Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

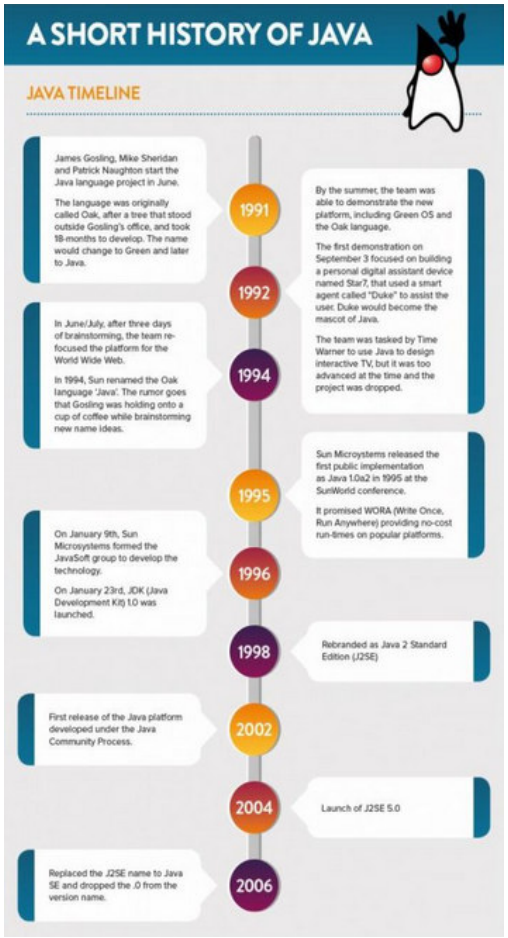
Grundlagen der Programmierung & Algorithmen und Datenstrukturen

Einführung in die strukturierte Programmierung mit Java – Teil 3

Die Inhalte der Vorlesung wurden primär auf Basis der angegebenen Literatur erstellt. Darüber hinaus orientiert sich diese an der Vorlesung von Prof. Dr.-Ing. Faustmann (ebenfalls HWR Berlin).

Inhalt des Parts

- Grundlegende Aspekte
- Integrierte Entwicklungsumgebung Eclipse
- Programmieren mit Java
  - Grundregeln der Programmiersprache
  - Anweisungen und Funktionen
  - Datentypen und Operatoren
  - Ein- und Ausgaben
- Einsatz von Kontrollstrukturen



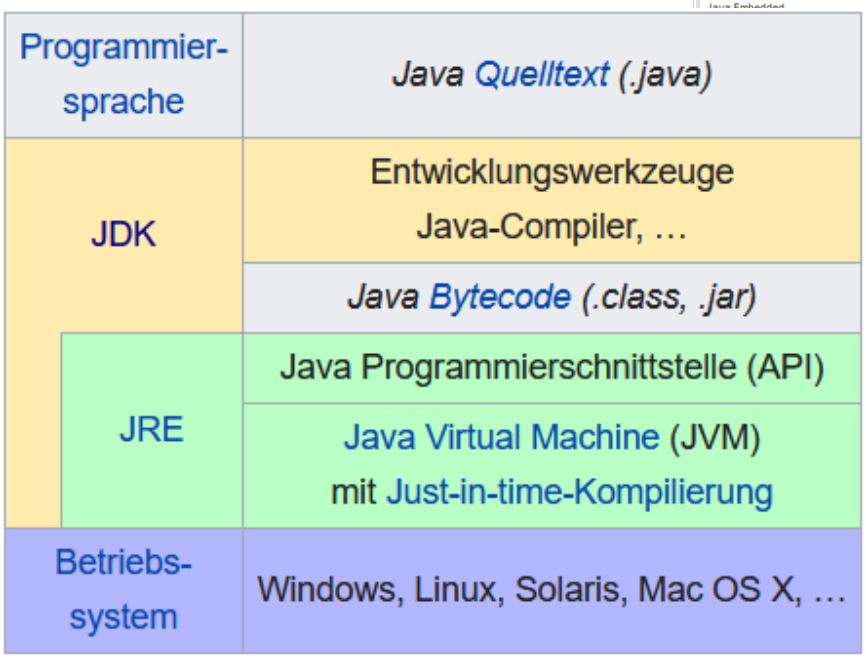
Quelle der Abb.: https://kettlemag.co.uk/the-history-of-the-java-programming-language/

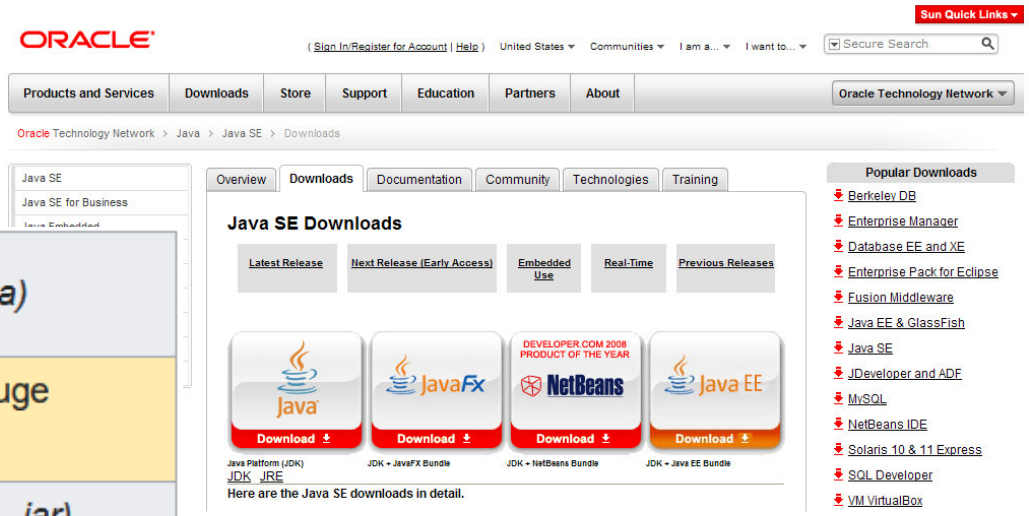
Grundlegende Aspekte

Grundlegende Aspekte

- Java-Softwareentwicklungskit – SDK oder kurz JDK
  - https://www.oracle.com/java/technologies/java-se-glance.html
  - http://www.java.com/de/
- Java ist auf unterschiedlichsten Plattformen verfügbar
  - Windows Betriebssystemfamilie – Fokus in der Lehre
  - UNIX/Linux-Versionen
  - MacOS-Version
- Strategische Positionierung
  - Portierbare Anwendungen „Write once run anywhere“
  - SUN plazierte Java als Middleware und Programmiersprache

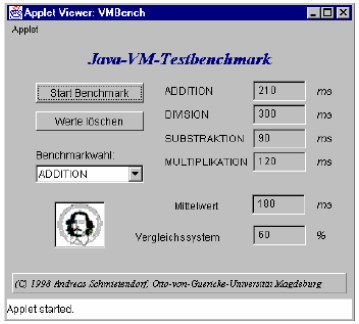
Oracle verantwortet Java – ab 2010





Quelle linke Abb.: https://de.wikipedia.org/wiki/Java-Technologie
Quelle rechte Abb.: http://www.oracle.com/technetwork/java/javase/overview/index.html

Werkzeuge des SDK - Beispiele



- javac - Java Compiler → Quellcode in Bytecode
- java - Java-Interpreter lädt Bytecode und führt ihn aus
- javadoc – HTML-Doku erzeugen (spezielle Kommentare)
- javah - Verknüpfung von Java-Klassen mit ext. Code (z.B. mit C++)
- javap - Java-Disassembler Bytecode in Quelltext zurückübersetzen
- jdb - Java Debugger (Test und Fehlersuche)
- jar - Java-Archive (Auslieferung von Komponenten und Programmen)
- ~~appletviewer - lädt HTML-Dokumente und zeigt Applets an~~

  → `Siehe z.B. unter c:\programme\java\jdkxxx\bin`

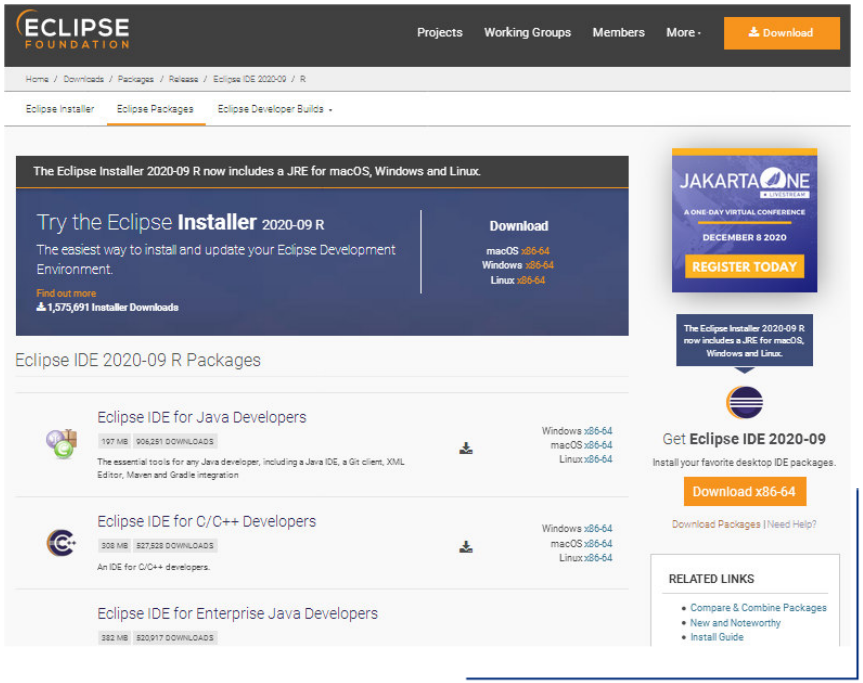
Entwicklungsumgebung Eclipse

Eclipse

- Angebot einer generischen Applikationsplattform (Open Source)
  - Basis war die IBM Entwicklungsumgebung Visual Age
  - Entwicklungsplattform für beliebige Anwendungen
  - Erweiterbarkeit auf der Basis so genannter „Plug Ins“ (via PDT)
  - Team Support mittels GIT, Build Support mittels Ant/Maven, …
  - Erstellung und Präsentation von Hilfetexten
- Java Development Tools (kurz JDT)
  - Erstellung von Programmen
  - Debuggen von Quellcode

Übung 1

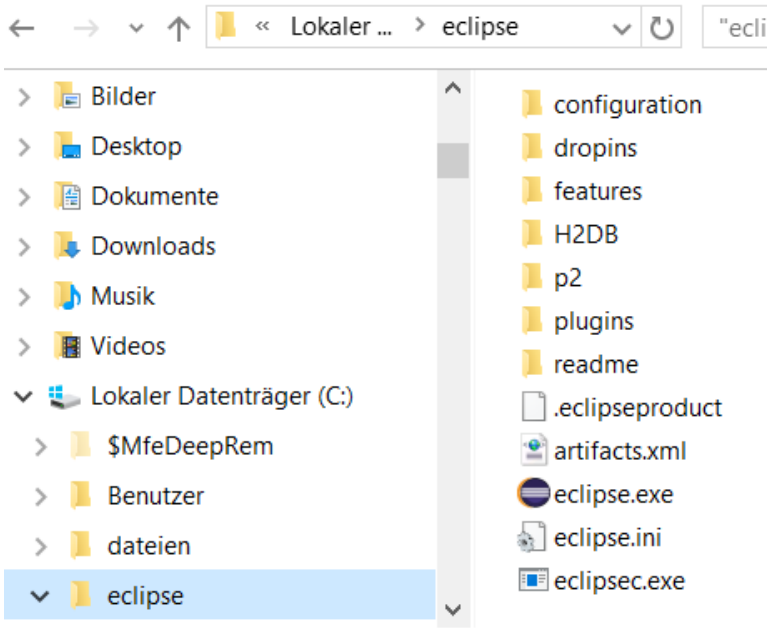
- Installieren von Eclipse (Windows, Linux oder MacOS):
  - IDE für JavaSE (Standard Edition)
  - Ggf. Enterprise Edition (optional)
- Benötigte JVM:
  - JRE – Java Runtime Edition
  - SDK – Software Development Kit
- Hilfe – siehe Webseiten



Quelle der Abb.:
https://www.eclipse.org/downloads/

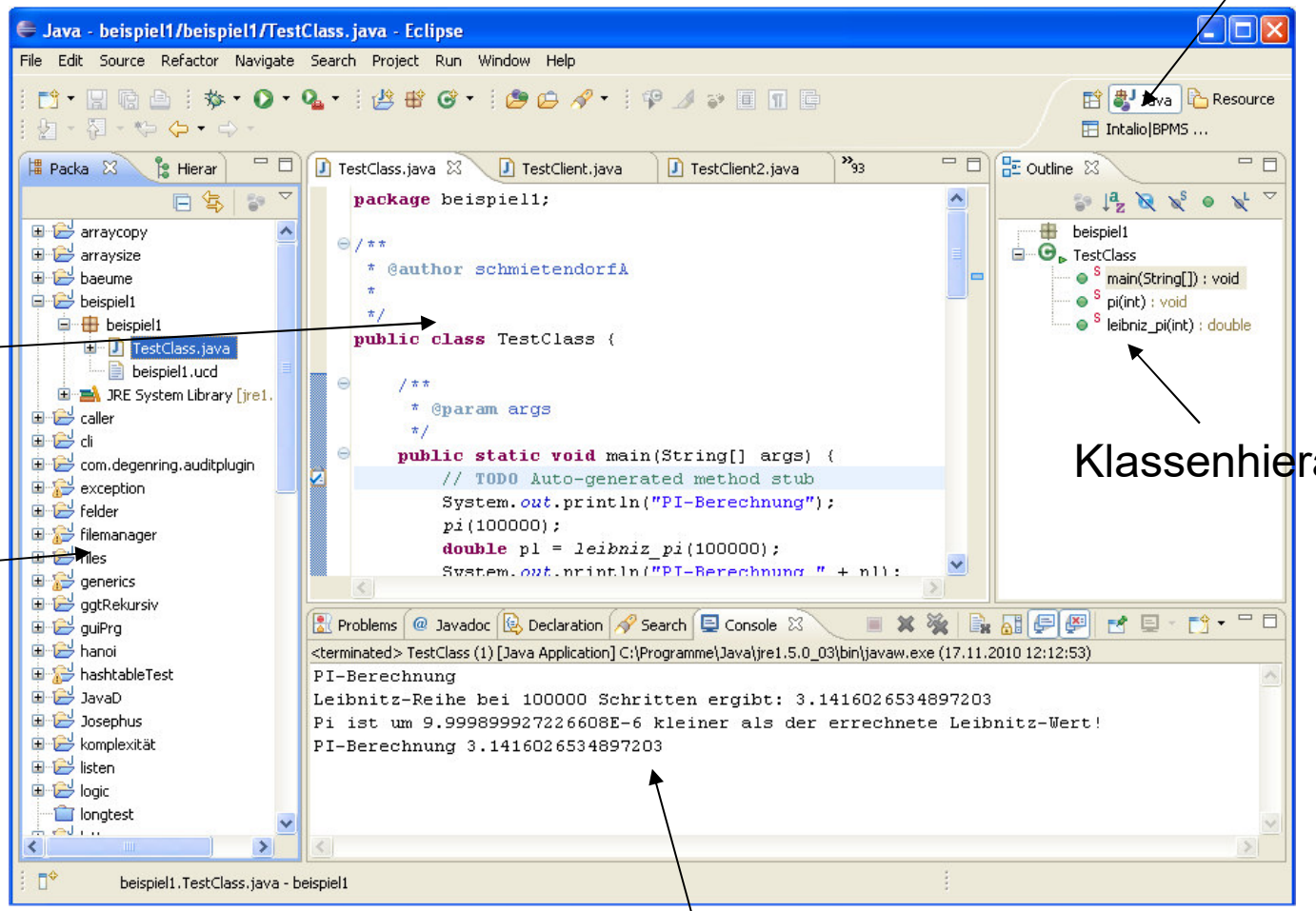
Eclipse

- Eclipse unter Windows
  - Support verschiedener JRE
  - u.a. Eclipse Che (Cloud-IDE)
  - Aktuell bis Java 17
- Konfiguration beeinflusst
  - Keine Registrydaten
  - Keine config-Dateien
  - Keine Systemeinträge



- Verwaltung von Projektinformationen in Workspaces
  - Speicherort für Code- und Hilfedateien
  - Speicherort für Einstellungen des Projekts oder der Perspektiven
  - Speicherung für Codeänderungen (Änderungen ggf. widerrufen)

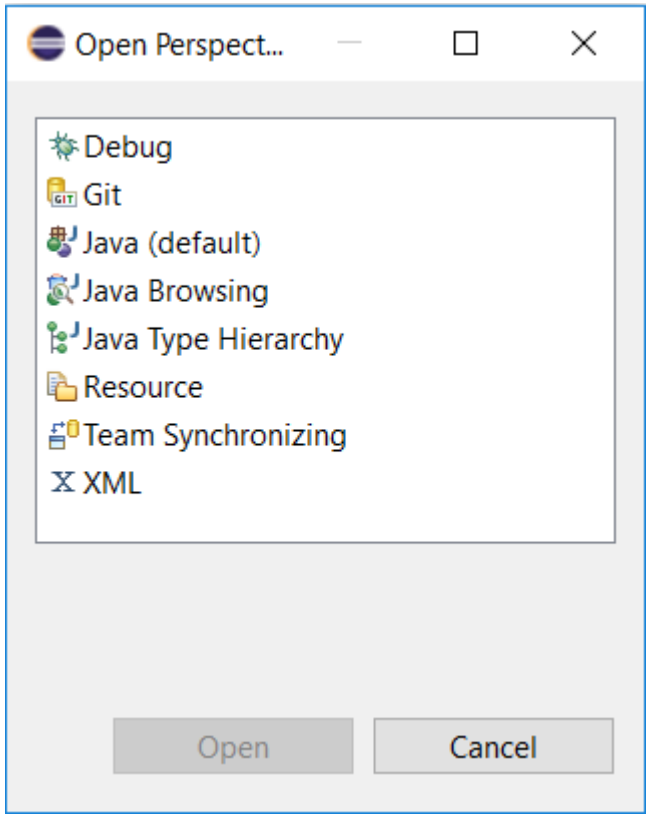
Hauptfenster von Eclipse



Eclipse

Perspektiven (Sichten auf bearbeitete Problemstellungen):

- Java-Perspektive
  - Package Explorer
  - Editor
  - Outline
  - Problems
  - Console
- Debug-Perspektive
- Git-Repository
- …



Views der Java Perspektive

- Editor

  Hier können Quelltexte eingegeben werden. Bereits hier werden syntaktische und teilweise sogar logische Fehler angezeigt (z.B. package wird nicht verwendet).

- Package Explorer

  Es wird die Zuordnung der Pakete zu den geöffneten Projekten dargestellt. Hier wird deutlich, welche Projekte gerade geöffnet sind und welche Programme ausgeführt werden können.

- Outliner

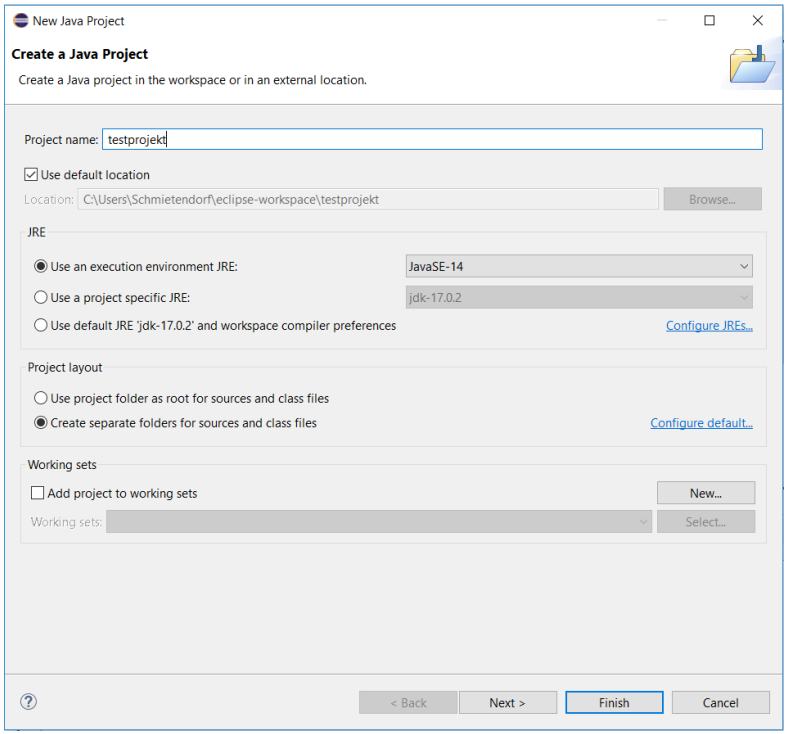
  Im Outliner wird die innere Programmstruktur dargestellt. Dazu gehören Klassen, sowie ihre Methoden und Attribute.

- Console

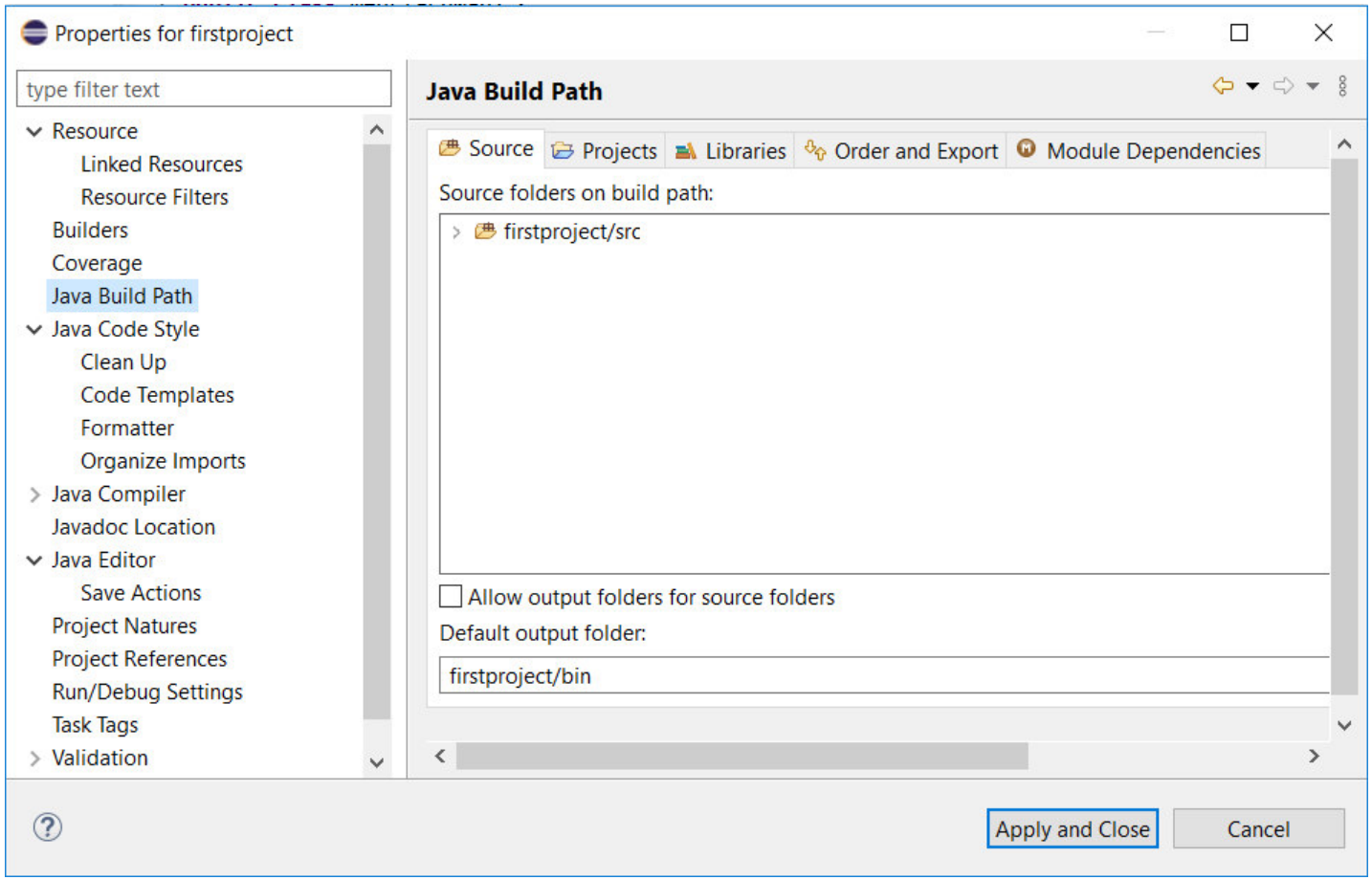
  Im Konsolenbereich erfolgt die Ausgabe von Daten, falls mit System.out.println() gearbeitet wird. Ebenso kann hier eine Eingabe angefragt werden.

Anlegen eines neuen Projektes

- Programme sind in einem eigenen Projekt anzulegen.
  - File → New → Java Project
- Project name
  - Anlegen eines neuen Projektes
  - Berücksichtigung exist. Quellen
  - Notw. Libraries (jar‘s)
- JDK-Kompatibilität
  - maximal JDK 17
  - oder weniger nutzen
- Project layout
  - Verzeichnisstrukturen



Eclipse Projekteinstellungen

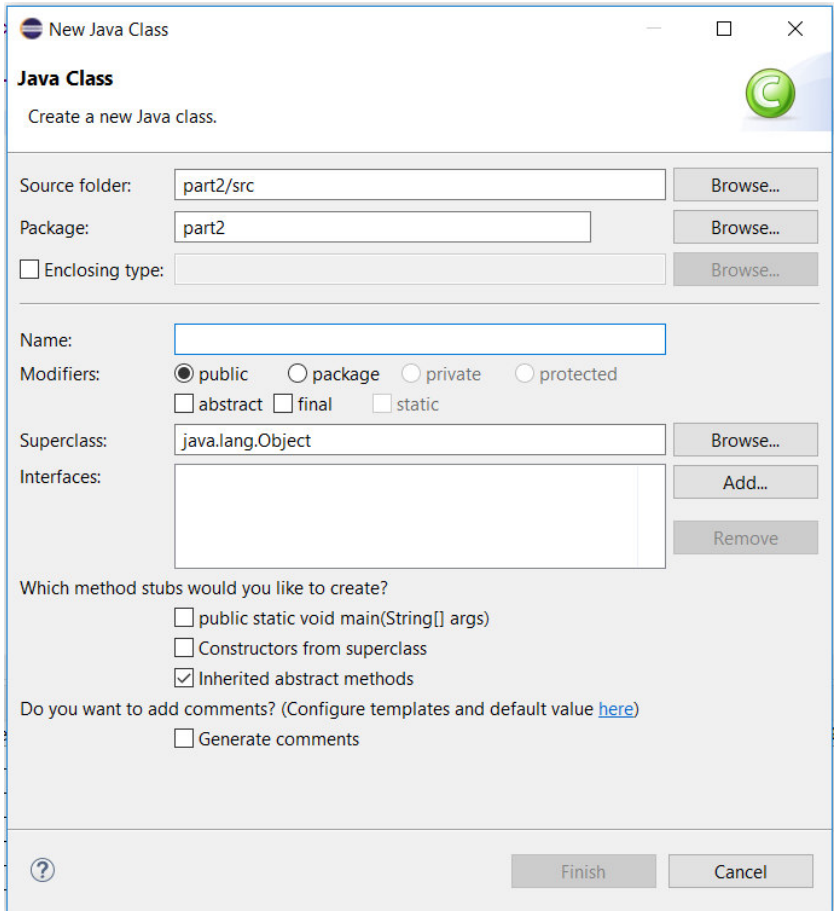


Einstellungen zum Editor (z.B. richtige Darstellung von Sonderzeichen)

Anlegen von Java-Klassen

Anlegen von Klassen

- Damit ein Programm in einem Projekt angelegt werden kann, muss mindestens eine Klasse kreiert werden. Wählen Sie dazu File->New->Class.
- Wählen Sie außerdem, dass die main-Methode automatisch generiert wird. Sie sparen sich damit Tipparbeit!
- Anschließend erscheint die neue Klasse im Editorfenster und Sie können beginnen, Ihr Programm zu erstellen.



Projekte ausführen

- Geöffnete Projekte können von Ihnen gestartet werden.
- Wird ein Projekt zum ersten Mal ausgeführt, so ist Run As → Java Application unter dem Symbolknopf (Wiedergabe) auszuwählen.
- Für einen erneuten Start eines Projekts gibt es zwei Möglichkeiten:
  - Wurde es als letztes Projekt ausgeführt, braucht nur der Symbolknopf gedrückt zu werden.
  - Im anderen Fall kann es aus der Liste der zu startenden Projekte ausgewählt werden.
- Achtung: Ein bereits einmal ausgeführtes Projekt erscheint auch in dieser Liste, wenn es gar nicht geladen ist!
- Projekte können geschlossen und wieder geöffnet werden.

Einstieg in die Java-Programmierung

Programmieren mit Java

Grundregeln der Sprache Java:

- Java ist „case sensitiv“ (d.h. groß- und kleinschreibungsgebunden)
- Textsteuerzeichen werden im ignoriert (Leerzeichen, Tabulatoren, …)
- Geschweifte Klammern stellen Gruppierungen dar (zeigen Beginn und Ende eines Programmabschnitts an)
- Alle Java-Anweisungen enden mit dem Semikolon (;)
- Verwendung von Kommentaren in Java
  - // einzeilige Kommentare
  - /* …. */ Kommentare über mehrere Zeilen
  - /** … */ spezielle Kommentare – genutzt von „javadoc“

Programmieren mit Java

Grundregeln der Sprache Java:

- Jedes Programm sollte mit mehreren Kommentarzeilen beginnen
  - Programmname
  - Name des Autors bzw. der Autoren
  - Informationen zum Urheberrecht
  - Datum der Erstellung des Programms
  - Beschreibung der Aufgabe und grundlegenden Funktionen
  - Geschichte zu Programmänderungen und –aktualisierungen
  - Verwendetet Werkzeuge
  - Informationen zur Soft- und Hardwareumgebung

Programmieren mit Java

```
// Ein einführendes Java-Programm

/* Einführung in die Programmierung */

public class HalloStudenten {

/**
 * @param args
 */
   public static void main(String[] args) {
        // TODO Auto-generated method stub

        System.out.println("Studenten der HWR Berlin");

   }

}
```

Erläuterungen I

- Die ersten beiden Zeilen zeigen die Verwendung unterschiedlicher Java-Kommentare
- public steht für ein Zugriffsrecht (aus anderen Klassen und Paketen), class bezeichnet eine Klassendefinition.
- System.out.println() ist eine Bibliotheksfunktion, die eine Ausgabe im Konsolenfenster produziert.
- Dieser Quellcode muss in einer Datei mit dem Namen „HalloStudenten.java“ gespeichert werden (Eclipse übernimmt durch Angabe des Klassennamens diese Aufgabe).

Erläuterungen II

- Durch Aufruf des Java-Compilers javac wird der Bytecode in der Datei „HalloStundenten.class“ generiert. Mit dem Java-Interpreter java kann der Bytecode dann ausgeführt werden.
- Geschweifte Klammern trennen die Teile eines Programms (Anweisungsblöcke). Sie treten immer paarweise auf.
- Jedes Programm muss eine main-Funktion der Form public static void main(String[ ] args) haben. Diese wird als Einstiegspunkt (driving class) in das Programm zuerst ausgeführt.

Übung 2

- Starten Sie die Entwicklungsumgebung Eclipse.
- Erstellen Sie ein neues Projekt mit dem Namen HalloStudenten.
- Legen Sie eine neue Klasse an
  - Verwenden Sie eine Methode main()
  - Nutzen Sie die verschiedenen Arten von Java-Kommentaren
  - Geben Sie ihren Namen auf der Konsole aus
- Starten Sie das Projekt.
- Korrigieren Sie eventuell aufgetretene Fehler.

Anweisungen und Funktionen

Anweisungen und Funktionen (Aufrufe):

- Anweisungen sind durch Semikolon getrennte Programmteile:

```
System.out.print("Zahl eingeben: ");
double nenner=1;
```

- Neben der main()-Funktion können weitere Funktionen definiert werden. Sie beisitzen die folgende Form:

```
static Rückgabetyp Funktionsname ( Parameterliste) {
    Funktionsblock
}
```

- Beispiel:

```
static void ausgabe(){
        System.out.println("Hallo");
}
```

leere Parameterliste

kein Rückgabewert

Übung 3

- Erstellen Sie neben der main()-Methode drei weitere Methoden denen Sie die Namen funktion1(), funktion2() und funktion3() geben. Im Rumpf der Methode soll lediglich eine Ausgabeanweisung in folgender Weise erfolgen:

```
System.out.print(„Funktion x ");
```

- Alle Methoden erhalten eine leere Parameterliste und erzeugen bei Beendigung kein Rückgabewert.
- Rufen Sie die Methoden innerhalb der main()-Methode nacheinander auf.
- Ändern Sie die Methoden so ab, dass beim Aufruf ein Integer-Wert als Parameter übergeben werden kann. (optional)

Bezeichner und Datentypen

Elementare Datentypen: Ganzzahlen

- Zahlen ohne gebrochenen Anteil. Auch negative Werte sind zulässig. Es existieren vier Ganzzahltypen:
  - `byte` 1 Byte -2^7 bis 2^7-1
  - `short` 2 Byte -2^{15} bis $2^{15}-1$
  - `int` 4 Byte -2^{31} bis $2^{31}-1$
  - `long` 8 Byte -2^{63} bis $2^{63}-1$
- Die angegebenen Datentypen hängen nicht von der jeweiligen Plattform ab (im Gegensatz zu C/C++, wo z.B. int 2 oder 4 Byte haben kann).

Bezeichner und Datentypen

Elementare Datentypen: Fließkommazahlen

- Zahlen mit gebrochenem Anteil. Auch negative Werte sind zulässig. Es existieren zwei Gleitkommazahlentypen:
  - `float` 4 Byte 7 bis 8 Ziffern (einfache Genauigkeit)
  - `double` 8 Byte 16 bis 17 Ziffern (doppelte Genauigkeit)
- Achtung: Beim Vergleich von Gleitkommazahlen stellen sich Rundungsfehler einstellen. So ist z.B. das Ergebnis des Ausdrucks 2.0-1.1 nicht 0.9, sondern 0.8999999999999999! Dies liegt an der internen Darstellung der Gleitkommazahlen im binären Zahlsystem (wie z.B. im Dezimalsystem der Bruch 1/3 nicht genau ausgedrückt werden kann, so kann im Binärsystem auch nicht 1/10 exakt abgebildet werden).

Bezeichner und Datentypen

Elementare Datentypen: Zeichen und Wahrheitswerte

- Bei char handelt es sich um einen Datentyp, der einzelne Zeichen aufzunehmen kann. Entsprechende Zeichenkonstanten sind dafür in einfache Anführungszeichen zu setzen: z.B. ‘c‘
  - `char` 2 Byte 65536 (Zeichenvorrat von Unicode)

  Bereichstellung von Escape-Sequenzen wie z.B. \b oder \n oder \t
- Boolsche Werte:
  - `boolean` -- true und false (Standardwert → false)

Bezeichner und Datentypen

Deklarieren von Variablen I

- Variablen speichern Daten im Hauptspeicher eines Programms und können gelesen und verändert werden.
- Die Variablendeklaration informiert den Compiler über Name und Typ der Variable und reserviert entsprechenden Speicherplatz.
- Beispiele:

```
int number1;
int number1, number2, number3; // mehrere Variablen des
   selben Tys
char zeichen = ‚s‘ // mit Initialisierung
```

Bezeichner und Datentypen

Deklarieren von Variablen II

- Bezeichner werden in Java zum Benennen von Variablen, Konstanten, Klassen und Methoden genutzt.
- Der Variablenname muss einen gültigen Java-Bezeichner darstellen.
  - Ein Bezeichner kann die Buchstaben ‘A‘-‘Z‘ und ‘a‘-‘z‘, die Zahlen ‘0‘-‘9‘, den
  - Underscore ‘_‘ und das Dollarzeichen ‘$‘ enthalten.
  - Der Bezeichner muss mit einem Buchstaben oder dem Underscore beginnen.
  - Der Bezeichner darf keinem Schlüsselwort aus Java entsprechen.
  - Bezeichner sind case-sensitiv.
- Ungültige Bezeichner: `1_value` oder `User name` oder `%%123`

Java Operatoren

Klassifikation der grundlegenden Java-Operatoren

- Einfache Zuweisungsoperatoren (keine algebraische Interpretierung!!)
- Arithmetische Operatoren
- Verknüpfungsoperatoren
- Inkrementierungs- und Dekrementierungsoperatoren
- Logische Operatoren
- Bit Operator

Entsprechend der Anzahl der Operanden unterscheidet man in Java unäre, binäre und ternäre Operatoren (auch Bedingungsoperator)

Java Operatoren

- Zuweisungsoperator: =
- Arithmetische
  - Addition: +
  - Subtraktion: -
  - Division: /
  - Multiplikation: *
  - Rest: (Restwert mod) %
- lvalues und rvalues
  - left values – Ausdruck der links vom = benutzt werden darf
  - right values - Ausdruck der rechts vom = benutzt werden darf

Java Operatoren

- Relationale Operatoren in Java
  - kleiner als: <
  - größer als: >
  - kleiner-gleich als <=
  - größer-gleich als >=
  - gleich ==
  - ungleich !=
- Logische Operatoren
  - Logisches AND &&
  - Logisches OR ||
  - Logisches NOT !
- Bit Operationen (AND, OR, XOR, NOT, Bitverschiebungen, …)

Java Operatoren

- Kombinationen:
  - x+=12; entspricht x = x + 12;
  - y*=23; entspricht y = y * 23;
- Inkrement bzw. Dekrement:
  - x++; entspricht x = x + 1;
  - x--; entspricht x = x – 1;
- Verwendung als
  - *Präfix-Form* (++i): Ausführung vor Verwendung
  - Postfix-Form (i++): Ausführung nach Verwendung

Übung 4

- Schauen Sie sich das folgende Programmfragment an und überlegen Sie welche Zahl jeweils ausgegeben wird.
- Prüfen Sie ihr Ergebnis mit Hilfe eines Java-Programms?
- Diskutieren Sie ggf. vorhandene Abweichungen

```
int a = 15;
int b = 13;
int c = a%b;
int d = c%b;
System.out.println("mod "+c);
System.out.println("mod "+d);
a = ++a + a++;
System.out.println(a);
a = a + a;
System.out.println(a);
b++;
System.out.println(b);
--b;
System.out.println(b);
b *= b;
System.out.println(b);
a += b;
System.out.println(a);
```

Übung 5

- Geben Sie drei Möglichkeiten an, den Wert 1 zur Variablen `count` zu addieren.
- Welche Werte nimmt die boolsche Variable `result` an, wenn `x = 6` und `y = 2` gesetzt wird?
  - `result = x > y; ?`
  - `result = x < y; ?`
  - `result = x == 0; ?`
  - `result = x != 0; ?`
  - `result = x <= 7; ?`
- Welche Werte nimmt die boolsche Variable `result` an, wenn x = 4, y = 2 und z = 0 gesetzt wird?
  - `result = (x > y && z == 0) ?`
  - `result = (x == 0 || z == 0) ?`
  - `result = !(y == 2 || z < x) ?`

Ein- und Ausgaben

- Die Ausgabe auf der Konsole wurde schon beispielhaft aufgezeigt. Mit `System.out.print()` kann eine Ausgabe auch ohne Zeilenumbruch erfolgen.
- Das Einlesen von Daten ist komplexer, es gibt z.B. die Möglichkeit mittels der swing-Klassenbibliothek Daten einzulesen:

  Ein eigenes *Eingabefenster* generieren mit dem folgenden Programmstück:

```
String eingabe = JOptionPane.showInputDialog("Zahl eingeben!");
int zahl = Integer.parseInt(eingabe);
```

- In diesem Fall muss noch das Paket `javax.swing.*` importiert werden, da nur dort die Klasse JOptionPane definiert ist:

```
import javax.swing.*;
```

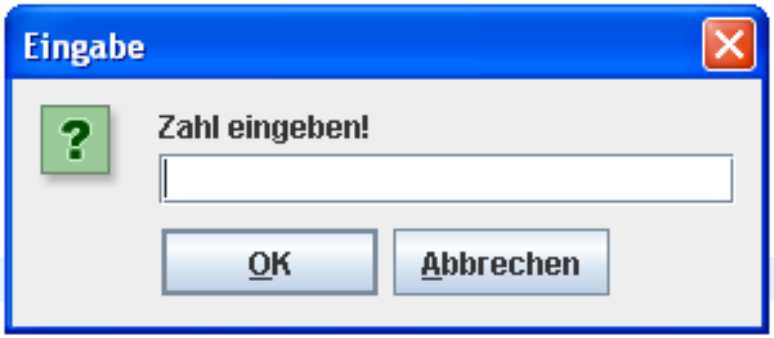
Ein- und Ausgaben

```
public class CharSearch {

    /**
     * @param args
     */
    public static void main(String[] args) {

    // Eingabe via Swing-Dialog
    public static int getEingabe(){
        String eingabe = JOptionPane.showInputDialog("Zahl eingeben!");
        int zahl = Integer.parseInt(eingabe);
        System.out.println("Eingabe war "+zahl);
        return zahl;
    }

}
```



Ein- und Ausgaben

- Von der *Konsole* direkt einlesen:

```
System.out.print("Zahl eingeben: ");
BufferedReader din = new BufferedReader( new
InputStreamReader(System.in));
int zahl = Integer.parseInt(din.readLine());
```

- Hier muss das zum einen das Paket java.io importiert werden und zum anderen ein potenzieller Fehler in main angezeigt werden:

```
import java.io.*; // Alles aus java.io importieren
static void main(String[] args) throws IOException {…}
```

Übung 6

- Testen Sie beide der angegebenen Möglichkeiten zur Eingabe von Werten aus. Nutzen Sie dafür jeweils eine eigene Methode entsprechend der folgenden Signaturen.
  - Eingabe innerhalb eine kleinen Windows!
    - public static void getValueSwing() {…}
  - Eingabe innerhalb der Konsole
    - public static void getValueStream() {…}
- Geben Sie die eingegebenen Werte direkt innerhalb der beiden Methoden auf der Konsole aus.
- Geben Sie die Werte innerhalb der main()-Methode aus, verändern Sie dafür die Signaturen beider Methoden. (optionale Aufgabe)

Kontrollstrukturen

- Alternative:

```
if(bedingung) {
        anweisung1;
}else{
        anweisung2;
}
```

- Liefert die Bedingung den Wahrheitswert `true`, dann wird `anweisung1` ausgeführt. Im anderen Fall wird der hinter dem Schlüsselwort `else` stehende Anweisungsblock (hier `anweisung2`) ausgeführt.
- Der else-Block ist dabei optional.

Kontrollstrukturen

- Mehrfachauswahl:

```
switch (ausdruck) {
   case constant: anweisung;
   ...
   default: ...
}
```

- Zunächst wird `ausdruck` ausgewertet, der vom Typ `byte`, `short`, `char` oder `int` sein muss. Dann wird die Sprungmarke `constant` angesprungen, die dem ermittelten Wert entspricht.
- **Achtung**: Wurde eine Sprungmarke angesprungen, so werden auch alle darunter liegenden Anweisungen abgearbeitet, unabhängig davon, ob sie hinter anderen Sprungmarken stehen. Um das zu verhindern, sollte jeder Fall mit einem break abgeschlossen werden.

Kontrollstrukturen

Beispiel einer implementierten Mehrfachauswahl:

```
public static void switchTest() throws IOException{

    System.out.print("Zahl eingeben: ");
    BufferedReader din = new BufferedReader(new InputStreamReader(System.in));
    int switchInput = Integer.parseInt(din.readLine());
    switch (switchInput){
        case 1: System.out.println("Sie haben 1 gewaehlt");
        break;
        case 2: System.out.println("Sie haben 2 gewaehlt");
        break;
        case 3: System.out.println("Sie haben 3 gewaehlt");
        break;
        default: System.out.println("Sie haben etwas anderes gewaehlt");
        break;
    }
}
```

Übung 7

- Schreiben Sie eine Funktion, der eine Jahreszahl übergeben wird und die einen wahren Wert (`true`) zurückliefert, wenn es sich um ein Schaltjahr handelt. Falls der übergebene Wert kein Schaltjahr ist, soll ein falscher Wert (`false`) zurückgeliefert werden.

  Signatur der Funktion:

  ```
  public static boolean isSchaltjahr(int jahr) { …}
  ```

  Ein Jahr ist kein Schaltjahr, wenn die Jahreszahl nicht durch 4 teilbar ist. Ein Jahr ist ein Schaltjahr, wenn die Jahreszahl durch 4, aber nicht durch 100 teilbar ist. Ein Jahr ist ebenfalls ein Schaltjahr, wenn die Jahreszahl durch 4, durch 100 und durch 400 teilbar ist.

- Schreiben Sie dazu eine main-Funktion, die vom Benutzer eine Jahreszahl einliest, die Funktion `isSchaltjahr()` ausführt und dem Benutzer das Ergebnis mitteilt.

Kontrollstrukturen

Kontrollstrukturen kopf- und fußgesteuerte Schleife

- **1. Kopfgesteuerte Schleife:**

```
while(bedingung){
   anweisung;
}
```

  - Der Anweisungsblock (hier `anweisung`) wird solange ausgeführt wie `bedingung` wahr ist. Ist `bedingung` zu Beginn false, so wird er nicht ausgeführt.

- **2. Fussgesteuerte Schleife:**

```
do{
   anweisung;
} while(bedingung)
```

  - Der Anweisungsblock (hier `anweisung`) wird zunächst einmal ausgeführt. Anschließend wird er solange ausgeführt wie `bedingung` wahr ist.

Kontrollstrukturen

Kontrollstruktur Zählschleife

```
for(anweisung1; bedingung; anweisung2){
    anweisung3;
}
```

entspricht:

```
Anweisung1 // Initialisierung
while (bedingung){
    anweisung3;
    anweisung2 // Inkrement oder Dekrement
}
```

- anweisung1 dient zur Initialisierung, danach wird der Anweisungsblock (hier anweisung3) und daran anschließend anweisung2 solange ausgeführt, bis bedingung (d.h. true) gilt.

Kontrollstrukturen

- Beispiel: Die folgenden Zählschleifen geben aufsteigend und absteigend die ersten einhundert ganzen Zahlen auf der Konsole aus.

```
// Vorwärts und rückwärts laufende Schleifen
public static void getIter(){
    for(int i = 1; i < 101; i++){
        System.out.println(i);
    }
    for(int j = 100; j > 0; j--){
        System.out.println(j);
    }
System.out.println("jeweils 100 Ausgaben - fertig!");

}
```

- i bzw. j wird als Zählvariable bezeichnet und kann in der Schleifen deklariert werden.

Übung 8

- Programmieren Sie eine Zählschleife, die die ersten einhundert geraden Zahlen in absteigender Reihenfolge ausgibt!
- Geben Sie innerhalb der Zählschleife auch die zur geraden Zahl gehörige Quadratzahl aus.
- Geben Sie darüber hinaus die jeweilige 2-er Potenz aus, dabei handelt es sich um das Ergebnis von 2^x, mit x als Wert korrespondierenden geraden Zahl!

  Bem.: mit Hilfe der Methode `pow(double basis, double exponent)`, diese steht im Rahmen der Klasse `Math` zur Verfügung, lässt sich die Zweierpotenz berechnen.

Übung 9

Kreiszahlberechnung nach Leibniz:

- Der Mathematiker Leibniz hat herausgefunden, dass man die Kreiszahl Pi auf die folgende Art und Weise berechnen kann:

$$4\sum_{v=0}^{\infty}\frac{(-1)^v}{2v+1} = (1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\ldots)*4 = \pi$$

- Entwerfen (Struktogramm) und implementieren (Java) Sie ein Programm, welches nach diesem Verfahren Pi berechnet. Da sich das Ergebnis immer weiter der tatsächlichen Zahl Pi annähert, je weiter die Reihe berechnet wird, soll das Programm vor der Berechnung fragen, wie weit die Reihe gebildet werden soll, damit keine unendliche Schleife entsteht.
- Vergleichen Sie ob ihr Ergebnis größer oder kleiner dem Wert der korrespondierenden Java-Konstante (Math.PI) ist.