                break;
            case TUESDAY:
                numLetters = 7;
                break;
            case THURSDAY:
            case SATURDAY:
                numLetters = 8;
                break;
            case WEDNESDAY:
                numLetters = 9;
                break;
            default:
                numLetters = -1;
        }
        System.out.println(numLetters);
    }
}
```

Seit Java 14

```
public class SwitchTest {

    public enum Day {
        SUNDAY, MONDAY, TUESDAY,
        WEDNESDAY, THURSDAY, FRIDAY, SATURDAY;
    }

    public static void main(String[] args) {
        Day day = Day.WEDNESDAY;
        int numLetters = switch (day) {
            case MONDAY, FRIDAY, SUNDAY -> 6;
            case TUESDAY -> 7;
            case THURSDAY, SATURDAY -> 8;
            case WEDNESDAY -> 9;
            default -> -1;
        };
        System.out.println(numLetters);
    }

}
```

switch-Ausdruck

- Syntaktische Änderung („->“ statt „:“)
- Mehrere Werte hinter case-Klausel
- Kein break mehr
  - Es existiert kein Fall-through!
- switch kann einen Wert zurückgeben
  - Hilfsvariablen können vermieden werden
- Compiler überprüft auf Vollständigkeit
  - Alle möglichen Werte müssen abgedeckt werden (direkt oder über default)

Schlüsselwort yield bei switch

- Zwei Anwendungsfälle

Alternative Schreibweise (ähnlich zu alter Syntax)

```
…
Day day = Day.WEDNESDAY;
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY: yield 6;
    case TUESDAY: yield 7;
    case THURSDAY, SATURDAY: yield 8;
    case WEDNESDAY: yield 9;
    default: yield -1;
};
…
```

Rückgabe eines Wertes bei mehreren Anweisungen in einem Block

```
…
Day day = Day.WEDNESDAY;
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> {
        if (day == Day.SUNDAY) {
            System.out.println("Sunday!");
        }
        yield 6;
    }
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
    default -> -1;
};
…
```

Anwendung von if und switch

- Abhängig von der Lesbarkeit und Ausdrucksstärke
- `if`
  - Wenige Werte vergleichen
  - Komplexere Ausdrücke in Bedingungen
- `switch`
  - Anhand mehrerer konstanter Werte verzweigen

Zusammenfassung

- if-Anweisung
  - if-else
  - Mehrfachverzweigung
- switch-Anweisung
- switch-Ausdruck