

Programmierungsmethodik 1

Programmiertechnik

Einführung

Änderungshistorie

- 25.3.2016
 - kleinere Typos und missverständliche Formulierungen korrigiert

Herzlich Willkommen in der Informatik der
HAW Hamburg!

Ziel der Veranstaltung

Spaß am Programmieren lernen!

Ziel der Veranstaltung

- Die Vorgehensweise bei der Programmentwicklung („*Softwareentwicklung*“) vermitteln und einüben
- Konzepte und Sprachmittel einer aktuellen Programmiersprache (*Java*) vermitteln und einüben

Motivation

- Basisaufgabe der Informatik:

Die Welt mit formalen Methoden beschreiben und diese zur Problemlösung einsetzen.

Voraussetzungen

- Keine Programmierkenntnisse nötig, aber dafür folgendes:
 - Erfahrungen im elementaren Umgang mit einem Computer (Programme starten, Dateien mit Texteditor bearbeiten und speichern, Internetbrowser zum Dateidownload verwenden)
 - Hohe Motivation
 - Fähigkeit, systematisch und gewissenhaft zu arbeiten
 - Bereitschaft, ein Buch oder Online-Dokumentation zu lesen
 - Bereitschaft zum intensiven Üben



Was ist Programmieren?

- Computerprogramme schreiben.
- Genauer (Wikipedia: Programmierung):

Programmierung [...] bezeichnet die Tätigkeit, Computerprogramme zu erstellen. Dies umfasst vor Allem die Umsetzung (Implementierung) des Softwareentwurfs in Quellcode sowie – je nach Programmiersprache – das Übersetzen des Quellcodes in die Maschinensprache, meist unter Verwendung eines Compilers.

Was ist ein Computerprogramm?

- Eine Reihenfolge von Befehlen, die dem Computer sagen, was er machen soll.

Umfrage

<https://users.informatik.haw-hamburg.de/~abo781/gcrs/vote.html>





Organisation

Organisation

- Drei Säulen

Vorlesung

Praktikum

Prüfung

Organisation: Vorlesung

- keine Anwesenheitspflicht
- jeweils 4-stündig statt 3-stündig
 - daher entfallen die letzten Termine im Semester

Organisation: Praktikum

- Anwesenheitspflicht
- Bearbeitung der Aufgaben in 2er-Teams
 - Anmeldung bereits erfolgt (StiSys)
- Abnahme
 - Vorstellung der Lösung zum Praktikumstermin
 - erfolgreiche Abnahme aller Praktikumsaufgaben ist Prüfungsvoraussetzung

Organisation: Praktikum

Anforderungen an abgegebene Lösungen

- Bearbeitung aller Teilaufgaben
- keine Kompilierfehler, keine Compiler-Warnungen
- später zusätzlich: Code-Konventionen
- beide Teammitglieder können gesamte Lösung erläutern
- Einhalten der Code-Konventionen
- Präsentation auf dem Poolrechner
- Entwicklungsumgebung: freigestellt, aber Support nur für Eclipse

Organisation Praktikum

- Abnahme in den Praktika jeweils
 - ein Professor + ein wissenschaftlicher Mitarbeiter
- bei uns
 - Prof. Axel Schmolitzky
 - Prof. Philipp Jenke
 - Norbert Kasperczyk (wissenschaftlicher Mitarbeiter)
- Sprechstunde
 - nach Vereinbarung (per Mail oder im Praktikum)



Organisation: Prüfung

- zwei Prüfungen am Semesterende
 - Klausur (schriftlich, Papier)
 - Rechnerprüfung (Programmieren am Rechner)
- zum Üben: Mid-Term
 - Probeklausur in der Mitte des Semesters
 - Teilnahme freiwillig
 - kein Einfluss auf die Note am Semesterende

Organisation: EMIL

- E-Learning-Plattform der HAW Hamburg
- zentraler Anlaufpunkt für alle Informationen und Materialien
- URL: *<http://www.elearning.haw-hamburg.de>*
 - Login: HAW-Login
- Unser Lernraum
 - URL: *<http://www.elearning.haw-hamburg.de/course/view.php?id=14522>*
 - Suche: Jenke, Programmiermethodik 1
 - Registrierung für Lernraum
 - Selbsteinschreibeschlüssel: *PM1PTSS16*



Organisation: Tutorium

- Angebot: wöchentliches Tutorium
 - jeden Montag (ab kommender Woche)
 - 16 Uhr
 - Raum 11.02
- zwei Tutoren: Studierende aus höherem Semester
 - Helena Lajevardi
 - Lennart Borchert
- Inhalte
 - zusätzliche Übungsaufgaben
 - Hilfe bei Fragen zu Praktikumsaufgaben
 - Unterstützung beim Ankommen an der Hochschule
 - z.B. Selbstorganisation, richtig Lernen, ...



Lernen

- Ziele
 - Bestehen der Prüfung
 - erfolgreicher Abschluss des Studiums
 - gute Prüfungsnote/gute Abschlussnote
- Lernen
 - Wie lerne ich?
 - Was muss ich lernen?
 - Wieviel Zeit benötige ich?
 - ...

Ererbte Methoden

Methodennutzung: Die JVM sucht zuerst in der Klasse selbst, dann in der Basisklasse, dann in deren Basisklasse usw.

```
SpeicherZaehler speicherZaehler = new SpeicherZaehler ();
speicherZaehler.erhoehen(); // erbt von Zaehler
speicherZaehler.speichern();
speicherZaehler.reset(); // erbt von Zaehler
speicherZaehler.wiederherstellen();
```

Redefinition abgeleiteter Methoden

```
/**
 * Ein beschränkter Zähler verhält sich wie ein Zähler,
 * der aber eine Obergrenze für seine Werte hat.
 */
```

```
public class BeschränkterZaehler extends Zaehler {
```

abgeleitete Methoden können in abgeleiteten Klassen neu definiert werden, das ist aber nicht möglich, wenn die Methode in der Basisklasse bereits definiert ist.

```
/**
 * Grenzwert.
 */
private final int grenze;
```

```
/**
 * Konstruktor.
 */
```

```
public BeschränkterZaehler(int grenze) {
    this.grenze = grenze;
}
```

```
/**
 * Getter.
 */
```

```
public int getGrenze() {
    return grenze;
}
```

```
@Override
public void erhoehen() { // gleiche Signatur
    if (getWert() < grenze) { // neuer Rumpf
        setWert(getWert() + 1);
    }
}
```

Rede
fini-
tion

über-
laden

```
/**
 * wenn Parameter List nicht gleich ist, redet  
wird die Methode überladen.
 */
```

```
public void erhoehen(int schrittweite) {
    wert += schrittweite;
}
```

eine abgeleitete Klasse kann Methoden redefi-
nieren, die bereits in der Basisklasse definiert sind.

Einschränken einer Basisklasse

abgeleitete Klassen können die Funktionalität
der Basisklasse erweitern oder ändern,
aber keine Calls einschränken.

```
private void erhoehen() { ... } // Fehler
```

Dynamisches Binden redefinierter Methoden

abgeleitete Klassen sind kompatibel
zu Basisklassen (vgl. auch Interfaces).
redefinierter Methoden werden dynamisch
gebunden.

```
Zaehler zaehler = new BeschränkterZaehler(5);
for (int i = 0; i < 10; i++) {
```

- Fachliche Anforderung (z.B. PM1/PT)
- Wahrnehmung fachlicher Anforderung (Was wird von mir gefordert?)
- Selbsteinschätzung (Was gelingt mir und was nicht?)
- Selbstreflexion (Warum gelingt mir etwas nicht?)
- Individuelle Lernplanung (Was kann ich tun, um in diesem Fach voranzukommen?)
- (Abstimmung) Zeitmanagement (Wieviel Zeit habe ich realistisch! für Lernen und Studium?)
- Lern- und Prüfungsstrategie (Wieviel lerne ich und wie bereite ich mich auf die Prüfung vor?)



Programmieren

Es gibt viele Programmiersprachen



Höhere Programmiersprachen

- Kategorisierung anhand der zentralen Programmierparadigmen

im Kern stehen
Anweisungen, die
nacheinander
abgearbeitet
werden

Imperative
Sprachen

Deklarative
Sprachen

u.a. Funktionale
Sprachen

im Kern stehen
Funktionen, die
Werte berechnen

Objektorientierte
Sprachen

im Kern stehen
Objekte mit
Eigenschaften, die
Nachrichten
austauschen

Was nehmen wir

- Wir nehmen Java

- Warum Java? Java ist ...

- modern
- einfach
- weit verbreitet
- für viele verschiedene Betriebssysteme verfügbar
- kostenlos
- für alle Arten von Problemen flexibel einsetzbar
- im Department Informatik in vielen Praktika im Einsatz



Objektorientierte
Sprachen

im Kern stehen
Objekte mit
Eigenschaften, die
Nachrichten
austauschen

Java

- Kostenlos erhältlich (für alle gängigen Betriebssysteme)
- aktuelle Version: Java 8

- Download:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

- JDK vs. JRE
 - Achtung: Wir benötigen JDK (Java Development Kit)
 - nicht JRE (Java Runtime Environment)

Literaturempfehlungen

- Reinhard Schiedermeier: Programmieren mit Java, 2. Auflage, Pearson Studium, 2010
Sehr gut lesbares Lehrbuch zur Einführung der grundlegenden Konzepte
- Kathy Sierra, Bert Bates: Java von Kopf bis Fuß, 2. oder 3. Auflage, O'Reilly
Spielerische Einführung in Java und objektorientierte Konzepte
- Philip Ackermann: Schrödinger programmiert Java, Galileo Computing
Ähnlich dem vorherigen Buch, ebenfalls "etwas andere" Didaktik
- Christian Ullenboom: Java ist auch eine Insel, 10. Auflage, Galileo Computing, 2012
Ausführliches Standardwerk zum Lernen und Nachschlagen mit vielen Beispielen Online-Version / Kostenloser Download unter <http://openbook.rheinwerk-verlag.de/javainsel/>
- Offizielle JAVA-Referenz: <http://www.oracle.com/technetwork/java/index.html>
Für das Praktikum ist das Development Kit (JDK) der Java Standard Edition (SE) nötig

Was umfasst „Programmieren“?

- Klären, welches Problem das neue Programm überhaupt lösen soll
 - „Anforderungsanalyse“: Was soll das Programm genau tun?
- Aufbau des neuen Programms überlegen
 - „Entwurf“: Wie soll das Programm strukturiert sein?
- Programmcode („Sourcecode“) in einer Programmiersprache schreiben
 - „Implementierung“
- Sicherstellen, dass das Programm zuverlässig funktioniert
 - „Test“
- Dokumentieren aller Schritte!

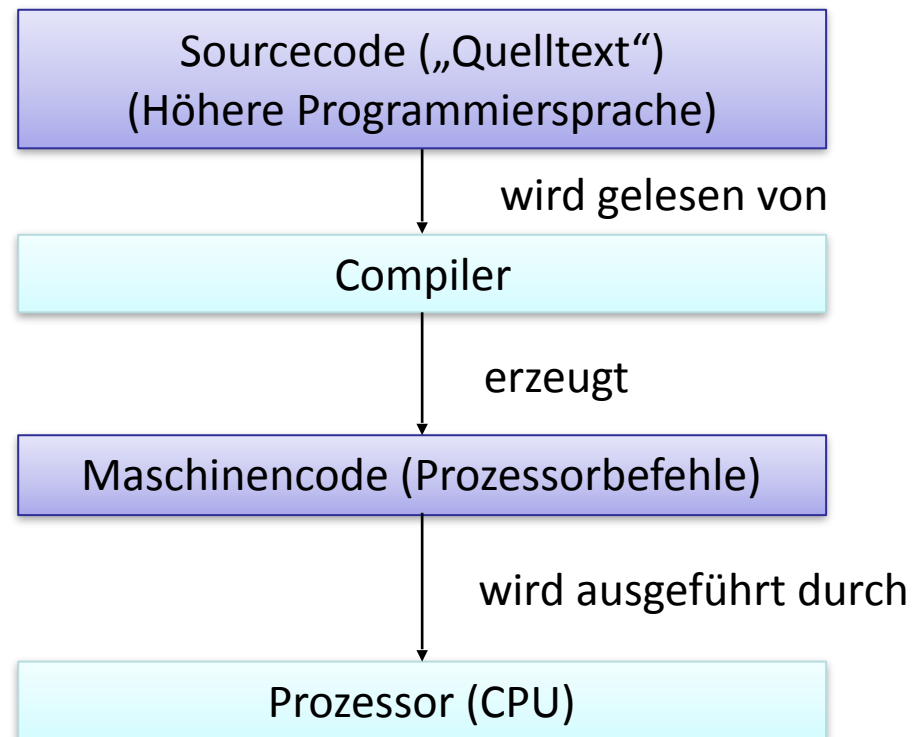
Beispielprogramm "Hallo Welt!"

- Anforderungsanalyse: „*Hallo Welt!*“ ausgeben
- Entwurf: Einfache Ausgabeanweisung verwenden
- Implementierung: Java-Sourcecode
 - Datei: *HalloWelt.java*

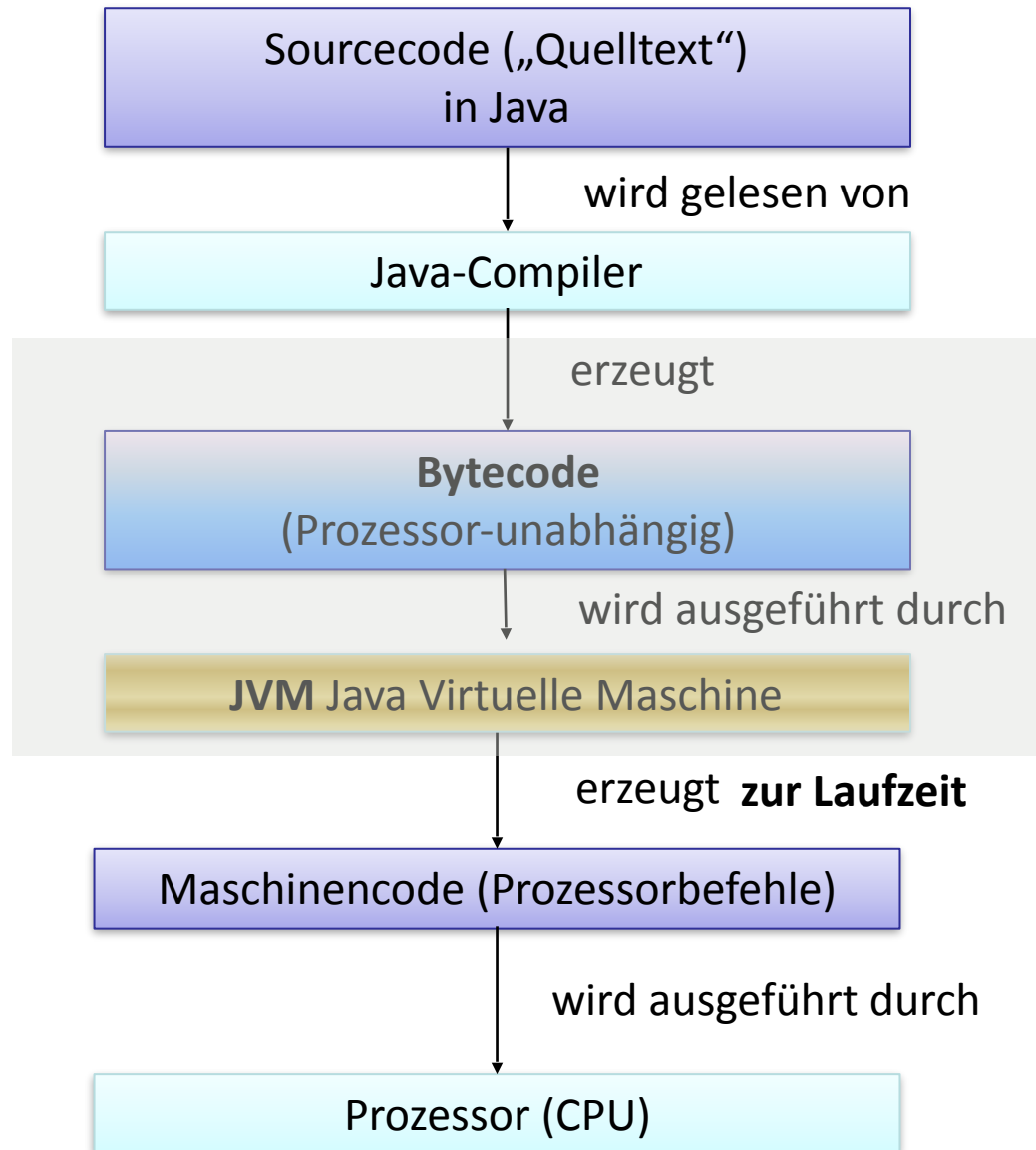
```
public class HalloWelt {  
    public static void main(String[] args) {  
        System.out.println("Hallo, Welt!");  
    }  
}
```

Wie kommen wir vom Quellcode zum
laufenden Programm?

Funktion eines Compilers



Spezialfall JAVA-Compiler



Beispielprogramm "Hallo Welt!"

- Sourcecode in Datei speichern
HalloWelt.java in *<Verzeichnis>*
- In Verzeichnis wechseln
cd <Verzeichnis>
- Java-Compiler aufrufen (Sourcecode → Bytecode):
javac HalloWelt.java
 - Ergebnis: neue Datei *HalloWelt.class*
- Java-VM *HalloWelt.class* (→ den Bytecode) ausführen lassen
java HalloWelt




Entwicklungsumgebung

Entwicklungsumgebung: Eclipse

- für größere Programme ist ein Editor hilfreich
 - Verwalten mehrerer Programm-Dateien
 - Kompilieren
 - Ausführen
 - Unterstützung bei der Programmierung (z.B. Syntax-Highlighting)
- Lösung: (Integrierte) Entwicklungsumgebung
 - auch Integrated Development Environment oder IDE
 - wir verwenden Eclipse
 - aktuelle Version: Eclipse Mars (4.4)



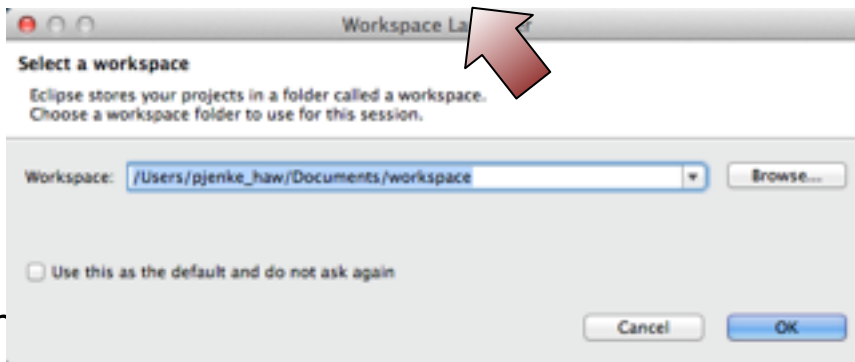
Installation und Start

- Download
 - <http://www.eclipse.org/>
 - Version: Eclipse IDE for **Java** Developers
 - Installation = Entpacken des Pakets
 - ggf. Verschieben des Verzeichnisses *eclipse* an einen zentralen Ort
 - Starten
 - Wechsel in das *eclipse*-Verzeichnis
 - Starten des Programms *eclipse(.exe)* (Windows)
- Achtung: gibt es auch für andere Programmiersprachen
- 

Erste Konfiguration

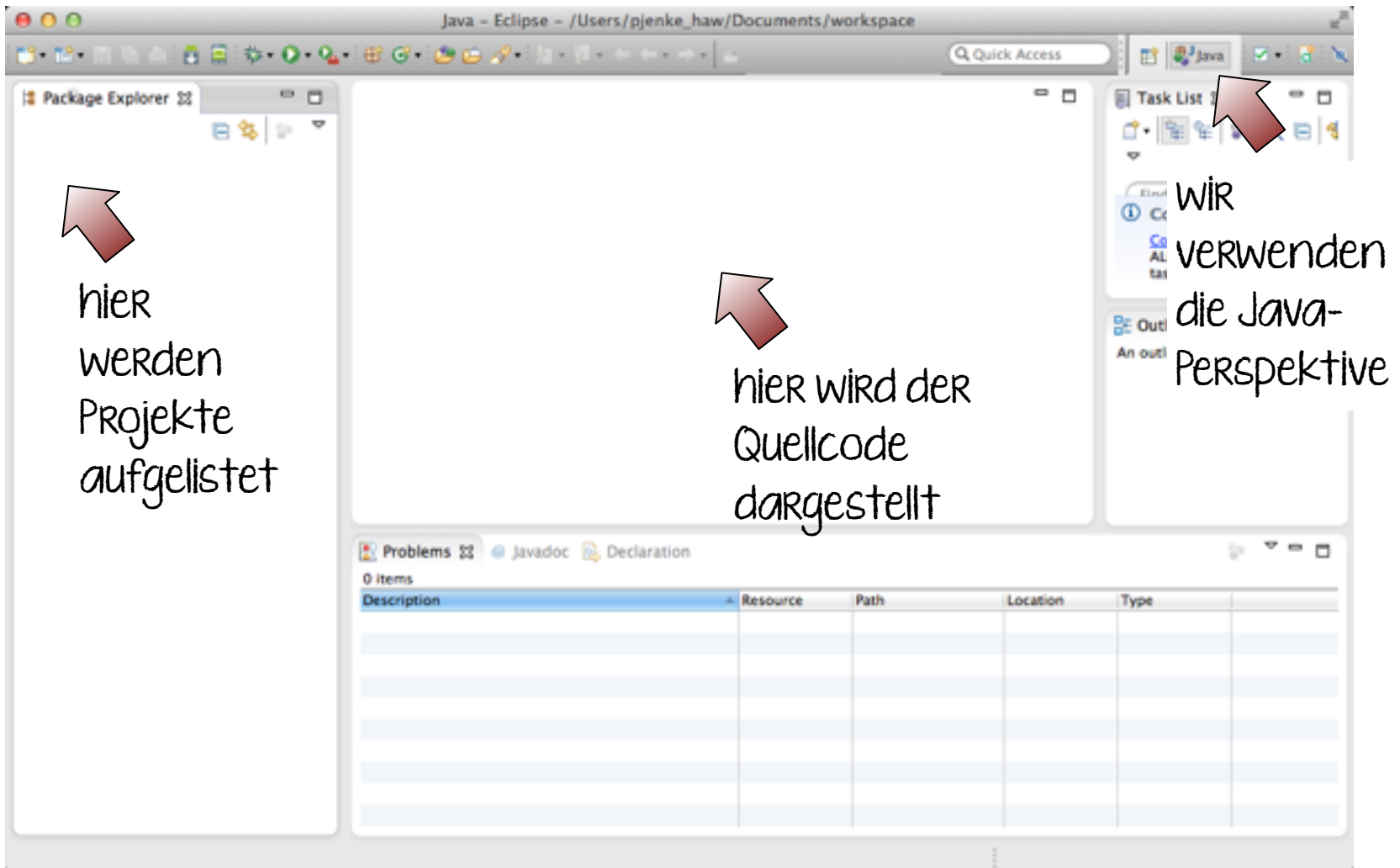
Achtung: Die Screenshots sehen je nach Version etwas anders aus!

- Setzen des Workspace

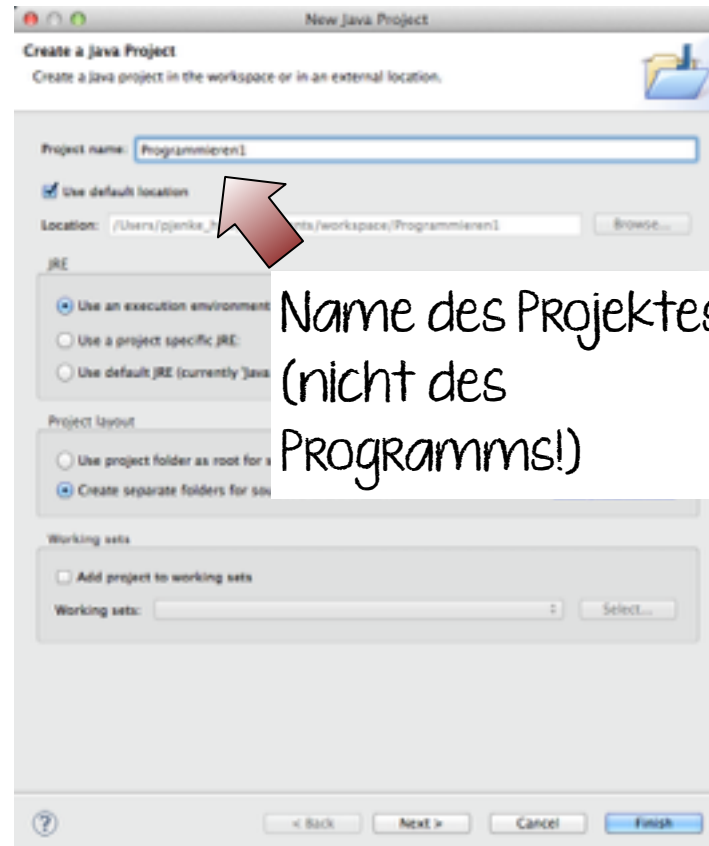
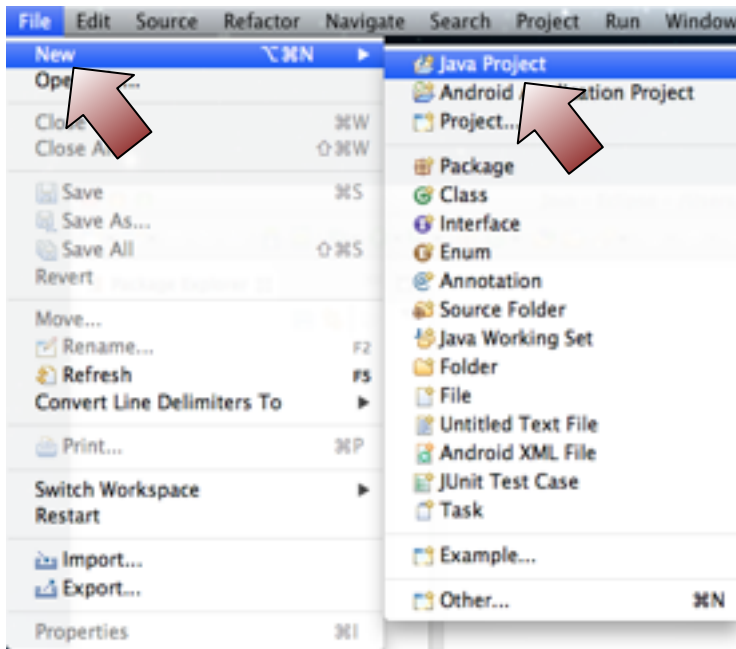


- in dieser die Projekte abgelegt

Die Perspektive



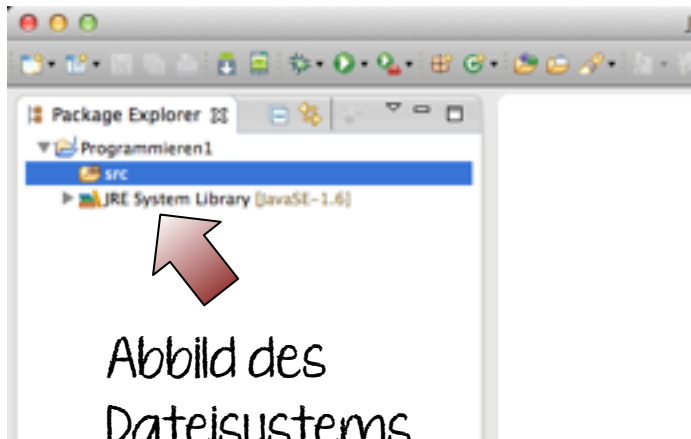
Neues Projekt Anlegen



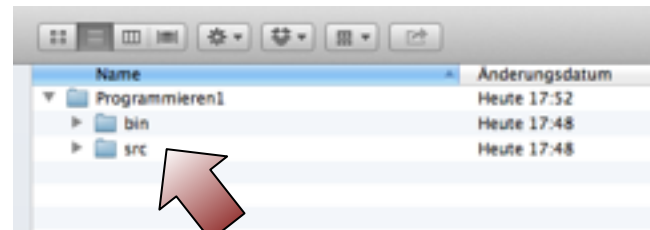
Neues Projekt

– in Eclipse

– im Dateisystem



Abbild des
Dateisystems
bzgl. des
Quellcodes

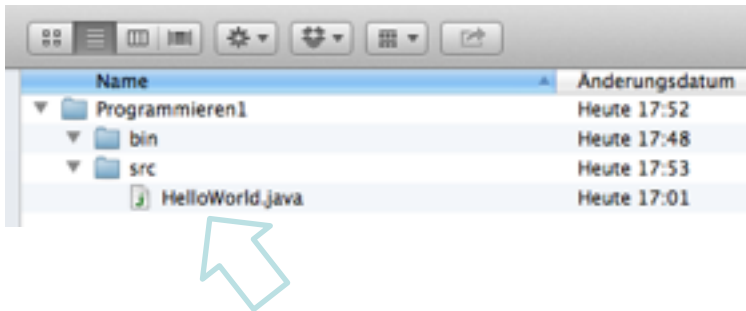


```
localhost:Programmieren1 pjenke_haw$ ls -la
total 32
drwxr-xr-x  8 pjenke_haw  staff   272 13 Mär 17:52 .
drwxr-xr-x  5 pjenke_haw  staff   170 13 Mär 17:52 ..
-rw-r--r--@ 1 pjenke_haw  staff  6148 13 Mär 17:54 .DS_Store
-rw-r--r--  1 pjenke_haw  staff   295 13 Mär 17:48 .classpath
-rw-r--r--  1 pjenke_haw  staff   373 13 Mär 17:48 .project
drwxr-xr-x  3 pjenke_haw  staff   102 13 Mär 17:48 .settings
drwxr-xr-x  4 pjenke_haw  staff   136 13 Mär 17:54 bin
drwxr-xr-x  4 pjenke_haw  staff   136 13 Mär 17:53 src
localhost:Programmieren1 pjenke_haw$
```

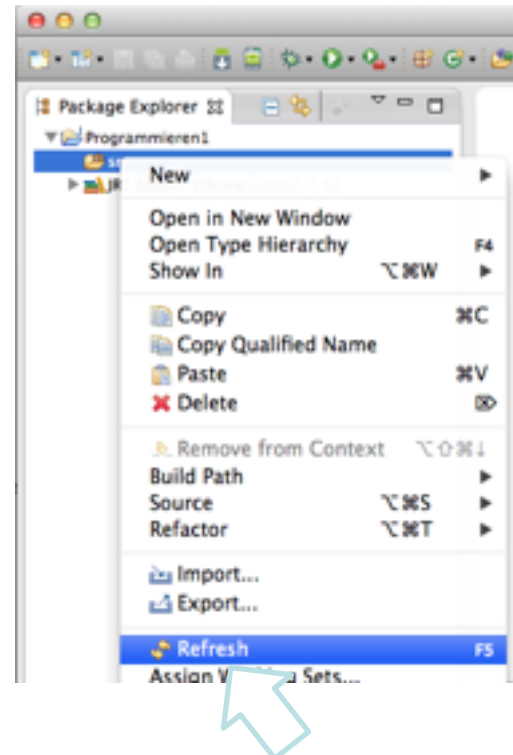
zusätzlich:
Projektdateien

Einfügen von HalloWelt.java

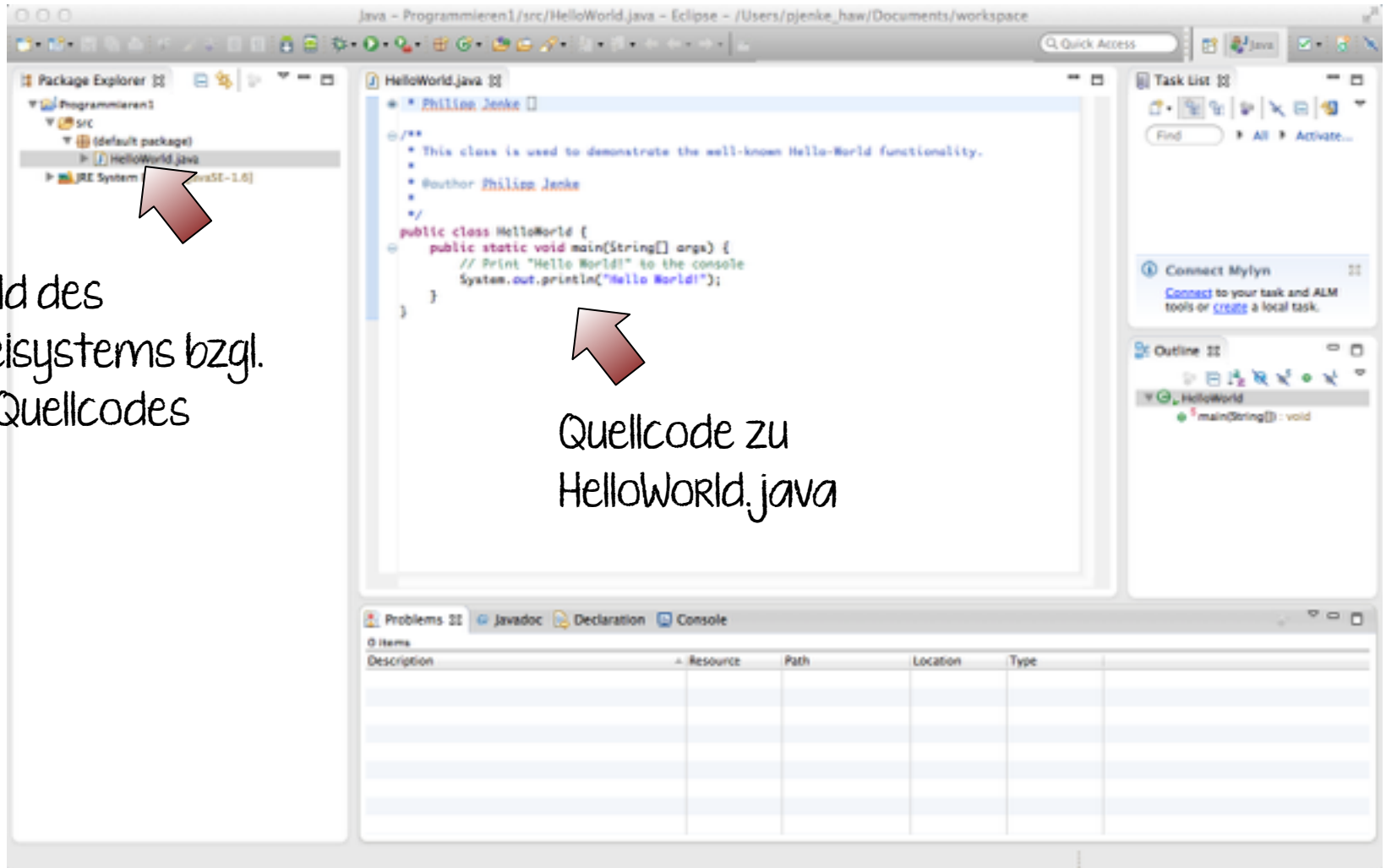
- im Dateisystem



- in Eclipse
 - Rechtsklick auf den Ordner *src*



HelloWorld in Eclipse



Kompilieren in Eclipse

- wird von Eclipse automatisch gemacht
- Blick in das Dateisystem:

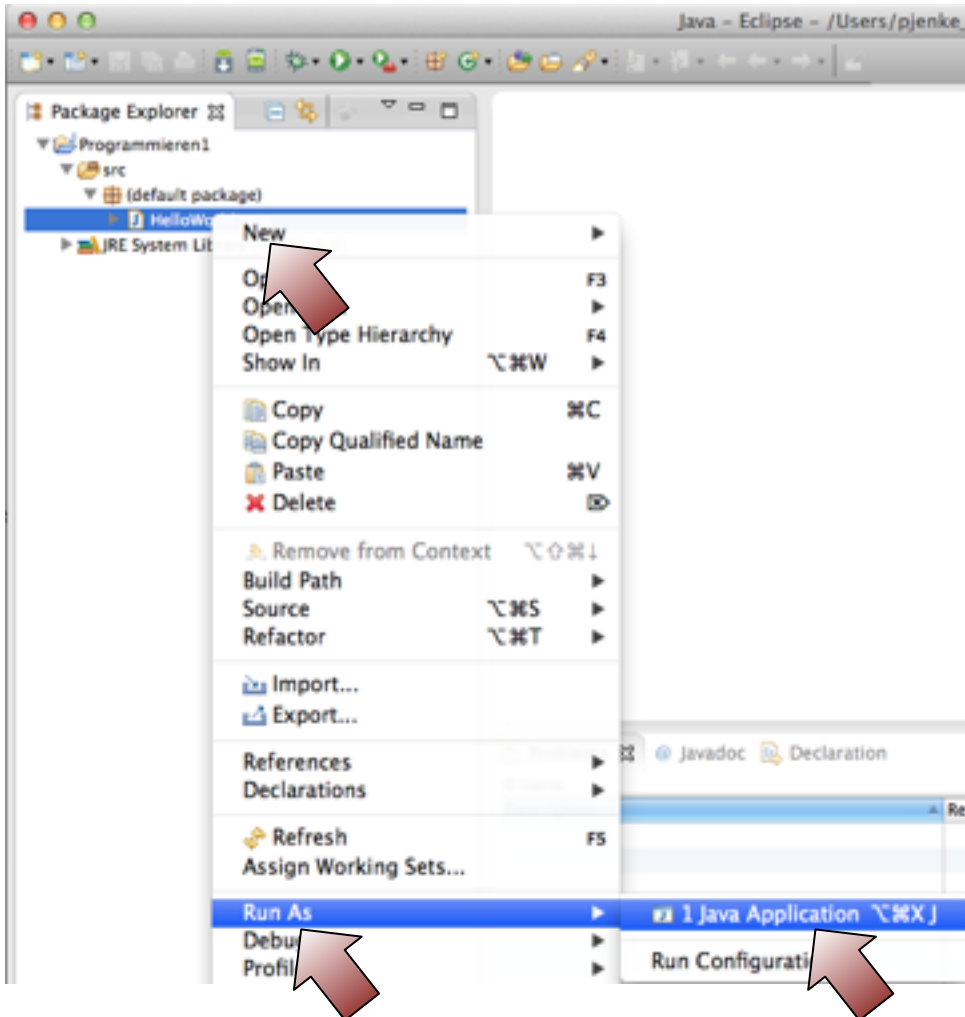


Name	Änderungsdatum	Größe	Art
▼ Programmieren1	Heute 17:52	--	Ordner
▼ bin	Heute 17:54	--	Ordner
HelloWorld.class	Heute 17:54	534 Byte	Java-...s-Datei
▼ src	Heute 17:53	--	Ordner
HelloWorld.java	Heute 17:01	471 Byte	Java Source

Bytecode

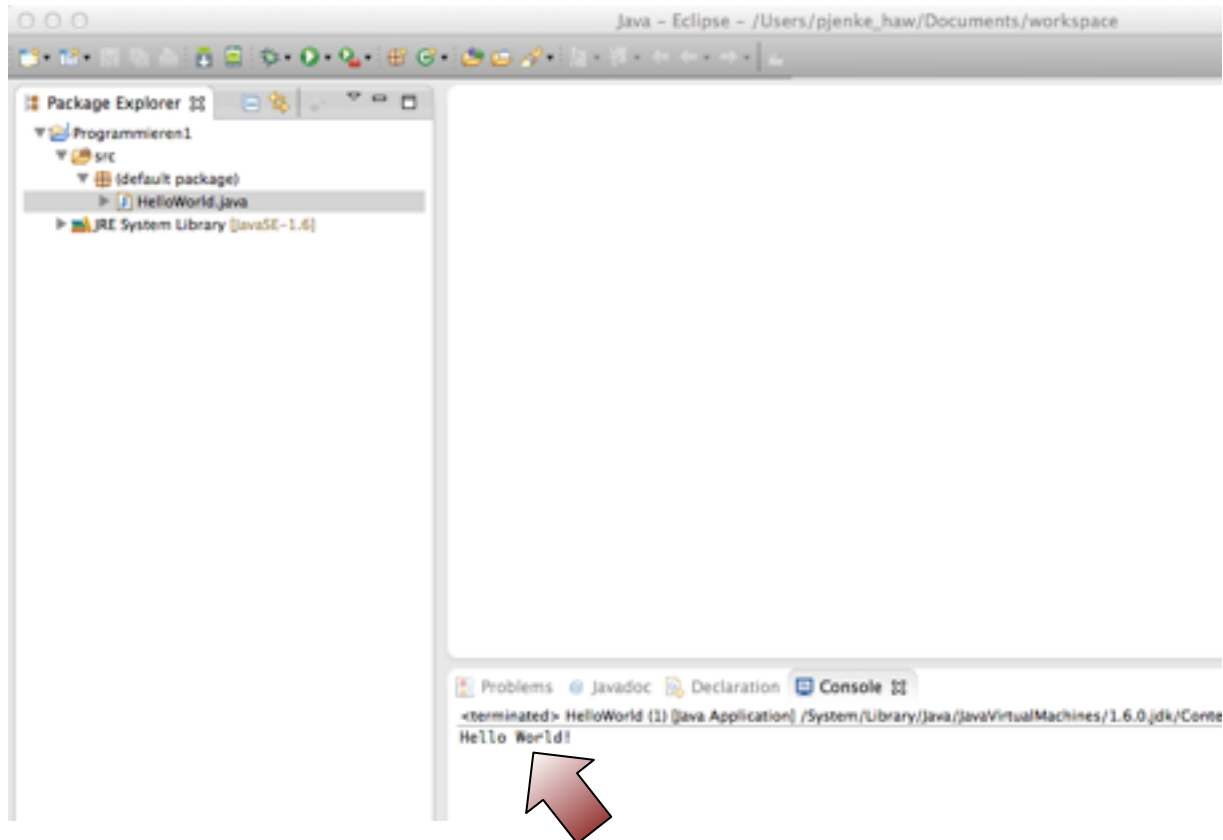
Ausführen eines Programms

- Rechtsklick auf *HelloWorld.java*

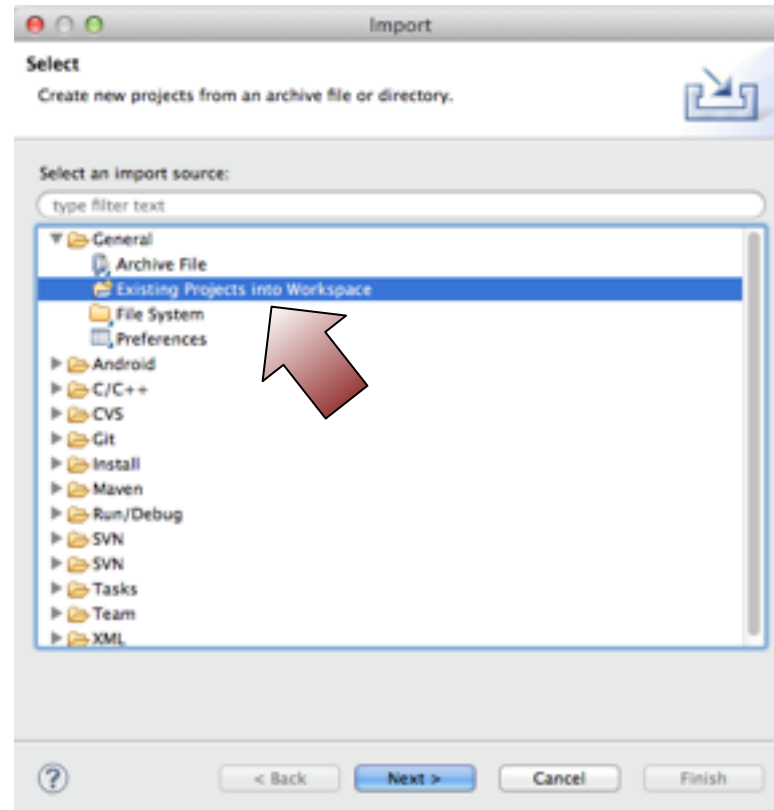
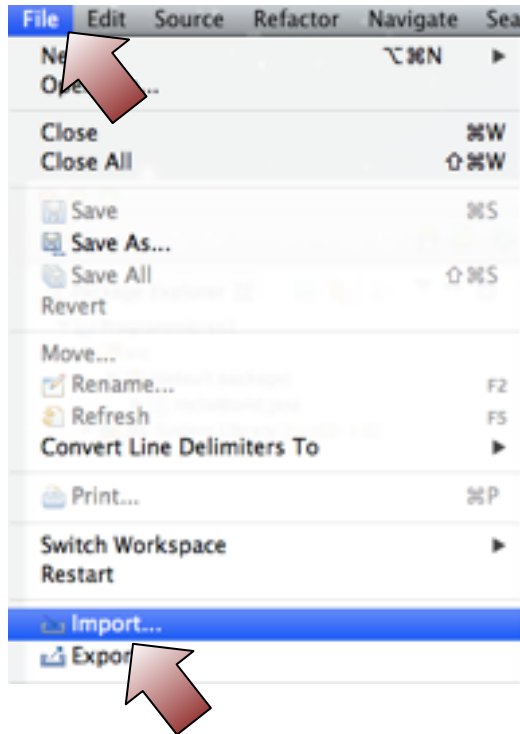


Ausführen eines Programms

- Konsolenausgabe

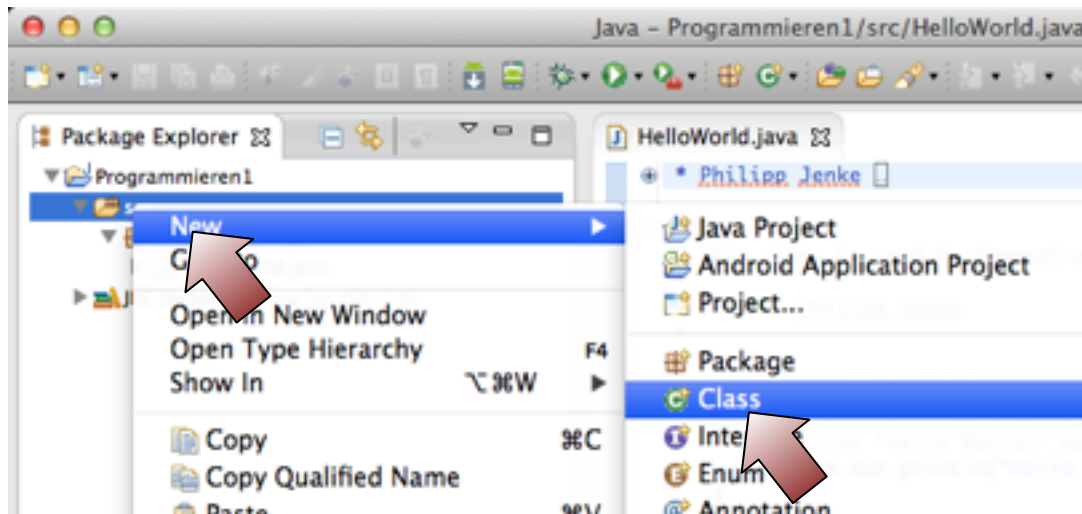


Importieren eines Projektes

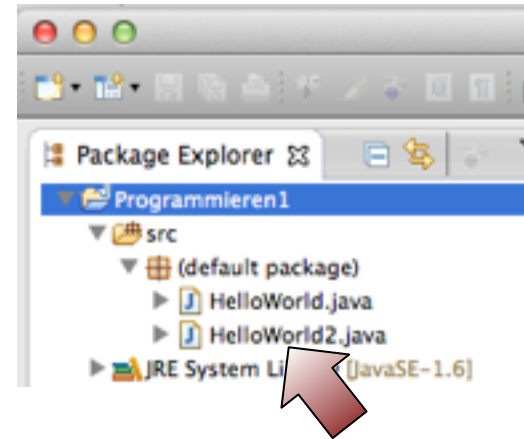
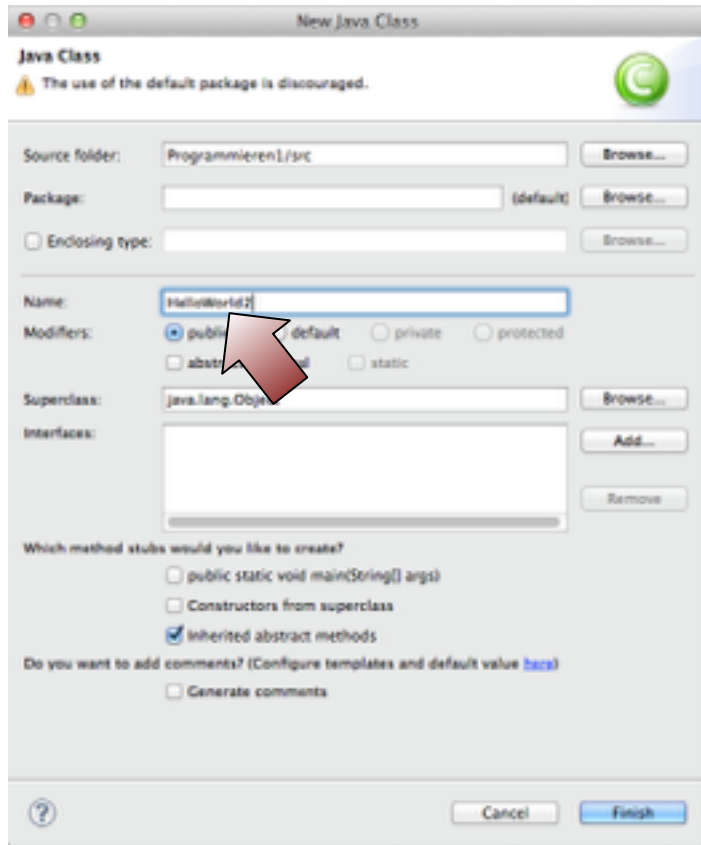


Anlegen eines neues Programms

- Rechtsklick auf den Ordner *src*
- Bei uns zunächst: Programm = Klasse (engl. Class)



Anlegen Neues Programm



Einschub: Package

- jede Java-Klasse sollte einem Package zugewiesen werden
- Packages können frei gewählt werden
- Beispiel: `aufgabenblatt1`;
 - Zuweisung im Quellcode (.java): `package aufgabenblatt1`;
 - in Eclipse: steht im Package-Explorer
- Package muss sich auch im Verzeichnisbaum widerspiegeln

Was ist was?

- `class`
 - kündigt ein neues Programm an
 - Name `Summe` folgt direkt danach
 - Programmname und Dateiname (*Summe.java*) müssen gleich sein!
- geschweifte Klammern `{`
 - grenzen zusammen gehörende Quelltext-Abschnitte ab („Blöcke“)
 - Klammern nach der `class`-Zeile umfassen das gesamte Programm.
- `main`
 - markiert den Start der eigentlichen Anweisungen
 - wird auch als „Einsprungpunkt“ bezeichnet, weil hier die Ausführung des Programms beginnt:

```
public static void main(String[] args)
```

Was ist was?

- eigentliches Programm
 - wieder zwischen geschweiften Klammern (Ende der `main`-Zeile)
 - besteht aus Anweisungen (meist eine Zeile)
- Anweisungen
 - nun folgen verschiedenen Anweisungen
 - enden mit einem Semikolon (;)
 - z.B. `System.out.println` = Ausgabe auf der Konsole

`System.out.println(„Text“);`

Zusammenfassung

- Einführung
- Programmieren
- Organisation
- Java
- Erstes Programm
- Eclipse