Schleifen

Einführung in die Programmierung 1

Wintersemester 21/22

Überblick

- Schleifen in Java
- Verschachtelte Schleifen
- Programmierstil
- Programmiermuster

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

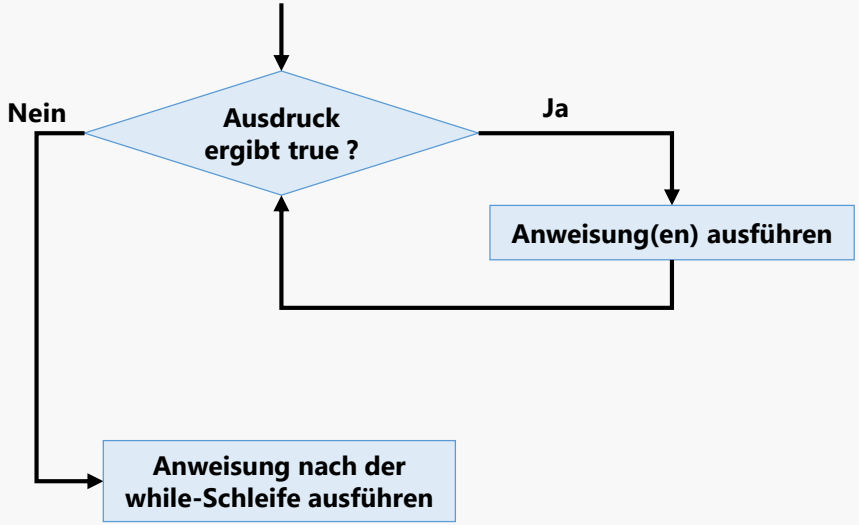
while	do-while	for
• Vor Schleifendurchlauf Ausdruck überprüfen • Dann Schleifenrumpf (Anweisung) ausführen	• Zuerst Schleifenrumpf (Anweisung) ausführen • Nach Schleifendurchlauf Ausdruck überprüfen	• Zuerst Vorbereitung (vor erstem Schleifendurchlauf) • Vor Schleifendurchlauf Ausdruck überprüfen • 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

```java
public class WhileTest {

    public static void main(String[] args) {
        int i = 0;
        System.out.println("Start:");
        while (i < 10) {
            System.out.println("i = " + i + " -> " + (i + 1) + ". Iteration");
            i++;
        }
        System.out.println("Ende!");
    }

}
```

```
Start:
i = 0 -> 1. Iteration
i = 1 -> 2. Iteration
i = 2 -> 3. Iteration
i = 3 -> 4. Iteration
i = 4 -> 5. Iteration
i = 5 -> 6. Iteration
i = 6 -> 7. Iteration
i = 7 -> 8. Iteration
i = 8 -> 9. Iteration
i = 9 -> 10. Iteration
Ende!
```

Beispiel (unbestimmte Anzahl an Schleifendurchläufen)

```
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

Beispiel (Einlesen)

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

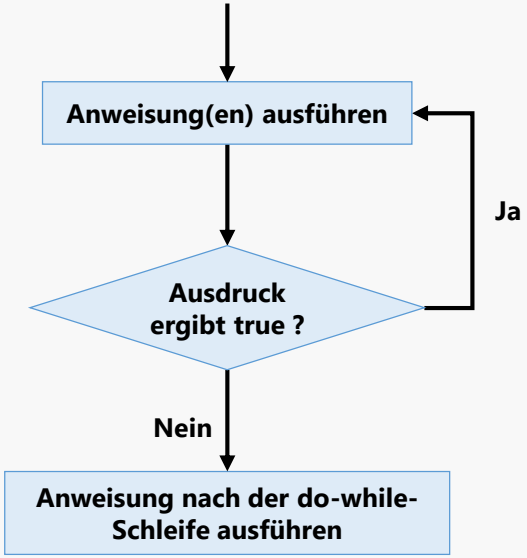
- Solange einlesen, solange Input vorhanden ist (Input-Schleife)
- Anzahl der Schleifendurchläufe ist nicht bekannt
- Abbruch der Schleife wird durch Input bestimmt

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) {
        int i = 0;
        System.out.println("Start:");
        do {
            System.out.println("i = " + i + " -> " + (i + 1) + ". Iteration");
            i++;
        } while (i < 10);
        System.out.println("Ende!");
    }

}
```

```
Start:
i = 0 -> 1. Iteration
i = 1 -> 2. Iteration
i = 2 -> 3. Iteration
i = 3 -> 4. Iteration
i = 4 -> 5. Iteration
i = 5 -> 6. Iteration
i = 6 -> 7. Iteration
i = 7 -> 8. Iteration
i = 8 -> 9. Iteration
i = 9 -> 10. Iteration
Ende!
```

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);`
	…

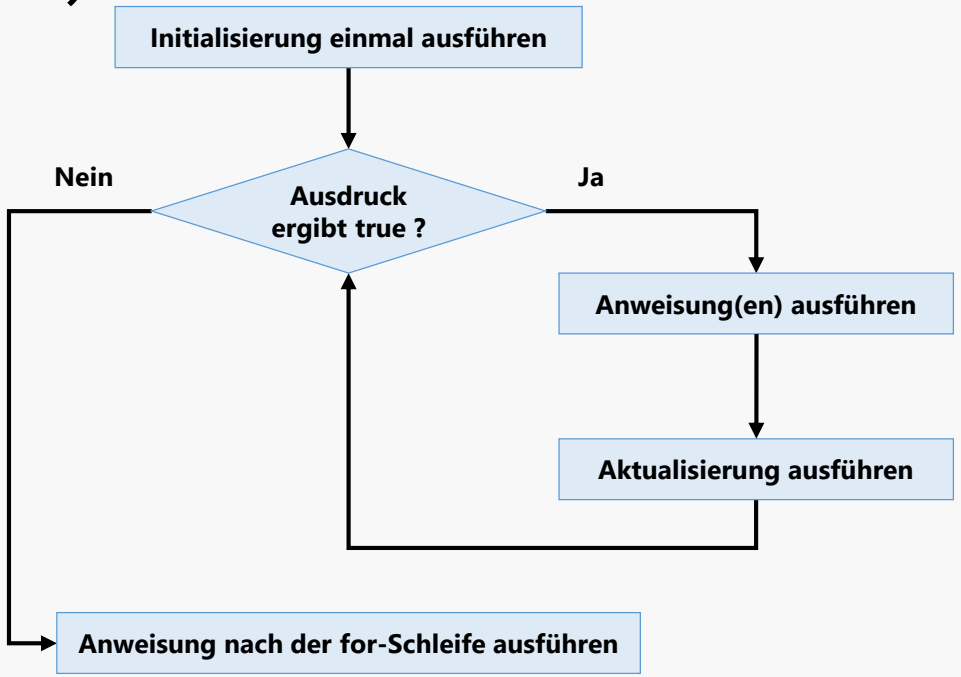
- Lösung mit `do-while`-Schleife ist korrekt
- Lösung mit `while`-Schleife hat ein Problem (0!)
  - Hinweis: i >= 0 löst hier nicht das Problem!

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/se17/html/jls-14.html#jls-14.14.1
- Weitere Bezeichnung
  - Zählschleife (Sonderform der vorprüfenden Schleife)

Beispiel (for)

```java
public class ForTest {

    public static void main(String[] args) {
        System.out.println("Start:");
        for (int i = 0; i < 10; i++) {
            System.out.println("i = " + i + " -> " + (i + 1) + ". Iteration");
        }
        System.out.println("Ende!");
    }

}
```

```
Start:
i = 0 -> 1. Iteration
i = 1 -> 2. Iteration
i = 2 -> 3. Iteration
i = 3 -> 4. Iteration
i = 4 -> 5. Iteration
i = 5 -> 6. Iteration
i = 6 -> 7. Iteration
i = 7 -> 8. Iteration
i = 8 -> 9. Iteration
i = 9 -> 10. Iteration
Ende!
```

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!

Beispiele (Zählschleifen)

- Aufsteigend

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

Zahlen von 0 bis 9 auf der Konsole ausgeben

- Absteigend

```
for (int i = 10; i >= 0; i--) {
    System.out.println(i);
}
```

Zahlen von 10 bis 0 auf der Konsole ausgeben

- Nicht triviales Inkrement

```
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 <= 10; i++) {`
`  factorial = factorial * i;`
`}`
	• Einfache, lesbare Variante
	• Typische for-Schleife

`int factorial, i;`
`for (factorial = 1, i = 2; i <= 10; 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 <= 10; factorial = factorial * i++);`	• Möglich, aber **sinnlos** • Variable `factorial` ist außerhalb der Schleife nicht verwendbar

`int factorial = 1, i = 2;`
`for (; i <= 10;) {`
`  factorial = factorial * i++;`
`}`
	• Initialisierung und Aktualisierung bleiben leer
	• Initialisieren bei der Deklaration
	• Aktualisieren im Schleifenrumpf

Beispiel (ein kleines Ratespiel)

- Ratespiel
  - Computer wählt zufällig eine Zahl im Bereich von 1 bis 100 aus
  - Person kann eine Zahl raten
    - Eingabe über `System.in`
    - Wenn falsch
      - Ausgabe, ob die Eingabe zu klein, zu groß oder keine korrekte Zahl ist
    - Wenn richtig
      - Ausgabe einer Gewinnmeldung und Ende des Programms
- Code in TUWEL
  - `GuessingGame.java`

break und continue

- break
  - 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

break und continue – Vergleich

```
int n = 5;
for (int i = 1; i < n; i++) {
    if (i % 5 == 0) {
        break;
    }
    System.out.println(i);
}
```

Wenn i den Wert 5 hat, wird die Schleife abgebrochen

```
1
2
3
4
```

```
int n = 5;
for (int i = 1; i < n; i++) {
    if (i % 5 == 0) {
        continue;
    }
    System.out.println(i);
}
```

Wenn i den Wert 5 hat, wird der Schleifendurchlauf abgebrochen

```
1
2
3
4
6
7
8
9
```

Beispiel (Prinzahltest mit vorzeitigem Abbruch)

```java
import java.util.Scanner;

public class PrimeTest {

    public static void main(String[] args) {
        Scanner inputScannner = new Scanner(System.in);
        System.out.print("Please enter a number: ");
        long n = inputScannner.nextLong();
        boolean isPrime = n >= 2;
        for (long factor = 2; factor < n; factor++) {
            if (n % factor == 0) {
                isPrime = false;
                break;
            }
        }
        if (isPrime) {
            System.out.println(n + " is prime");
        } else {
            System.out.println(n + " is not prime");
        }
    }

}
```

An dieser Stelle ist das Ergebnis klar und die Schleife muss nicht weiter ausgeführt werden

Verschachtelte Schleifen

Verschachtelte Schleifen

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

Beispiel (2D-Muster auf der Konsole ausgeben)

```
int n = 3;
int m = 5;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

```
*****
*****
*****
```

Entwicklung der Zählvariablen

```
i:0 j:0
i:0 j:1
i:0 j:2
i:0 j:3
i:0 j:4

i:1 j:0
i:1 j:1
i:1 j:2
i:1 j:3
i:1 j:4

i:2 j:0
i:2 j:1
i:2 j:2
i:2 j:3
i:2 j:4
```

Weitere Beispiele für Ausgaben

```
int n = 5;
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

```
*****
****
***
**
*
```

```
int n = 5;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.print("*");
    }
    for (int j = i; j < n; j++) {
        System.out.print("#");
    }
    System.out.println();
}
```

```
#####
*####
**###
***##
****#
```

```
int n = 5;
for (int i = 0; i < n; i++) {
    for (int j = 0; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

```
*
**
***
****
*****
```

```
int n = 5;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print(i + j);
    }
    System.out.println();
}
```

```
01234
12345
23456
34567
45678
```

Beispiel (Multiplikationstabelle, Angabe)

- Eine Multiplikationstabelle für das kleine Einmaleins in der folgenden Form ausgeben

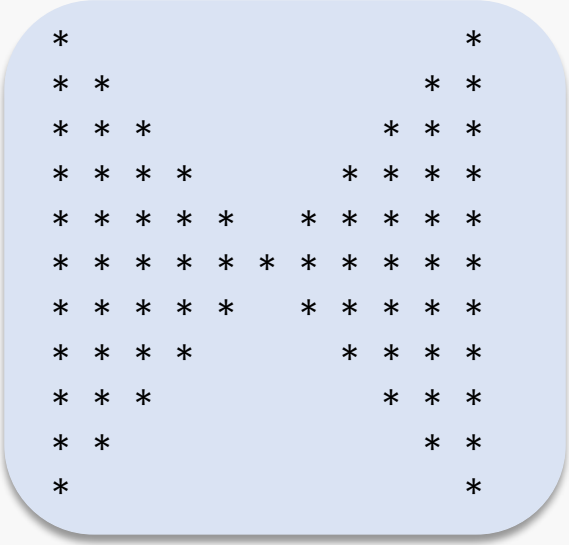
*	1	2	…	9	10
1	1	2	…	9	10
2	2	4	…	18	20
3	3	6	…	27	30
4	4	8	…	36	40
5	5	10	…	45	50
6	6	12	…	54	60
7	7	14	…	63	70
8	8	16	…	72	80
9	9	18	…	81	90
10	10	20	…	90	100

Beispiel (Multiplikationstabelle, Lösung)

```
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();
        }
    }

}
```

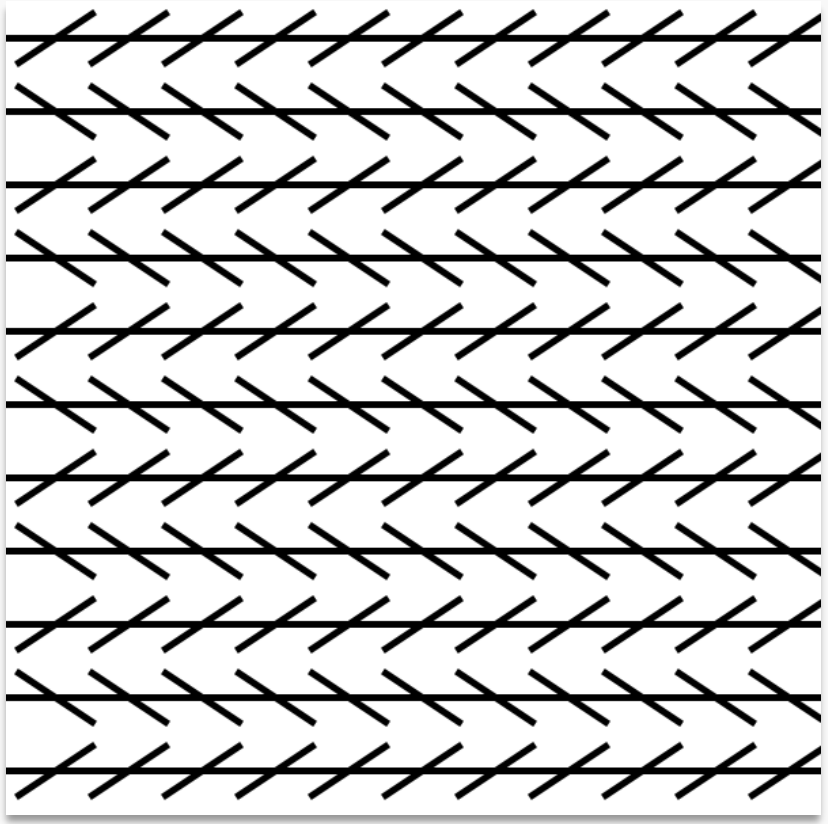
Beispiel (Verschachtelte Schleifen mit if-else)

```
public class StarPattern {

    public static void main(String[] args) {
        int n = 5;
        for (int i = -n; i <= n; i++) {
            for (int j = -n; j <= n; j++) {
                if (i * i <= j * j) {
                    System.out.print("* ");
                } else {
                    System.out.print("  ");
                }
            }
            System.out.println();
        }
    }

}
```



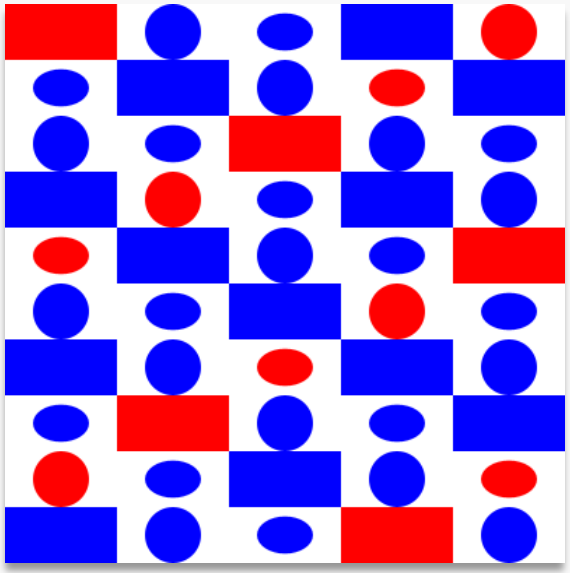
Beispiel (Optische Täuschung)

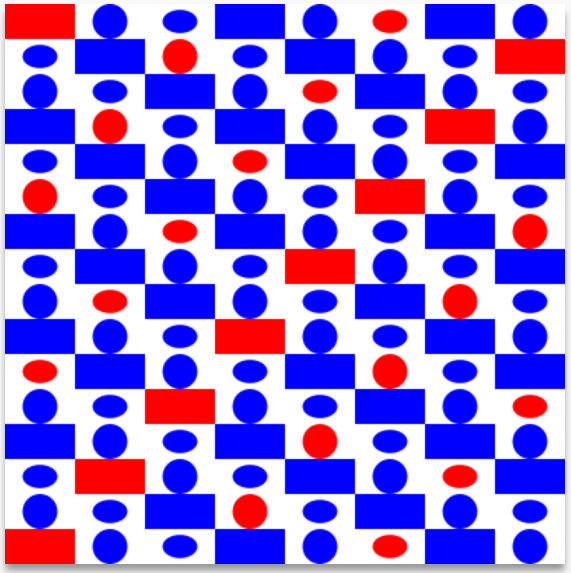
- Schleifen
  - Horizontale Linien zeichnen
  - Pro horizontaler Linie schräge Linien zeichnen
  - Mehrere Zeilen mit abwechselndem Muster zeichnen
- Code in TUWEL
  - `Illusion.java`



Beispiel (Muster mit CodeDraw erzeugen)

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



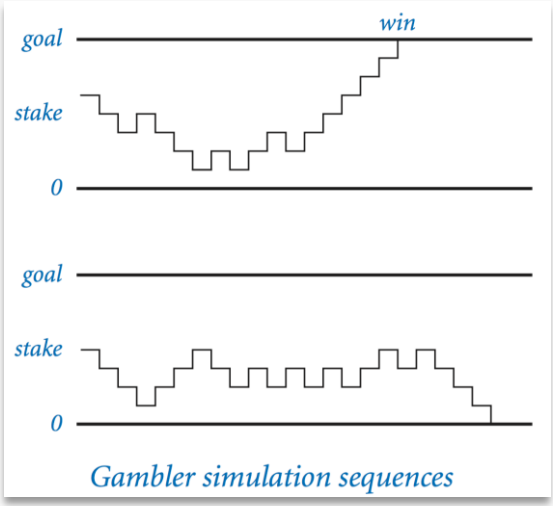


Beispiel (Simulation)

- Spiel beginnt mit einem Startbetrag (stake)
- In jeder Runde wird 1 Euro eingesetzt
  - 50 % Chance auf Gewinn von einem Euro (Einsatz zurück + 1 Euro)
  - 50 % Chance auf Verlust des Einsatzes (- 1 Euro)
- Es wird am Anfang ein Zielbetrag (goal) festgelegt
  - Wird dieser erreicht, dann wird das Spiel beendet

Beispiel (Simulation)

- Simulation
  - Zufälliger Entscheid, ob Gewinn oder Verlust
  - Abbruch, wenn Zielbetrag erreicht oder kein Geld mehr vorhanden
  - Mehrmals wiederholen (trials) und Mittelwerte berechnen
- Code in TUWEL
  - `GamblingSimulation.java`



https://introcs.cs.princeton.edu/java/13flow/images/gambler.png

Programmierstil

Schleifenrumpf

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

break und continue

- In Schleifen **nur sparsam** verwenden
- Können übermäßige Verschachtelungen vermeiden
- Können oft durch eine alternative Formulierung des Ausdrucks umgangen werden

Beispiel für break und seine Vermeidung (1)

- Mit break

```
Scanner input = new Scanner(System.in);
System.out.print("Please enter an integer > 1: ");
int n = input.nextInt();
int factor = 2;
while (factor <= n) {
    if (n % factor == 0) {
        break;
    }
    factor++;
}
System.out.println("The smallest factor other than 1 for " + n + " is " + factor);
```

Abbruch, wenn ein Faktor != 1 gefunden wurde. Mit break kürzer als untere Version.

- Ohne break

```
Scanner input = new Scanner(System.in);
System.out.print("Please enter an integer > 1: ");
int n = input.nextInt();
int factor = 2;
boolean found = false;
while (factor <= n && !found) {
    if (n % factor == 0) {
        found = true;
    } else {
        factor++;
    }
}
System.out.println("The smallest factor other than 1 for " + n + " is " + factor);
```

Version ohne break. Benötigt eine zusätzliche Variable und ist textuell länger. Dafür kann der Ausstieg aus der Schleife nur an einer Stelle geschehen (Überprüfung der Schleifenbedingung).

Beispiel für break und seine Vermeidung (2)

- Anderer Schleifenansatz (kein break erforderlich)

```
Scanner input = new Scanner(System.in);
System.out.print("Please enter an integer > 1: ");
int n = input.nextInt();
int factor = 2;
while (n % factor != 0) {
    factor++;
}
System.out.println("The smallest factor other than 1 for " + n + " is " + factor);
```

- Nicht immer so einfach möglich!

Beispiel für continue und seine Vermeidung

- Mit continue

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

- 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());
    }
}
```

Programmiermuster

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;`
	`}`

Akkumulierende Schleifen

- Schleifen, mit denen ein Ergebnis konstruiert wird
- Schema
  - Akkumulierende Variable vor der Schleife initialisieren
  - Akkumulierende Variable in der Schleife verändern
  - Nach der Schleife Ergebnis weiterverwenden
- Beispiele

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

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

```
String text = "Hello";
String result = "#";
for (int i = 0; i < text.length(); i++) {
    result += text.charAt(i) + "#";
}
System.out.println(result);
```

Zaunpfahlproblem – Motivation

- Motivierendes Beispiel
  - Die Zahlen von 1 bis n durch ein Komma getrennt ausgeben
- Falsche Lösung

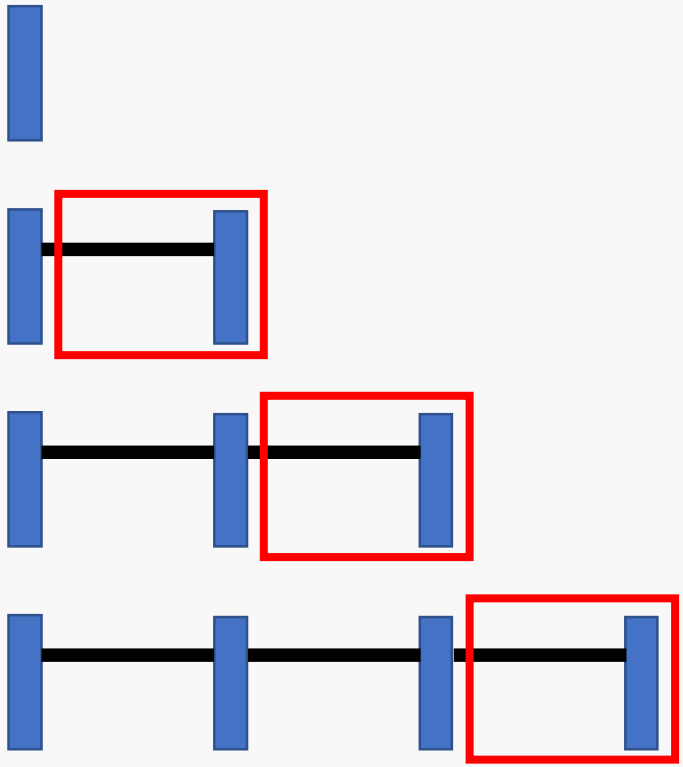
```
int n = 5
for (int i = 1; i <= n; i++) {
    System.out.print(", " + i);
}
System.out.println();
```

```
, 1, 2, 3, 4, 5
```

- Problem
  - n Zahlen werden ausgegeben, aber nur n-1 Kommas werden benötigt
  - `System.out.print(i + ", ");` löst nicht das Problem!
    - Ergibt `1, 2, 3, 4, 5,`

Zaunpfahlproblem – Analogie

- Analogie Zaun
  - 4 Pfähle, 3 Streben
- Aufbau
  - Einen Pfahl fixieren
  - Wiederhole
    - Strebe + nächsten Pfahl hinzugeben



Zaunpfahlproblem – Lösungen

- 1. Variante
  - Vor der Schleife Start fixieren

```
int n = 5;
System.out.print(1);
for (int i = 2; i <= n; i++) {
    System.out.print(", " + i);
}
System.out.println();
```

- 2. Variante
  - Nach der Schleife abschließen

```
int n = 5;
for (int i = 1; i <= n - 1; i++) {
    System.out.print(i + ", ");
}
System.out.println(n);
```

- 3. Variante
  - Mit Bedingung Ausgabe steuern

```
int n = 5;
for (int i = 1; i <= n; i++) {
    System.out.print(i);
    if (i < n) {
        System.out.print(", ");
    }
}
System.out.println();
```

```
1, 2, 3, 4, 5
```

Schleifen mit einem Sentinel (Hinweiszeichen)

- Variation des Zaunpfahlproblems
- Eine Folge von Zahlen einlesen
  - Abbruch, wenn ein bestimmter Wert (Sentinel) eingegeben wird
  - Sentinel sollte nicht in das Ergebnis miteinbezogen werden
- Vorgehensweise
  - Wert einlesen
  - Schleife
    - Wert verarbeiten
    - Wert einlesen

Beispiel (Sentinel)

```
int sum = 0;
Scanner input = new Scanner(System.in);
System.out.println("Next integer (-1 to quit): ");
int number = input.nextInt();
while (number != -1) {
    sum += number;
    System.out.println("Next integer (-1 to quit): ");
    number = input.nextInt();
}
System.out.println(sum);
```

```
Next integer (-1 to quit):
3
Next integer (-1 to quit):
4
Next integer (-1 to quit):
5
Next integer (-1 to quit):
-1
12
```

```
int sum = 0;
Scanner input = new Scanner(System.in);
while (true) {
    System.out.println("Next integer (-1 to quit): ");
    int number = input.nextInt();
    if (number == -1) {
        break;
    }
    sum += number;
}
System.out.println(sum);
```

Variante mit Endlosschleife und break – solange einlesen bis Sentinel eingegeben wird. Eingabeprompt muss nur einmal implementiert werden

```
Next integer (-1 to quit):
3
Next integer (-1 to quit):
4
Next integer (-1 to quit):
5
Next integer (-1 to quit):
-1
12
```

Zusammenfassung

- Schleifen in Java
- Verschachtelte Schleifen
- Programmierstil
- Programmiermuster