Schleifen

Einführung in die Programmierung 1

Sommersemester 21

TU Wien

Überblick

- Schleifen in Java
- Praktische Überlegungen
- Verschachtelte Schleifen

Schleifen in Java

Motivation

- Bisher fast nur Anweisungsfolgen (mit Verzweigungen)
- Viele Anweisungsfolgen müssen aber wiederholt ausgeführt werden
- Bisherige Beispiele für Wiederholungen
  - Wiederholtes Einlesen
  - Berechnung(en) mehrmals wiederholen
  - Grafische Ausgabe wiederholt neu zeichnen und ausgeben (Animation)

Schleifen

- Varianten in Java

while

- Vor Schleifendurchlauf Ausdruck überprüfen
- Dann Schleifenrumpf (Anweisung) ausführen

do-while

- Zuerst Schleifenrumpf (Anweisung) ausführen
- Nach Schleifendurchlauf Ausdruck überprüfen

for

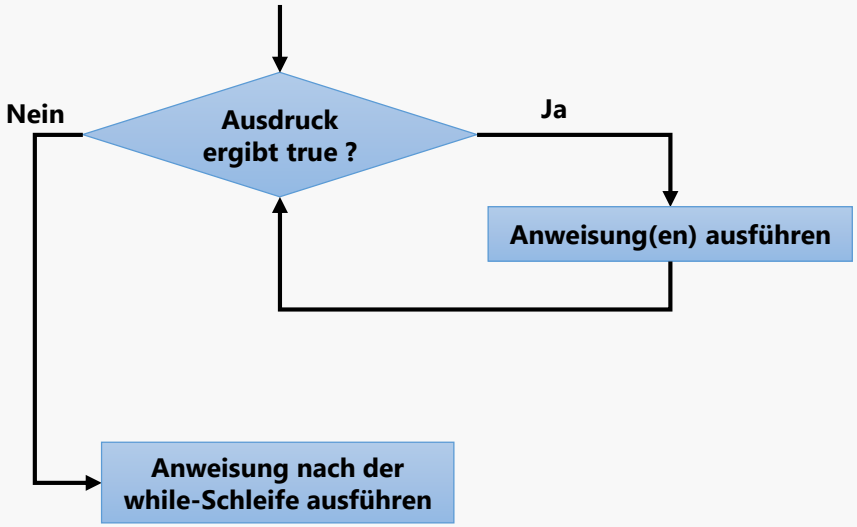
- Zuerst Vorbereitung (vor erstem Schleifendurchlauf)
- Vor Schleifendurchlauf Ausdruck überprüfen
- Dann Schleifenrumpf (Anweisung) ausführen

while (1)

- Allgemeine Form

```
while (Ausdruck)
    Anweisung(en)
```

- Ausführung



while (2)

- Beliebiger Teil des Ausdrucks sollte in Anweisung(en) (Schleifenrumpf) verändert werden
  - Ansonsten kann der Wert des Ausdrucks im Schleifenkopf z. B. nicht von wahr auf falsch wechseln (Endlosschleife)
- Weitere Bezeichnungen
  - Abweisschleife
  - Vorprüfende oder kopfgesteuerte Schleife

Beispiel (while)

```java
public class WhileTest {

    public static void main(String[] args) {
        long i = 2;
        System.out.println("N \tN^2 \tN^3");
        while (i <= 1024) {
            System.out.println(i + "\t" + i * i + "\t" + i * i * i);
            i *= 2;
        }
    }

}
```

```
N       N^2       N^3
2       4         8
4       16        64
8       64        512
16      256       4096
32      1024      32768
64      4096      262144
128     16384     2097152
256     65536     16777216
512     262144    134217728
1024    1048576   1073741824
```

Weitere Beispiele für while-Schleifen

```java
int i = 0;
while (i < 10) {
    System.out.print(i + " ");
    i++;
}
```

- Zahlen von 0 bis 9 auf der Konsole ausgeben
- Anzahl der Schleifendurchläufe vorgegeben

```java
int n = 17;
int count = 0;
while (n >= 1) {
    n /= 2;
    count++;
}
System.out.println(count);
```

- Zählen, wie oft eine Zahl größer 0 durch 2 dividiert werden kann (Hinweis: zählt die Anzahl der Binärstellen einer Zahl größer 0)
- Input (in diesem Fall n) bestimmt die Anzahl der Schleifendurchläufe

```java
int sum = 0;
int i = 1;
while (i <= 10) {
    sum += i;
    i++;
}
System.out.println(sum);
```

- Alle Zahlen von 1 (einschließlich) bis 10 (einschließlich) addieren und Summe ausgeben
- Anzahl der Schleifendurchläufe vorgegeben
- Summe wird in einer Variable akkumuliert (Akkumulierende Schleife)

```java
Scanner scanner = new Scanner(System.in);
while (scanner.hasNext()) {
    if (scanner.hasNextInt()) {
        System.out.println(scanner.nextInt());
    } else {
        System.out.println("Failure: " + scanner.next());
    }
}
```

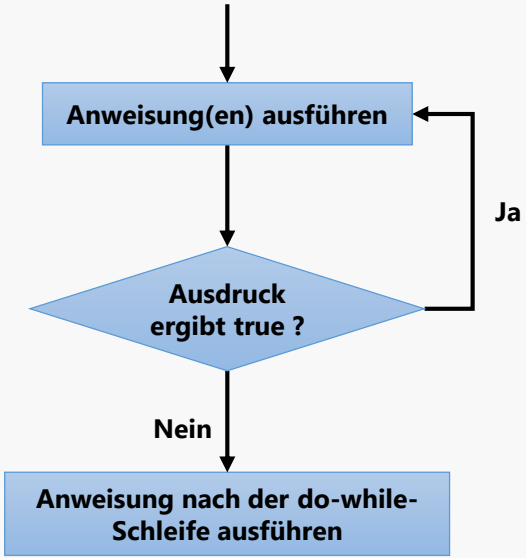
- Solange einlesen, solange Input vorhanden ist (Input-Schleife)
- Anzahl der Schleifendurchläufe ist nicht bekannt
- Abbruch der Schleife wird durch Input bestimmt

do-while (1)

- Allgemeine Form

```
do
    Anweisung(en)
while (Ausdruck);
```

- Ausführung



do-while (2)

- Hinweis
  - Anweisung wird **mindestens einmal** ausgeführt
- Weitere Bezeichnungen
  - Durchlaufschleife
  - Nachprüfende oder fußgesteuerte Schleife

Beispiel (do-while)

```
public class DoWhileTest {

    public static void main(String[] args) {
        long i = 2;
        System.out.println("N \tN^2 \tN^3");
        do {
            System.out.println(i + "\t" + i * i + "\t" + i * i * i);
            i *= 2;
        } while (i <= 1024);
    }

}
```

N	N^2	N^3
2	4	8
4	16	64
8	64	512
16	256	4096
32	1024	32768
64	4096	262144
128	16384	2097152
256	65536	16777216
512	262144	134217728
1024	1048576	1073741824

Unterschied zwischen while und do-while

- Beispiel: Die Ziffern einer nicht negativen Zahl i in umgekehrter Reihenfolge ausgeben

Lösung mit while-Schleife	Lösung mit do-while-Schleife

…
`while (i > 0) {`
`    System.out.println(i % 10);`
`    i = i / 10;`
`}`
…
	…
	`do {`
	`    System.out.println(i % 10);`
	`    i = i / 10;`
	`} while (i > 0);`
	…

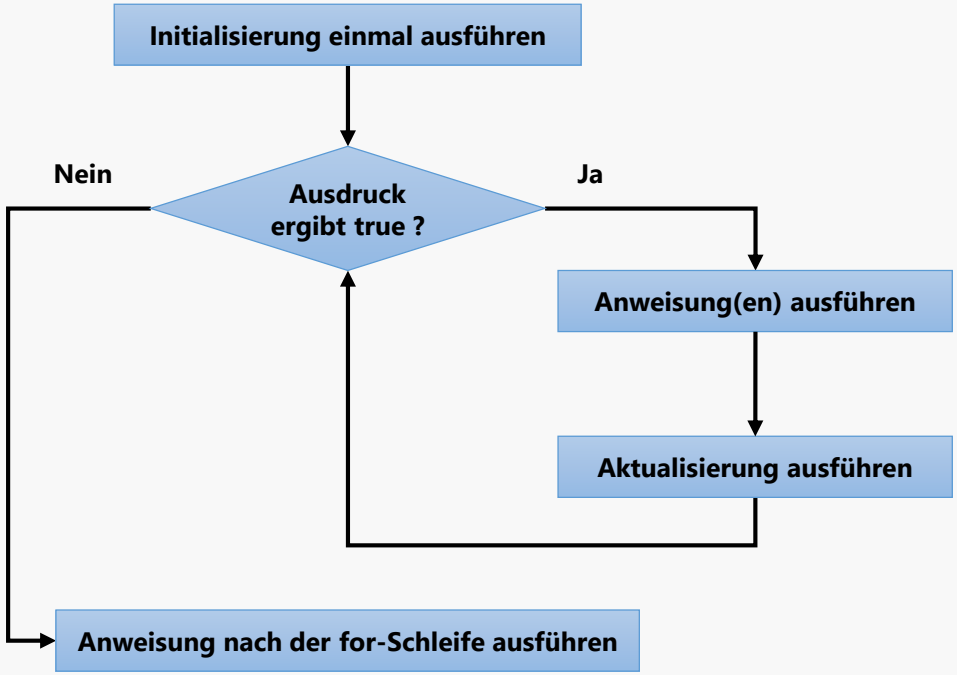
- Lösung mit `do-while`-Schleife ist korrekt
- Lösung mit `while`-Schleife hat ein Problem (0!)
  - Hinweis: i >= 0 löst hier nicht das Problem!

for (1)

- Allgemeine Form

```
for (Initialisierung; Ausdruck; Aktualisierung)
    Anweisung(en)
```

- Ausführung



for (2)

- Jeder der drei Teile im Schleifenkopf kann weggelassen werden
  - https://docs.oracle.com/javase/specs/jls/se15/html/jls-14.html#jls-14.14.1
- Weitere Bezeichnungen
  - Zählschleife (Sonderform der vorprüfenden Schleife)

Beispiel (for)

```java
public class ForTest {

    public static void main(String[] args) {
        System.out.println("N \tN^2 \tN^3");
        for (long i = 2; i <= 1024; i *= 2) {
            System.out.println(i + "\t" + i * i + "\t" + i * i * i);
        }
    }

}
```

```
N       N^2       N^3
2       4         8
4       16        64
8       64        512
16      256       4096
32      1024      32768
64      4096      262144
128     16384     2097152
256     65536     16777216
512     262144    134217728
1024    1048576   1073741824
```

Sichtbarkeit von Variablen

- Sichtbarkeitsbereich einer Variable
  - Programmstück, in dem auf die Variable zugegriffen werden kann
- Variable in einem Block (z. B. Block einer `for`-Schleife)
  - Sichtbar von der Deklaration bis zum Ende des Blocks, in dem sie deklariert wurde, z. B.

```
for (int i = 0; i < 100; i++) {
    …
    int x = 20;
    …
}
…
System.out.println(i + x);
```

Ab hier ist `i` in der Schleife bekannt

Ab hier ist x im aktuellen Schleifendurchlauf bekannt

Hier sind `i` und x **nicht mehr** bekannt

Gibt einen Übersetzungsfehler!

Weitere Beispiele für for-Schleifen

`for (int i = 0; i < 10; i++) {`
`  System.out.println(i);`
`}`
	Zahlen von 0 bis 9 auf der Konsole ausgeben

`for (int i = 10; i >= 0; i--) {`
`  System.out.println(i);`
`}`
	Zahlen von 10 bis 0 auf der Konsole ausgeben

`int sum = 0;`
`for (int i = 2; i <= 100; i += 2) {`
`  sum += i;`
`}`
`System.out.println(sum);`
	Alle geraden Zahlen von 2 (einschließlich) bis 100 (einschließlich) addieren und Summe ausgeben

`for (long i = 1; i < 1000000000000L; i *= 3) {`
`  System.out.println(i);`
`}`
	Alle Potenzen von 3, die kleiner als 10^{12} sind, ausgeben

Ein Beispiel – unterschiedliche for-Schleifen

- 10! berechnen

`int factorial = 1;`
`for (int i = 2; i < 11; i++) {`
`  factorial = factorial * i;`
`}`
	• Einfache, lesbare Variante
	• Typische for-Schleife

`int factorial, i;` `for (factorial = 1, i = 2; i < 11; factorial = factorial * i, i++);`	• Mehrere, durch Beistrich getrennte, Ausdrücke in der Initialisierung und Aktualisierung • Kein Schleifenrumpf notwendig • Schlechter lesbar
`for (int factorial = 1, i = 2; i < 11; factorial = factorial * i++);`	• Möglich, aber **sinnlos** • Variable `factorial` ist außerhalb der Schleife nicht verwendbar

`int factorial = 1, i = 2;`
`for (; i < 11;) {`
`  factorial = factorial * i++;`
`}`
	• Initialisierung und Aktualisierung bleiben leer
	• Initialisieren bei der Deklaration
	• Aktualisieren im Schleifenrumpf

Praktische Überlegungen

Schleifenrumpf

- Schleifenrumpf immer als Block implementieren
  - Klammern mit Einrückungen erhöhen Lesbarkeit
  - Schleifenrumpf kann leicht erweitert werden

Auswahl einer Schleife (1)

- `for`-Schleife
  - Vorgegebener Start- und Endwert
  - Fixe Schrittweite
- `while`-Schleife
  - Lediglich Endkriterium gegeben
  - Nicht vorhersagbar, wie oft die Schleifenanweisungen ausgeführt werden müssen, bis das Endkriterium zutrifft

Auswahl einer Schleife (2)

- Transformation
  - Die verschiedenen Schleifenarten lassen sich auch ineinander transformieren
- Anwendung
  - Abhängig vom Problem ist meist eine der Schleifen leichter ausformulierbar (lesbar)

Beispiele für Transformationen (beide Richtungen!)

while-Schleife	**entsprechende do-while-Schleife**

`while (ausdr)`
`    Schleifenanweisung;`
	`if (ausdr)`
	`    do`
	`        Schleifenanweisung;`
	`    while (ausdr);`

do-while-Schleife	**entsprechende while-Schleife**

`do`
`    Schleifenanweisung;`
`while (ausdr);`
	`Schleifenanweisung;`
	`while (ausdr)`
	`    Schleifenanweisung;`

for-Schleife	**entsprechende while-Schleife**

`for (ausdr1; ausdr2; ausdr3)`
`    Schleifenanweisung;`
	`ausdr1;`
	`while (ausdr2) {`
	`    Schleifenanweisung;`
	`    ausdr3;`
	`}`

ausdr1 hat auch nach der Schleife Auswirkung!

break und continue

- `break` bei Schleifen
  - Führt zum sofortigen Ausstieg aus dem aktuellen Schleifendurchlauf und damit zum Ausstieg aus der Schleife
- `continue`
  - Überspringt die restlichen Anweisungen im aktuellen Schleifendurchlauf und geht wieder zum Ausdruck
- Verwendung
  - In Schleifen **nur sparsam** verwenden
    - Kann übermäßige Verschachtelung vermeiden
    - Kann oft durch eine alternative Formulierung des Ausdrucks umgangen werden

Beispiel für break und seine Vermeidung

- Version mit break

```
for (int i = 0; i < 10; i++) {
    if (!scanner.hasNextLine()) {
        break;
    }
    System.out.println(i + ": " + scanner.nextLine());
}
```

- Version ohne break

```
for (int i = 0; i < 10 && scanner.hasNextLine(); i++) {
    System.out.println(i + ": " + scanner.nextLine());
}
```

Beispiel für continue und seine Vermeidung

- Version mit continue

```
while (scanner.hasNext()) {
    if (!scanner.hasNextInt()) {
        System.out.println("Wrong input : " + scanner.next());
        continue;
    }
    System.out.println(scanner.nextInt());
}
```

- Versionen ohne continue

```
while (scanner.hasNext()) {
    if (!scanner.hasNextInt()) {
        System.out.println("Wrong input : " + scanner.next());
    } else {
        System.out.println(scanner.nextInt());
    }
}
```

```
while (scanner.hasNext()) {
    if (scanner.hasNextInt()) {
        System.out.println(scanner.nextInt());
    } else {
        System.out.println("Wrong input : " + scanner.next());
    }
}
```

Verschachtelte Schleifen

Verschachtelte Schleifen

- Eine Schleife ist eine Anweisung
- Daher kann eine Schleife auch in einer Schleife vorkommen

Beispiel (Multiplikationstabelle, Angabe)

- Multiplikationstabelle für das kleine Einmaleins in der folgenden Form ausgeben

*	1	2	…	9	10
1	1	2	…	9	10
2	2	4	…	18	20
3	3	6	…	27	30
4	4	8	…	36	40
5	5	10	…	45	50
6	6	12	…	54	60
7	7	14	…	63	70
8	8	16	…	72	80
9	9	18	…	81	90
10	10	20	…	90	100

Beispiel (Multiplikationstabelle, Lösung)

```java
public class Multiplication {

    public static void main(String[] args) {
        int n = 10;
        System.out.print("*\t");
        for (int i = 1; i <= n; i++) {
            System.out.print("\t" + i);
        }
        System.out.println("\n");
        for (int i = 1; i <= n; i++) {
            System.out.print(i + "\t");
            for (int j = 1; j <= n; j++) {
                System.out.print("\t" + i * j);
            }
            System.out.println();
        }
    }

}
```

Beispiel (Verschachtelte Schleifen mit if-else)

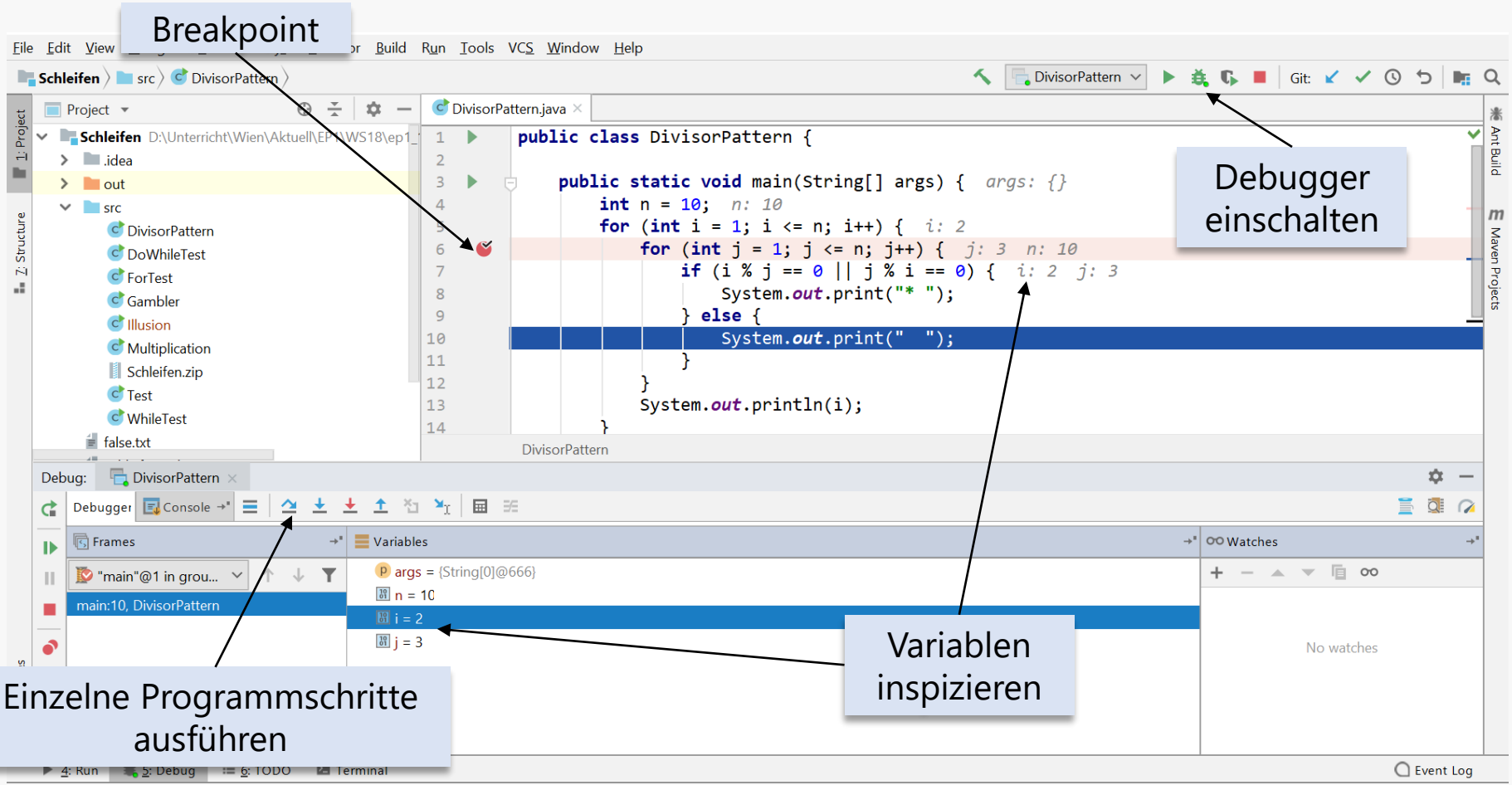
```
public class DivisorPattern {

    public static void main(String[] args) {
        int n = 10;
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    System.out.print("* ");
                } else {
                    System.out.print("  ");
                }
            }
            System.out.println(i);
        }
    }

}
```

```
* * * * * * * * * * 1
* *   *   *   *   * 2
*   *     *     *   3
* *   *       *     4
*       *         * 5
* * *     *         6
*           *       7
* *   *       *     8
*   *           *   9
* *     *         * 10
```

Debugging – ein erster Überblick

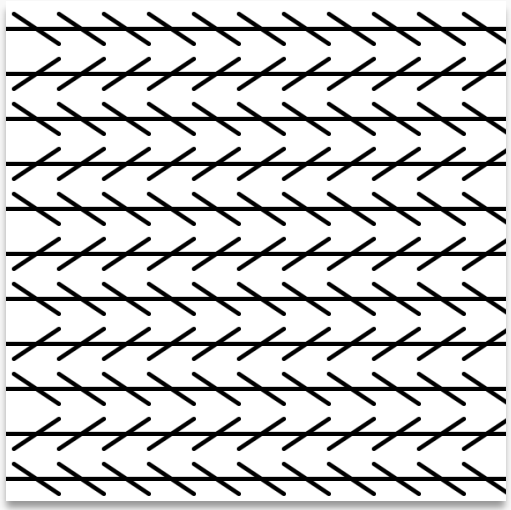
- Debugger
  - Werkzeug zum Diagnostizieren und Auffinden von Fehlern in Programmen
- Typische Funktionen
  - Steuerung des Programmablaufs (Haltepunkte, engl. Breakpoints)
  - Schrittweise Durchführung von Programmen
  - Inspizieren und modifizieren von Variablen

Debugging in IntelliJ



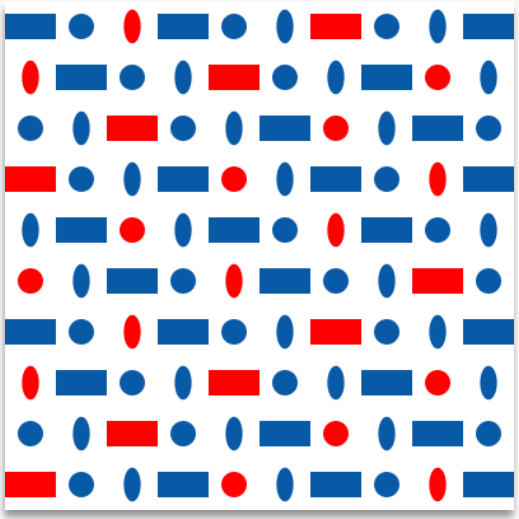
Beispiel (Optische Täuschung)

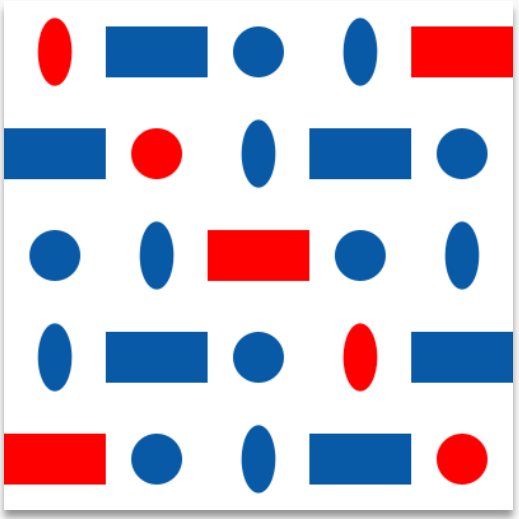
- Erstellen einer optischen Täuschung
- Schleifen
  - Horizontale Linien zeichnen
  - Pro horizontaler Linie schräge Linien zeichnen
  - Mehrere Zeilen mit abwechselndem Muster zeichnen
- Code in TUWEL
  - `Illusion.java`



Beispiel (Muster erzeugen)

- Muster aus unterschiedlichen Objekten
  - Kombination von Schleifen mit unterschiedlichen grafischen Objekten
- Code in TUWEL
  - `Pattern.java`





Zusammenfassung

- while, do-while, for
- break, continue
- Verschachtelte Schleifen
- Debugging