
Java™

Wiederholung

- So sieht ein „leeres“ Java-Programm aus

```
public class Programmname {

    public static void main (String[] args) {

        // Hier stehen die Anweisungen

    }

}
```

Wiederholung

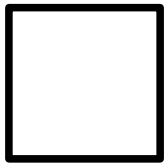
- Anweisungen mit Variablen

  - Deklaration einer Variablen (muss initialisiert werden)
    ```
    Datentyp variablenName;
    ```

  - Deklaration und Initialisierung einer Variablen
    ```
    Datentyp variablenName = Wert;
    ```

  - Zuweisung eines Wertes zu einer Variablen
    ```
    variablenName = Ausdruck;
    ```

Variablen deklarieren



```
Datentyp variablenname;
```

Wertzuweisung


```
variablenname = wert;
```

Wiederholung

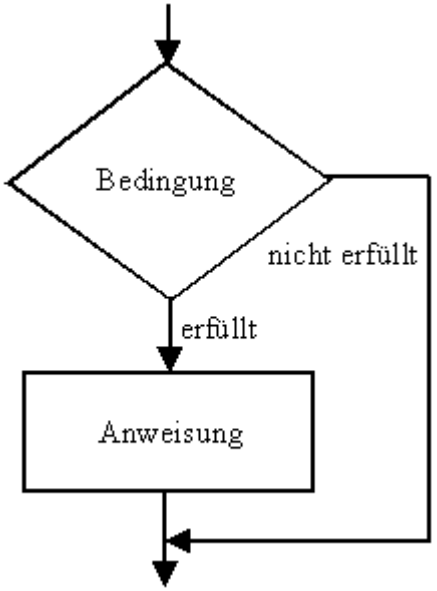
- Anweisungen durch Methodenaufrufe
  - Ausgabe auf der Konsole
    ```
    System.out.println(...);
    ```
  - Benutzereingabe einlesen
    ```
    … = scanner.nextDatentyp();
    ```
  - Mathematische Funktionen
    ```
    Math.sqrt(…), Math.sin(…), Math.random(), usw.
    ```

Kontrollstrukturen

- Sequenz
  - Eine Anweisung wird nach der anderen ausgeführt
- Selektion
  - Anweisung wird in Abhängigkeit einer Bedingung ausgeführt
- Iteration
  - Anweisung wird mehrfach wiederholt ausgeführt

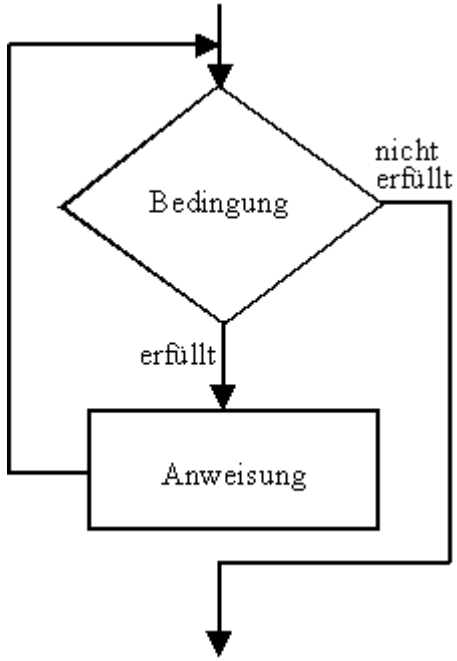
Selektion

```
if (...) {
   ...
}
```



Iteration

```
while (…) {
   …
}
```



Java Methoden

Math Methoden

- Konstanten
  - Eulersche Zahl `Math.E`
  - Kreiszahl `Math.PI`
- Methoden
  - Absolutwert (Betrag, positiver Wert) einer Zahl `Math.abs`
  - Arkuskosinus `Math.acos`
  - Arkussinus `Math.asin`
  - Arkustangens `Math.atan`
  - Arkustangens mit zwei Argumenten `Math.atan2`
  - Kubikwurzel `Math.cbrt`
  - Aufrunden `Math.ceil`
  - … (http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html)

Methoden

- Methodenkopf
  - Name
  - 0, 1 oder mehrere Parameter-Datentypen
  - 0 oder 1 Rückgabe-Datentyp
- Methodenrumpf
  - Anweisungen

Methoden Beispiel 1

- Methodenkopf
  - Name `Math.sqrt`
  - 1 Parameter-Datentyp `double`
  - 1 Rückgabe-Datentyp `double`
- Methodenrumpf
  - Berechne die Wurzel aus dem angegebenen Parameter und gib diese Zurück
- Beispiel
  - ```
    double quadrat = 81;
    double wurzel = Math.sqrt(quadrat);
    System.out.println(wurzel);
    ```
    ```

Methoden Beispiel 2

- Methodenkopf
  - Name `Math.cos`
  - 1 Parameter-Datentyp `double`
  - 1 Rückgabe-Datentyp `double`
- Methodenrumpf
  - Berechne den Kosinus des angegebenen Parameters und gib diesen zurück
- Beispiel
  - ```
    double winkel = Math.PI/4;
    double hypotenuse = Math.sqrt(2);
    double ankathete = Math.cos(winkel)*hypotenuse;
    System.out.println(ankatete);
    ```
    ```

Methoden Beispiel 3

- Methodenkopf
  - Name `Math.pow`
  - 2 Parameter-Datentypen `double, double`
  - 1 Rückgabe-Datentyp `double`
- Methodenrumpf
  - Berechne die Potenz (erster Parameter hoch zweiter Parameter) und gib das Ergebnis zurück
- Beispiel
  - ```
    double basis = 5;
    double exponent = 3;
    double potenz = Math.pow(basis,exponent);
    System.out.println(potenz);
    ```
    ```

Methoden Beispiel 4

- Methodenkopf
  - Name `Math.random`
  - 0 Parameter-Datentypen
  - 1 Rückgabe-Datentyp `double`
- Methodenrumpf
  - Erzeuge eine Zufallszahl im Intervall [0,1) und gib diese zurück
- Beispiel
  - ```
    double zahl = Math.random();
    System.out.println(zahl);
    ```
    ```

Methoden Beispiel 5

- Methodenkopf
  - Name `System.out.println`
  - 1 Parameter-Datentyp `String`
  - 0 Rückgabe-Datentyp
- Methodenrumpf
  - Gib den angegebenen Parameter auf der Konsole aus (keine Rückgabe)
- Beispiel
  - `System.out.println("Hello World");`

Überladene Methoden

- Methodenkopf
  - Name `Math.round`
  - 0, 1 oder mehrere Parameter-Datentypen `float`
  - 0 oder 1 Rückgabe-Datentyp `int`
- Methodenrumpf
  - Runde die als Parameter angegebene Kommazahl und gib den gerundeten Wert als Ganzzahl zurück
- Beispiel
  - ```
    float ungerundet = 2.7f;
    int gerundet = Math.round(ungerundet);
    System.out.println(gerundet);
    ```
    ```

Überladene Methoden

- Methodenkopf
  - Name `Math.round`
  - 0, 1 oder mehrere Parameter-Datentypen `double`
  - 0 oder 1 Rückgabe-Datentyp `long`
- Methodenrumpf
  - Runde die als Parameter angegebene Kommazahl und gib den gerundeten Wert als Ganzzahl zurück
- Beispiel
  - ```
    double ungerundet = 2.7;
    long gerundet = Math.round(ungerundet);
    System.out.println(gerundet);
    ```
    ```

Überladene Methoden

- Methoden mit …
  - Gleichem Namen
  - Unterschiedlichen Parameter-Datentypen (in Art und/oder Anzahl)

… sind unterschiedliche Methoden!

  - Können unterschiedliche Rückgabe-Datentypen oder -Werte haben
  - Können sich im Methodenrumpf unterscheiden

Weitere Methoden

- Systemzeit in Nanosekunden
  - `System.nanoTime`
- Formatierte Ausgabe
  - `System.out.printf`
- Java Application Program Interface (API)
  - http://docs.oracle.com/javase/8/docs/api/
  - > 10 000 Methoden
- Frameworks, Libraries, Tools
  - Externe Bibliotheken
  - > 1 000 000 Methoden

Formatierte Ausgabe

- System.out.printf
  - 1. Parameter: String
    mit Platzhaltern (`%d (int)`, `%f (float)`, `%s (String)`, ...)
  - (Anzahl Platzhalter viele) Parameter, die die Platzhalter ersetzen
- Formatierung
  - Zum Beispiel Stellen vor (`V`) und hinterm (`H`) Komma: `%V.Hf`
- Beispiel
  - ```
    System.out.printf
    ("%s WiSe %d/%d Note %.1f","Informatik",2015,2016,1);
    ```
  - Ausgabe: `Informatik WiSe 2015/2016 Note 1.0`
- Weitere Infos
  - https://docs.oracle.com/javase/tutorial/java/data/numberformat.html
    ```

Java Struktur

- Projekte / Bibliotheken
  - `Informatik`
  - `JRE System Library`
- Pakete (und Unterpakete)
  - `uebung01`
  - `java`
  - `java.util`
- Klassen
  - `java.lang.Math`
  - `java.util.Scanner`
  - `HelloWorld`
- Methoden
  - `java.lang.Math.sqrt`
  - `main`

Problemlösung

Problemlösung

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das A, B und C tut

- Lösung
  - Zerlegen der Aufgabe in mehrere kleine (hoffentlich einfachere) Teilaufgaben A, B und C
  - Falls möglich weiter zerlegen

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Zerlegung
  - Einlesen
  - Sortieren
  - Ausgeben

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Zerlegung mit Methoden (kommt später, wie das geht)

```
zahlen = einlesen();
sortieren(zahlen);
ausgeben(zahlen);
```

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Einlesen
  - Meldung ausgeben, dass erste Zahl eingegeben werden soll
  - Erste Zahl vom Benutzer einlesen
  - Meldung ausgeben, dass zweite Zahl eingegeben werden soll
  - Zweite Zahl vom Benutzer einlesen
  - Meldung ausgeben, dass dritte Zahl eingegeben werden soll
  - Dritte Zahl vom Benutzer einlesen

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Einlesen

```
System.out.println("Geben Sie die erste Zahl ein: ");
```

  - Erste Zahl vom Benutzer einlesen
  - Meldung ausgeben, dass zweite Zahl eingegeben werden soll
  - Zweite Zahl vom Benutzer einlesen
  - Meldung ausgeben, dass dritte Zahl eingegeben werden soll
  - Dritte Zahl vom Benutzer einlesen

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Einlesen

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

  - Meldung ausgeben, dass zweite Zahl eingegeben werden soll
  - Zweite Zahl vom Benutzer einlesen
  - Meldung ausgeben, dass dritte Zahl eingegeben werden soll
  - Dritte Zahl vom Benutzer einlesen

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Einlesen

```
System.out.println("Geben Sie die erste Zahl ein: ");
int zahl1 = scanner.nextInt();
System.out.println("Geben Sie die zweite Zahl ein: ");
int zahl2 = scanner.nextInt();
System.out.println("Geben Sie die dritte Zahl ein: ");
int zahl3 = scanner.nextInt();
```

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Sortieren
  - Minimum bestimmen
  - Minimum mit erster Zahl tauschen
  - Es verbleiben noch zwei Zahlen, die sortiert werden müssen
  - Prüfen, ob zweite oder dritte Zahl kleiner ist
  - Die kleinere von beiden Zahlen an die zweite Position tauschen

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Minimum bestimmen
  - Ist Zahl 1 kleiner als Zahl 2 und kleiner als Zahl 3 ?
    Dann ist Zahl 1 das Minimum
  - Ist Zahl 2 kleiner als Zahl 1 und kleiner als Zahl 3 ?
    Dann ist Zahl 2 das Minimum
  - Ist Zahl 3 kleiner als Zahl 1 und kleiner als Zahl 2 ?
    Dann ist Zahl 3 das Minimum

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Zwei Zahlen a und b vertauschen
  - Lege Hilfsvariable an
  - Setze den Wert der Hilfsvariablen auf den Wert von a
    - Der Wert von a ist nun gesichert
  - Setze den Wert von a auf den Wert von b
  - Setze den Wert von b auf den Wert der Hilfsvariablen

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Minimum bestimmen und mit erster Zahl tauschen
  - Das ist nur nötig, wenn Zahl 2 oder Zahl 3 das Minimum ist

```
if (zahl2 < zahl1 && zahl2 < zahl3) {
    int h = zahl1;
    zahl1 = zahl2;
    zahl2 = h;
}
if (zahl3 < zahl1 && zahl3 < zahl2) {
    int h = zahl1;
    zahl1 = zahl3;
    zahl3 = h;
}
```

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Sortieren
  - Minimum bestimmen
  - Minimum mit erster Zahl tauschen
  - Es verbleiben noch zwei Zahlen, die sortiert werden müssen
  - Prüfen, ob zweite oder dritte Zahl kleiner ist
  - Die kleinere von beiden Zahlen an die zweite Position tauschen
  - Dann ist Zahl 1 die kleinste und Zahl 3 die größte Zahl

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Sortieren
  - Minimum bestimmen
  - Minimum mit erster Zahl tauschen

```
if (zahl3 < zahl2) {
   int h = zahl3;
   zahl3 = zahl2;
   zahl2 = h;
}
```

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Ausgabe
  - Meldung, dass Ausgabe kommt
  - Zahl 1 ausgeben
  - Zahl 2 ausgeben
  - Zahl 3 ausgeben

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Ausgabe

```
System.out.println("Sortiert: ");
```

  - Zahl 1 ausgeben
  - Zahl 2 ausgeben
  - Zahl 3 ausgeben

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Ausgabe

```
System.out.println("Sortiert: ");
System.out.println(zahl1);
```

  - Zahl 2 ausgeben
  - Zahl 3 ausgeben

Beispiel

- Aufgabenstellung
  - Schreiben Sie ein Java-Programm, das drei Zahlen vom Benutzer einliest und anschließend aufsteigend sortiert ausgibt

- Ausgabe

```
System.out.println("Sortiert: ");
System.out.println(zahl1);
System.out.println(zahl2);
System.out.println(zahl3);
```

Studienleistung

- Nächste Prüfung des Levels im Testat
  - 30. November
  - Mindestens **Level 7** erreichen

Eigene Methoden

Was sind Methoden?

- Vordefinierte Anweisungen zum vereinfachten Aufruf
  - Beispiele

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

double x = Math.sqrt(2);

long time = System.nanoTime();
```

- Wir behandeln zunächst nur statische Methoden (static)
  - Werden auch Funktionen genannt

Wozu Methoden?

- Wiederkehrende Programmteile
  - Wenn benötigt, kann der Teil wieder verwendet werden
  - Kein Copy-Paste, bei Änderung wird jedes Auftreten geändert
- Übersichtlichkeit
  - Komplexe Programme in Teilprogramme zerlegen
  - Programmcode in unterschiedlichen Dateien speichern
- Nachteil
  - Methodenaufrufe kosten Ressourcen (Zeit + Speicherplatz)

    (Ist nur bei zeitkritischen Anwendungen ein Nachteil, die man ohnehin besser in einer anderen Programmiersprache schreibt)

Methoden

- Methodenkopf (Schnittstelle mit der Außenwelt)
  - Name
  - 0, 1 oder mehrere Parameter-Datentypen
  - 0 oder 1 Rückgabe-Datentyp
- Methodenrumpf
  - Anweisungen

Methoden

- Deklaration einer Methode
  - `public static RückgabeDatentyp name (Parameterliste) {`
    `Anweisungen`
    `}`
  - Rückgabe-Datentyp
    - Beliebiger Datentyp oder `void`, falls keine Rückgabe erfolgt
  - Name
    - Klein Schreiben (wie bei Variablen)
  - Parameterliste
    - Datentyp parameterName
    - Mit Komma getrennt (falls mehrere Parameter verwendet werden)
  - Anweisungen
    - Beliebige Anweisungen (siehe Kontrollstrukturen)
    - Letzte Anweisung muss `return`-Statement sein

Beispiel

- Schreiben Sie ein Java-Programm „Romanify“, das eine ganze Zahl vom Benutzer einliest und anschließend die entsprechenden römischen Ziffern ausgibt
- Idee

```
public class Romanify {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);

        int number = scanner.nextInt();
        String numeral = romanNumeral(number);
        System.out.println(numeral);

        scanner.close();
    }
}
```

Beispiel

- Eigene Methode

```
public class Romanify {

    public static String romanNumeral (int arabic) {
        …
        return "…";
    }

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

        int number = scanner.nextInt();
        String numeral = romanNumeral(number);
        System.out.println(numeral);

        scanner.close();
    }
}
```

Beispiel

- Oder umgekehrt: Aus einer römischen eine arabische Zahl?

```
public class Arabify {

    public static int arabicNumeral (String roman) {
        …
        return 0;
    }

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

        String text = scanner.nextLine();
        int number = arabicNumeral(text);
        System.out.println(number);

        scanner.close();
    }
}
```

Methoden schreiben

- Deklaration einer Methode
  - `public static RückgabeDatentyp name (Parameterliste) {`
    `Anweisungen`
    `}`
- Anweisungen (Methodenrumpf)
  - Beliebige Anweisungen möglich
    - Methodenaufrufe, Deklarationen und Wertzuweisungen
    - Kontrollstrukturen (Selektion, Iteration)
  - Das `return`-Statement beendet die Methode
    - Alle Anweisungen hinter diesem Statement werden nicht mehr beachtet
  - Mehrere `return`-Statements sind möglich
    - Je nach Kontrollstruktur beendet das erste Statement die Methode
  - Datentyp des Ausdrucks hinter dem Schlüsselwort `return` muss dem Rückgabe-Datentyp entsrpechen

Beispiel

- Legen Sie eine neue Klasse „Numeral“ mit einer Methode „letterToNumber“ an, die zu einem einzelnen Zeichen der römischen Zahlen (char) den Wert (int) zurück gibt

```
public class Numeral {
    public static int letterToNumber(char letter) {
        switch (letter) {
            case 'I': return 1;
            case 'V': return 5;
            case 'X': return 10;
            case 'L': return 50;
            case 'C': return 100;
            case 'D': return 500;
            case 'M': return 1000;
            default: return 0;
        }
    }
}
```

Methoden aufrufen

```
public class Test {

    public static void main (String[] args) {

        int x = Numeral.letterToNumber('X');

        System.out.println(x);

        double sqrt = Math.sqrt(x);

        System.out.println(sqrt);

    }

}
```

Beispiel

```
public class RockPaperScissors {
   public static String symbol(int nr) {
      if (nr == 1)
         return "Rock";
      if (nr == 2)
         return "Paper";
      if (nr == 3)
         return "Scissors";
      return "ERROR";
   }
   public static void main (String[] args) {
      Scanner scanner = new Scanner(System.in);

      int user = scanner.nextInt();
      System.out.println("Sie nehmen "+symbol(user)+".");

      int pc = (int)(Math.random()*3+1);
      System.out.println("Der PC nimmt "+symbol(pc)+".");

      scanner.close();
```

Rekursion

- Methode ruft sich selbst auf
- Iteration
  - Rekursion kann als Alternative zur Schleife genutzt werden
- Wechselseitige Rekursion
  - Mehrere Methoden rufen sich gegenseitig auf

Rekursion


DREI WÜNSCHE ? OK...
1. Ne Million Euro
2. Ewige Gesundheit
3. Nochmal drei Wünsche
Jojo'02

GGT mit Schleife

```
public static int ggt(int a, int b) {
    int rest;
    do {
        rest = a % b;
        a = b;
        b = rest;
    } while (rest != 0);
    return a;
}
```

GGT mit Rekursion

```
public static int ggt(int a, int b) {
    if (b == 0) {
        return a;
    }
    else {
        return ggt(b,a%b);
    }
}
```

Konventionen

Konventionen

- Java Code Conventions
  - http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

- Richtlinien zur Gestaltung von Programmcode
- Werden nicht vom Compiler überprüft

Wozu Konventionen?

- Die Verwaltung von Software nimmt auf lange Sicht die meiste Zeit der Softwareentwicklung in Anspruch
- Software wird in der Regel
  - Von mehreren Entwicklern in Kooperation erstellt
  - Nicht vom Erstautor während ihres gesamten Bestehens verwaltet
- Code-Konventionen verbessern die Lesbarkeit
  - Unbekannter Code kann von geübten Programmierern leichter verstanden werden

Konventionen

- Eine Anweisung pro Zeile
- Keine Umlaute, Sonder- oder Leerzeichen in Namen
- Einrücken
  - Blöcke (das was innerhalb von geschweiften Klammern { } steht) einen Tab (oder 4 Leerzeichen) tiefer einrücken
- Zeilenlänge
  - Maximal 80 Zeichen pro Zeile
  - Umbruch
    - Nach einem Komma
    - Vor einem Operator
    - Eingerückt entsprechend der aktuellen Zeile in Anweisungen
    - 2 Tabs tiefer eingerückt im Kopf von Kontrollstrukturen

Namenskonventionen

- Variablen
  - Kurz, aber aussagekräftig
  - Beginnt mit Kleinbuchstaben (gut: `mySize`, schlecht: `MySize`)
  - Weitere Wörter beginnen mit Großbuchstaben
- Konstanten
  - Ausschließlich Großbuchstaben (`MY_SIZE`)
  - Mehrere Wörter durch Unterstrich getrennt

Namenskonventionen

- Methoden
  - Sollten Verben sein
  - Beginnt mit Kleinbuchstaben (gut: `calcSize`, schlecht: `CalcSize`)
  - Weitere Wörter beginnen mit Großbuchstaben
- Klassen
  - Sollten Substantive sein
  - Beginnt mit Großbuchstaben (`Image`)
  - Weitere Wörter beginnen mit Großbuchstaben
- Paket
  - Möglichst nur ein Wort komplett klein geschrieben

Kommentare

- Einzeiliger Kommentar
  - Beginnt mit `//`.
  - Beispiel: `// Dies ist ein Kommentar`
- Mehrzeiliger Kommentar
  - Beginnt mit `/*` und endet mit `*/`.
  - Beispiel: `/* Dies ist ein Kommentar */`
- JavaDoc
  - Beginnt mit `/**` und endet mit `*/`.
  - Jede Zeile innerhalb des Kommentars beginnt mit `*`.
  - Steht vor einer Methode
  - Daraus kann automatisiert eine Dokumentation generiert werden

Zusammenfassung

- Vorhandene Java Methoden (Genaueres in der API)
  - `System.out.println`
  - `Math.sqrt`
- Eigene Methoden schreiben

```
public class Klassenname {
    public static Rückgabe methodenname (Parameter) {
        // Hier stehen die Anweisungen
    }
}
```

- Methoden aufrufen
  - `System.out.println("Hello World");`
  - `double x = Math.sqrt(4);`
  - **Allgemein:** `Klassenname.methodenname(Parameter);`

Hausaufgaben

2.7 Methoden einer Klasse