

Programmieren I

Die Programmiersprache Java

Institut für Automation und angewandte Informatik

```
final List<String> allResults = new ArrayList<String>();  
final Map<String, Integer> typeWordResultCount = new HashMap<String, Integer>();  
final Map<String, Integer> typePoints = new HashMap<String, Integer>();  
evaluation.put(type, typePoints);  
  
for (final Sheet sheet : this.sheets) {  
    final String sheetResult = sheet.getPlayerInput(type);  
    if (sheetResult.startsWith(start) && this.isValidWord(sheetResult, type)) {  
        validWordCountForType++;  
        allResults.add(sheetResult);  
    }  
}
```

RedMonk

Popularity Rank on Stack Overflow (by # of Tags)

Popularity Rank on GitHub (by # of Projects)

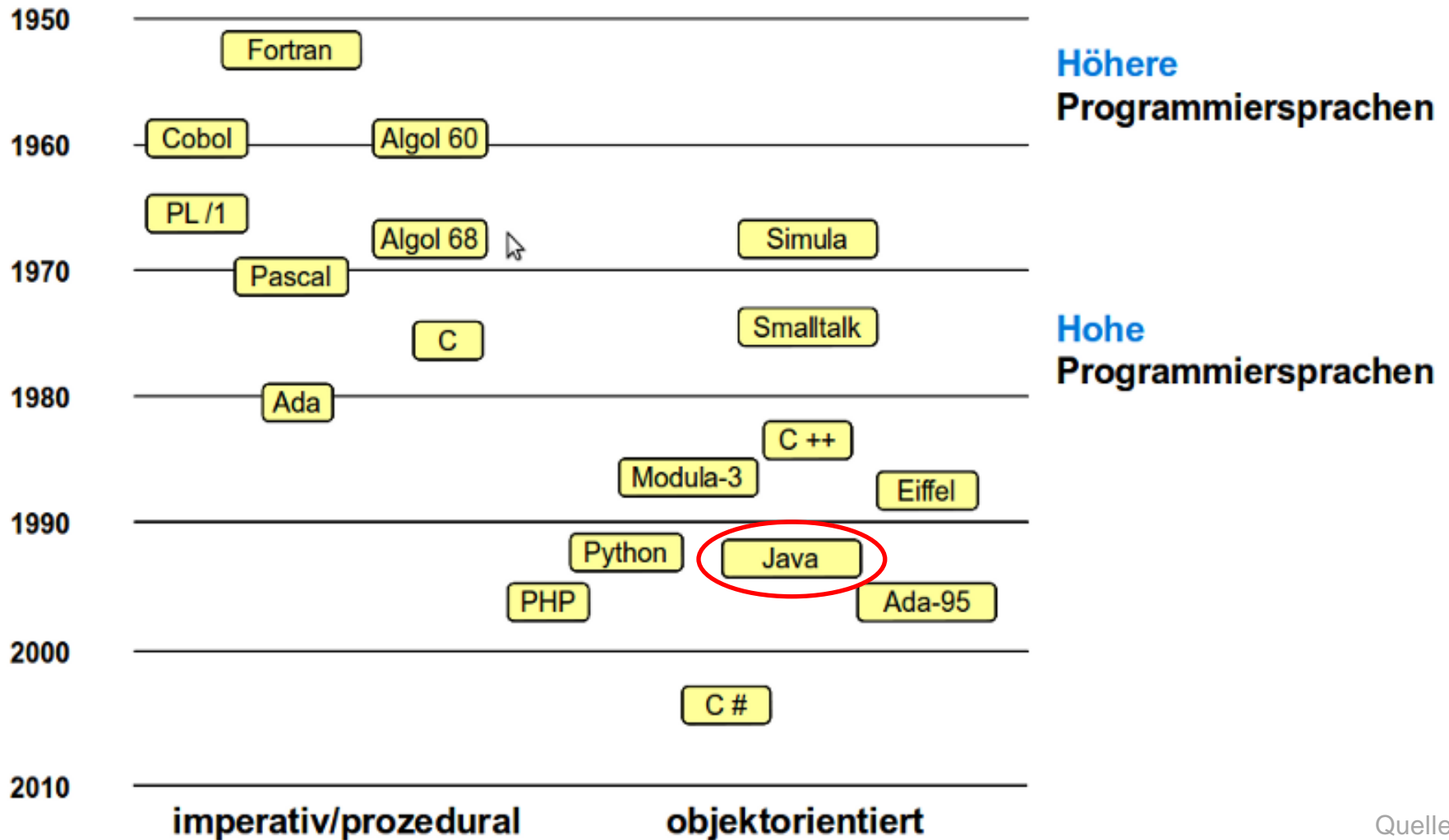
Scatter plot showing the correlation between Popularity Rank on Stack Overflow (Y-axis) and Popularity Rank on GitHub (X-axis). The diagonal line represents the identity line (y=x). The plot shows a strong positive correlation between the two metrics for most programming languages and tools.

Key programming languages and tools plotted include:

- ASP
- Visual Basic
- Matlab
- Assembly
- ColdFusion
- Arduino
- F#
- FORTRAN
- Dart
- Erlang
- Kotlin
- CoffeeScript
- Rust
- Scala
- Shell
- TypeScript
- Go
- Perl
- PowerShell
- Haskell
- Clojure
- Lua
- Julia
- OCaml
- Elixir
- Emacs
- Lisp
- Puppet
- D
- PLpgSQL
- Smarty
- Common Lisp
- Processing
- Racket
- Mathematica
- Verilog
- VHDL
- GLSL
- Cucumber
- Prolog
- SAS
- FreeMarker
- NSIS
- XQuery
- Coq
- Standard ML
- Web Ontology Language
- Liquid
- Objective-C++
- PostScript
- BitBake
- Modelica
- Vala
- Pascal
- Apex
- SaltStack
- Elim
- Haxe
- Crystal
- PureScript
- Chapel
- Logos
- SourcePawn
- SQLPL
- LiveScript
- MAXScript
- API Platform
- Butterchicken
- Eagle Roff
- Nim
- Gherkin
- GAP
- KiCad
- API Platform
- Butterchicken
- Eagle Roff
- SQLF
- HCL
- Nix
- VimL
- TeX
- JavaScript
- Java
- C#
- PHP
- C++
- Python
- CSS
- Ruby
- C
- Swift
- Objective-C
- R

2

Programmiersprachen, Entwicklung



Quelle:
<http://www.ilsb.tuwien.ac.at/~pahr/317.530/>

Eigenschaften von Java

- Java ist eine von der Firma Sun Microsystems (mittlerweile aufgegangen in Oracle) entwickelte objektorientierte Programmiersprache.
- Java is...



Heusch 2.2
Ratz 1.1

„...a simple, object-oriented, distributed, interpreted, robust, secure, architectural neutral, portable, high-performance, multithreaded, and dynamic language.“

(Sun Microsystems)

- Plattformunabhängigkeit – „Write Once, Run Anywhere“
- Java enthält standardmäßig eine große Anzahl von Bibliotheken, z.B. zur Grafikprogrammierung, zum Netzwerkzugriff oder zur Arbeit mit Dateien und Datenbanken.
- Java ist frei verfügbar. Außerdem stehen auch zahlreiche kostenfreie Entwicklungsumgebungen zur Verfügung.

Das Java Development Kit (JDK) (1)

■ Historie von Java

- 1991: Erste Anfänge bei Sun Microsystems, heute Oracle
- 1995: Vorstellung von Java auf der SUNWORLD '95
- 1997: Freigabe der Version Java 1.1
- 1998: Freigabe der Version Java 1.2 („Java 2“)
- Derzeit aktuelle Version Java 11

■ Bestandteile des Java Development-Kit (JDK)

- Java Runtime Environment (JRE)
- Tools
- Bibliotheken (Java-Klassen)
- Keine (!) grafische Entwicklungsumgebung

Das Java Development Kit (JDK) (2)

■ Tools

- Java-Compiler `javac`
- Java-Interpreter `java` zum Ausführen des vom Compiler erzeugten Bytecodes
- `appletviewer` zum Ausführen von Java-Applets
- Weitere Programme, beispielsweise zur Erstellung von Dokumentationen und zur Erzeugung so genannter JAR-Archive

■ Bibliotheken (Java-API - Application Programming Interface)

- Sammlung von Komponenten (Klassen), beispielsweise zur Entwicklung von grafischen Anwendungen, oder von Anwendungen mit Zugriff auf Dateien bzw. Ein- und Ausgabe über Tastatur und Bildschirm etc.
- Liste und Dokumentation unter <http://www.oracle.com>

Einsatzgebiete des JDK

Das JDK liegt für verschiedene Einsatzgebiete vor:

■ **Java Platform, Standard Edition (J2SE, SE)**

- Die Standard Edition wird zur Entwicklung von Programmen für Desktop-Computer eingesetzt.

■ **Java Platform, Enterprise Edition (J2EE, EE)**

- Mit der Enterprise Edition werden zusätzliche Bibliotheken für verteilte Anwendungen und Web-Services angeboten

■ **Java Platform, Micro Edition (J2ME, ME)**

- Diese Edition ist für Anwendungen gedacht, die in kleinen Geräten wie Telefonen, Handheldcomputern (PDA), Waschmaschinen etc. betrieben werden können, da diese Geräte wenig Speicher und geringe Rechenleistung besitzen.

Sprachkonzepte von Java

- Angelehnt an C/C++, aber stark vereinfacht:
 - Kein Präprozessor
 - Keine Pointer
 - Keine eigenen/überladenen Operatoren, Mehrfachvererbung, ...

- Bewährte Konzepte anderer Sprachen wurden integriert:
 - Exceptions: Fehlerbehandlung
 - Garbage Collection: Automatische Speicherfreigabe
 - Package-Konzept: Zusammenfassung von Klassen
 - Concurrency: Nebenläufigkeit

Java-Programme

■ Java-Applikationen (Anwendung, engl. Application)

- kann direkt auf der Betriebssystemebene gestartet werden
- besteht aus einer oder mehreren Klassen
- muss eine `main`-Methode enthalten
- wird mit Hilfe des Java-Interpreters gestartet und ausgeführt
- werden meistens einfach als „Java-Programme“ bezeichnet

■ Java-Applets

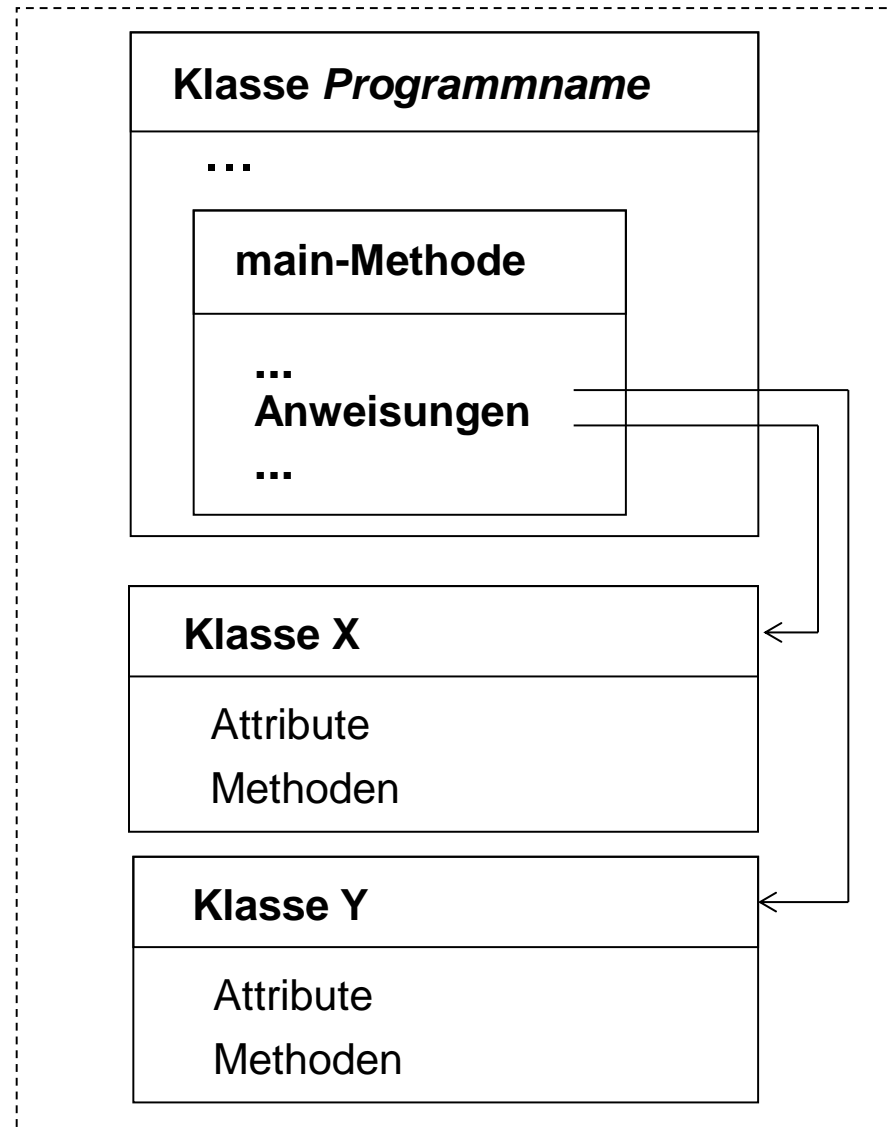
- können in HTML-Seiten eingebunden werden
(HTML = HyperText Markup Language)
- können durch das Programm `appletviewer` (Teil des JDK) oder einen WWW-Browser mit Java-Unterstützung ausgeführt werden

■ Weitere spezialisierte Anwendungen

- Servlet, Portlet, Web-Service, ...

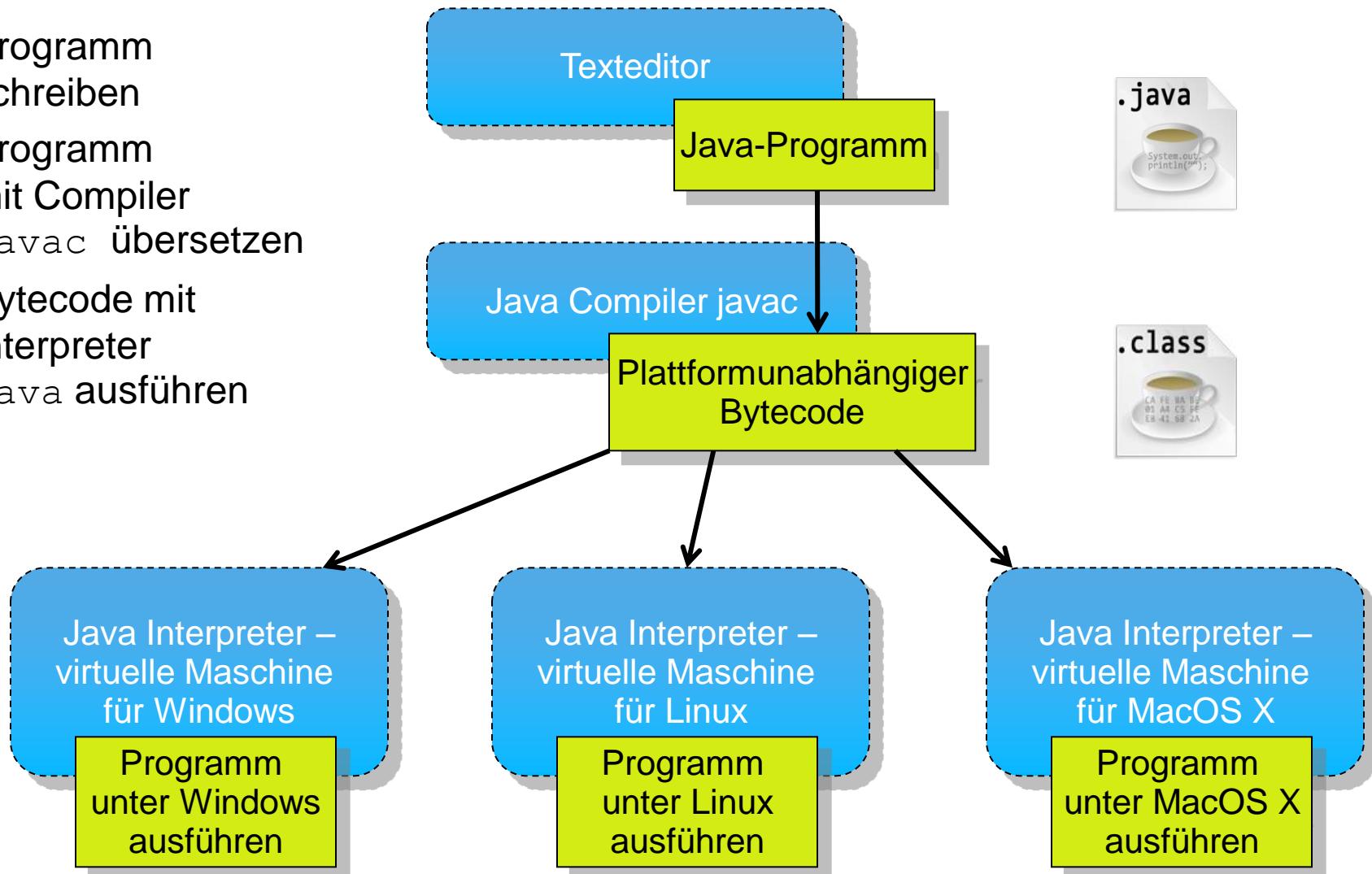
Aufbau einer Anwendung

- Java-Programme bestehen aus Klassen. Diese beinhalten alle Informationen, die zur Ausführung notwendig sind.
- Klassen umfassen Attribute (Daten) und Methoden (Funktionalität). Methoden beinhalten die Anweisungen, die ausgeführt werden sollen.
- Die einfachste Java-Anwendung besteht aus einer Klasse mit dem Namen der Anwendung. Diese Klasse enthält das Hauptprogramm, die `main`-Methode



Erstellung und Ausführung eines Java-Programms

- Programm schreiben
- Programm mit Compiler
`javac` übersetzen
- Bytecode mit Interpreter
`java` ausführen



Programmentwicklung in Java (1) – allg. Vorgehen

- Formulierung der Aufgabenstellung (Problem)
- Entwurf eines Lösungsalgorithmus
 - Formulierung auf abstrakter Ebene
 - Beachten von Strukturregeln
 - Korrektheit des Lösungsalgorithmus prüfen
 - Effizienzuntersuchungen
- Implementierung, d.h. Übertragung des Lösungsalgorithmus in eine Programmiersprache. Ergebnis ist ein Programm als Quellcode.
- Übersetzen (engl.: to compile) des Programms in eine maschinennahe Zwischensprache. Das geschieht mit Hilfe des Compilers (`javac`). Das Ergebnis ist Bytecode.
- Ausführen und Testen des Programms (`java`)

Programmentwicklung in Java (2) – Beispiel: Aufgabe

- Formulierung der Aufgabenstellung:
„Berechne den Quotienten zweier Zahlen a , b (d.h. a/b), falls $b \neq 0$. Sonst melde, dass b ein unzulässiger Wert ist.“
- Entwurf eines Lösungsalgorithmus

```
Lies zwei Zahlen a und b ein
WENN b ungleich 0
DANN
    quotient sei a / b
    schreibe quotient
SONST
    schreibe "b ist unzulässig."
```
- ... Strukturregeln, Korrektheit, Effizienz.

Programmentwicklung in Java (3) – Quellcode

Implementierung (Speichern als `Quotient.java`):

```
class Quotient {  
    public static void main(String args[]) {  
        float a, b, quotient;  
        java.util.Scanner scan = new java.util.Scanner(System.in);  
        System.out.println("Enter a: ");  
        a = scan.nextFloat();  
        System.out.println("Enter b: ");  
        b = scan.nextFloat();  
        if (b != 0) {  
            quotient = a / b;  
            System.out.println("Quotient: " + quotient);  
        } else {  
            System.out.println(b + " for b is invalid.");  
        }  
    }  
}
```

- Übersetzen des Programms (mit Hilfe des JDK):


```
> javac Quotient.java
```

(Erzeugt die Datei `Quotient.class`)



- Starten und Ausführen (Kommandozeile):

```
> java Quotient
Enter a: 3
Enter b: 6
Quotient 0.5
> java Quotient
Enter a: 3
Enter b: 0
0.0 for b is invalid.
```

 Groß-/Kleinschreibung beachten!

Programmentwicklung in Java (5) – Systemvorbereitungen

■ *Vorbereitungen*, beispielsweise unter Windows 7 wie folgt:

■ Pfadvariable ergänzen (mit Administrator-Rechten):

Start > Systemsteuerung > System >
Erweiterte Systemeinstellungen >
Umgebungsvariablen... > Systemvariablen,
Variable Path, Bearbeiten...

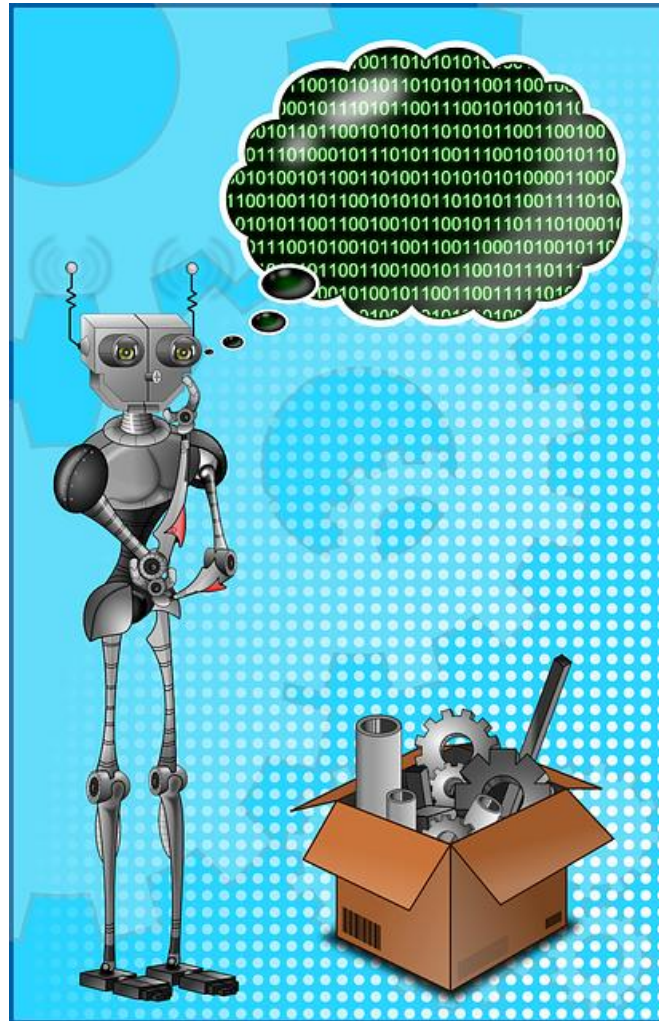
■ Konsolenfenster starten und vorbereiten:

Start > Alle Programme > Zubehör >
Eingabeaufforderung

■ Ins Verzeichnis des Quellprogramms navigieren:

cd Documents > cd ...

Muss man das alles von Hand machen?



Java-Entwicklungsumgebungen (1)

- bestehen (mindestens) aus
 - Source-Code-Editor
 - Übersetzer (*Compiler*)
 - Werkzeugen zum automatisierten Bau (*Build*) von Paketen
 - Werkzeugen zur Fehlersuche und -analyse (*Debugger*)
- helfen beim Entwickeln von Java-Programmen
 - „Schablonen“ für Anwendungen
 - Ändern von Code (Refactoring)
 - Anzeigen von Fehlermeldungen, Debugging
 - Automatisches Vervollständigen von Code
 - Design von grafischen Oberflächen
 - Umsetzen von UML-Diagrammen in Code
 - und, und, und...

Java-Entwicklungsumgebungen (2)

- ...gibt es wie Sand am Meer
 - **Eclipse** (www.eclipse.org)
 - NetWeaver Developer Studio (SAP)
 - **NetBeans** (netbeans.apache.org)
 - **IntelliJ IDEA** (www.jetbrains.com/idea/)
 - MS Visual Studio
 - JCreator
 - JDeveloper (Oracle)
 - JBuilder
 - ...



Java-Entwicklungsumgebungen (3)

■ Wir empfehlen entweder

- Eclipse (4.4 oder höher)
oder
- NetBeans (8.0 oder höher)
oder
- IntelliJ IDEA (14.0 oder höher)



■ Auswahl auf Basis von Erfahrung und/oder Einsatz im Ausbildungsbetrieb

Java-Entwicklungsumgebungen (4)

