

Software Design

Übungsstunde 1 : Einführung Java/Eclipse

Stefan Müller

`stefan.mueller@inf.ethz.ch`

Department of Computer Science ETH Zurich, Switzerland

Februar 20, 2012

■ Name :

- Stefan Müller (stefan.mueller@inf.ethz.ch)

■ Tätigkeit:

- Doktorand bei Prof. Gustavo Alonso

■ Forschungsgebiet:

- Scientific Data Processing
- Euclid: ESA Mission to Map the Dark Universe

■ Mein Vorgänger:

- Louis Woods
- Diese Folien basieren auf der Vorlage von Louis Woods

- Folien nach Übungsstunde online:

<https://web.imvs.technik.fhnw.ch/zope/sd>

- Offizielle Übungsstunde : Mo, 10:15-11:00, IFW C 42
- Fragestunde : Mo, 11:10-11:55, IFW C 42
- Büro: CAB E 77.2

■ Übungen

- Übungen 1-3: **Alle obligatorisch**
Jeder Student löst die Übungen **individuell**
- Übungen 4-11: 75% (min. 6 der 8 bearbeitet)
2er-Teams

■ Abgabe

- Deadline: Zwei Wochen nach Herausgabe, **Montags, 8:00**
- Übungen 1-3: **Quellcode** per E-Mail an stefan.mueller@inf.ethz.ch
 - ➡ Subject: SD:netzname_ueb#
Bsp: SD:smuelle_ueb01
 - ➡ Ihr erhaltet eine Bestätigung innert 24h

■ Abgabe

- Übungen 4-11:

Source JAR + Ausführbares JAR

per E-Mail an smuelle@student.ethz.ch



Subject: SD:netzname1_netzname2_ueb#

Ersetze: # , netzname 1, netzname 2



JAR-Datei Benennung analog:

netzname1_netzname2_src_ueb# .jar

netzname1_netzname2_exec_ueb# .jar



Optional: Code Listings in Papier-Form für
Korrekturen direkt im Code

- Code-Konventionen von Java einhalten
- Kann in Eclipse automatisiert werden

Programm für heute

- Installation von Java + Eclipse
- Erste Schritte mit Eclipse (Demo)
- Hello World in Java
- Vektor-Klasse

■ Java Development Kit (JDK) installieren

- **Version 7** (1.7) oder 1.6 (in Versionen ≤ 1.4 fehlen Features, die wir im Kurs verwenden)
- **Windows:** Download von <http://www.oracle.com/technetwork/java/javase/downloads/index.html> (Version JDK ohne JavaFX , J2EE oder NetBeans)
- **Linux:** Installation via Package-Manager (apt-get, Synaptic etc.) oder Download von SUN
- **MacOS X:** Java 1.5 in Developer-Tools enthalten 1.6 nur für 64-bit Intel-Plattform

■ Achtung:

- **Java Runtime Environment (JRE)** alleine reicht nicht: Compiler fehlt!
- JRE ist in JDK enthalten

Hello World Programm in Java

- Programm-Listing in Text-Editor eingeben und als Datei 'HelloWorld.java' abspeichern.
- Achtung: Dateiname muss mit dem Name der Klasse übereinstimmen.

```
public class HelloWorld {  
  
    public static void main(String args[]) {  
        System.out.println("Hello World!");  
    }  
  
}
```

HelloWorld.java

Hello World Programm in Java

■ Name der Klasse “HelloWorld”

■ Programmeinstiegspunkt: `main(...)`

- Die Java **V**irtual **M**achine (VM) sucht die statische Methode “`static void main(String[])`” in der beim Start angegebenen Klasse.
- Array **args** enthält Kommandozeilen-Argumente beim Aufruf

```
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello World!");  
    }  
}
```

Klasse System

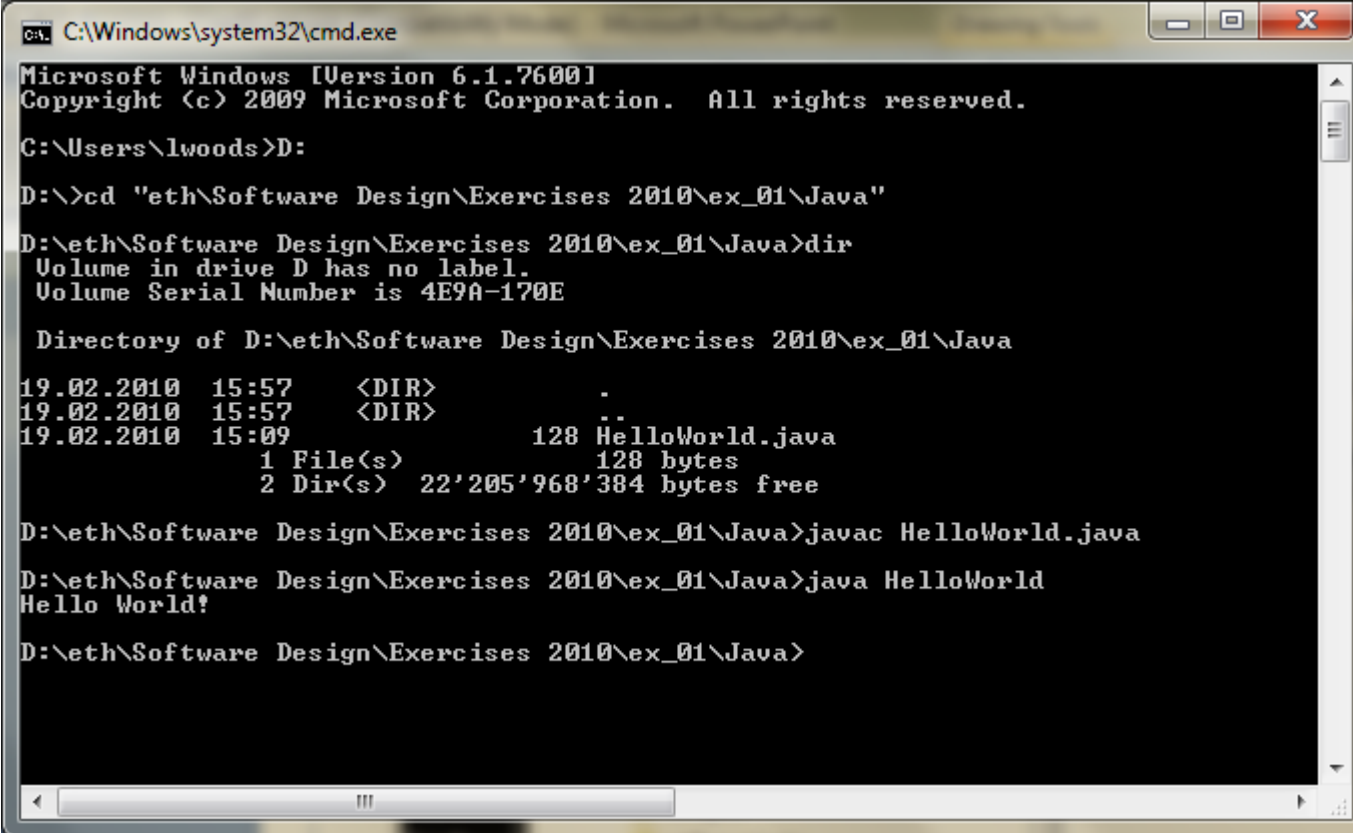
Stat. Feld (Klassenvariable)

Klassen-Methode

public
Sichtbarkeitsbereich (alle)

Aufruf von Java von der Kommando-Zeile

- Die Java VM (java.exe), sowie der Compiler (javac.exe) können von der Kommando-Zeile aufgerufen werden.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\lwoods>D:

D:\>cd "eth\Software Design\Exercises 2010\ex_01\Java"

D:\eth\Software Design\Exercises 2010\ex_01\Java>dir
Volume in drive D has no label.
Volume Serial Number is 4E9A-170E

Directory of D:\eth\Software Design\Exercises 2010\ex_01\Java

19.02.2010  15:57    <DIR>          .
19.02.2010  15:57    <DIR>          ..
19.02.2010  15:09                128 HelloWorld.java
               1 File(s)                128 bytes
               2 Dir(s)  22'205'968'384 bytes free

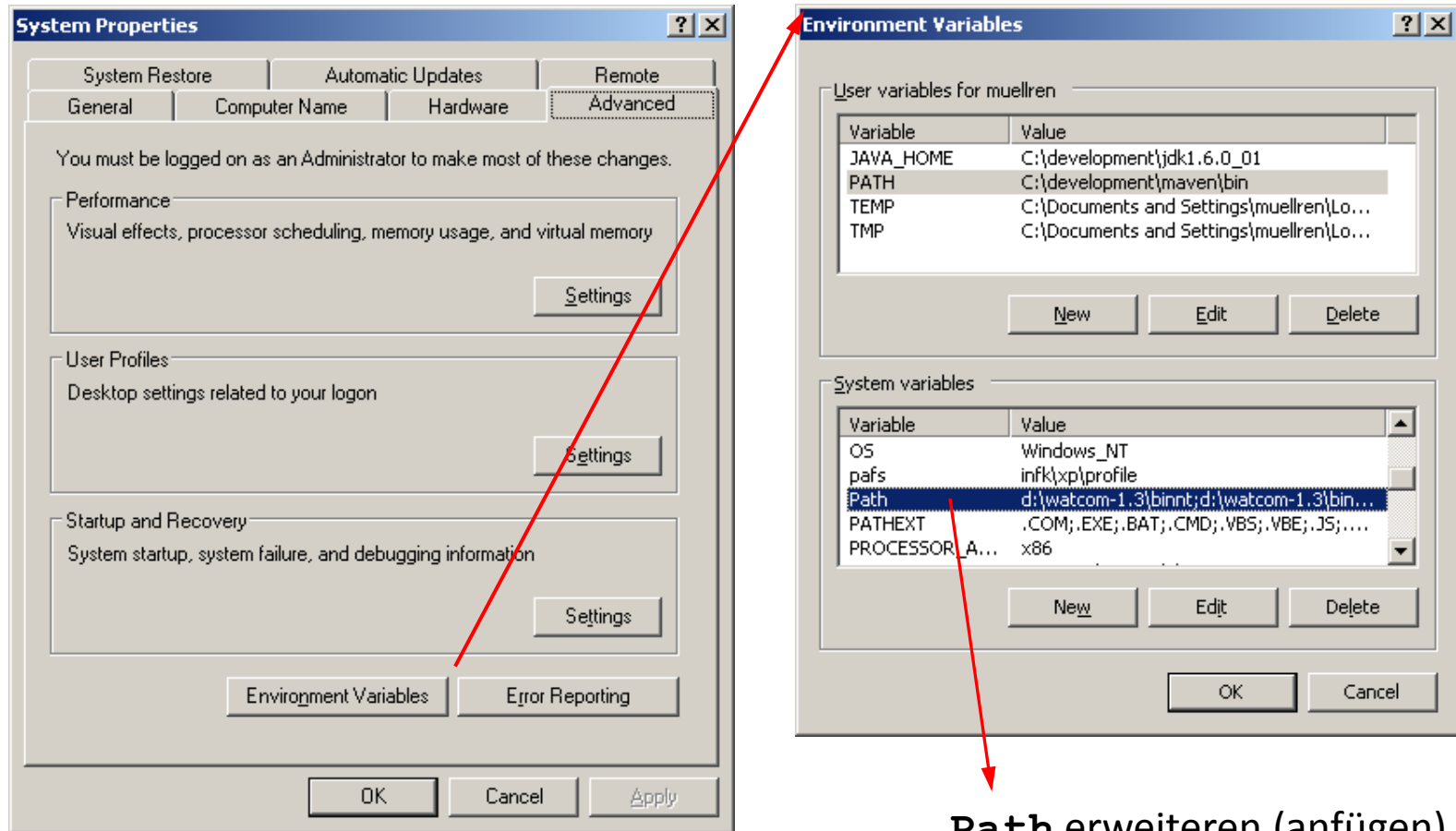
D:\eth\Software Design\Exercises 2010\ex_01\Java>javac HelloWorld.java

D:\eth\Software Design\Exercises 2010\ex_01\Java>java HelloWorld
Hello World!

D:\eth\Software Design\Exercises 2010\ex_01\Java>
```

Erweitern des Suchpfades (Windows)

- Das 'bin' Verzeichnis des JDK muss dem PATH Suchpfad hinzugefügt werden: Unter : Arbeitsplatz → Eigenschaften



Path erweitern (anfügen)

`" ; C : \<INSTALLATIONS-VERZEICHNIS> \Java \jdk1.6.0_18 \bin ; "`


Eclipse – IDE







Eclipse Downloads



Packages Developer Builds Projects



Compare Packages Older Versions Eclipse Indigo (3.7.1) Packages for Windows

 **Eclipse IDE for Java Developers**, 128 MB
Downloaded 3,686,216 Times [Details](#) **Empfehlung**  [Windows 32 Bit](#)
[Windows 64 Bit](#)



 **Eclipse IDE for Java EE Developers**, 212 MB
Downloaded 2,601,317 Times [Details](#)  [Windows 32 Bit](#)
[Windows 64 Bit](#)


 **Eclipse Classic 3.7.1**, 174 MB
Downloaded 1,241,158 Times [Details](#) [Other Downloads](#)  [Windows 32 Bit](#)
[Windows 64 Bit](#)

 **Eclipse IDE for C/C++ Developers (includes Incubating components)**, 107 MB
Downloaded 772,235 Times [Details](#)  [Windows 32 Bit](#)
[Windows 64 Bit](#)

 **Eclipse IDE for JavaScript Web Developers**, 110 MB
Downloaded 290,155 Times [Details](#)  [Windows 32 Bit](#)
[Windows 64 Bit](#)

 **Eclipse Modeling Tools**, 271 MB
Downloaded 136,528 Times [Details](#)  [Windows 32 Bit](#)
[Windows 64 Bit](#)

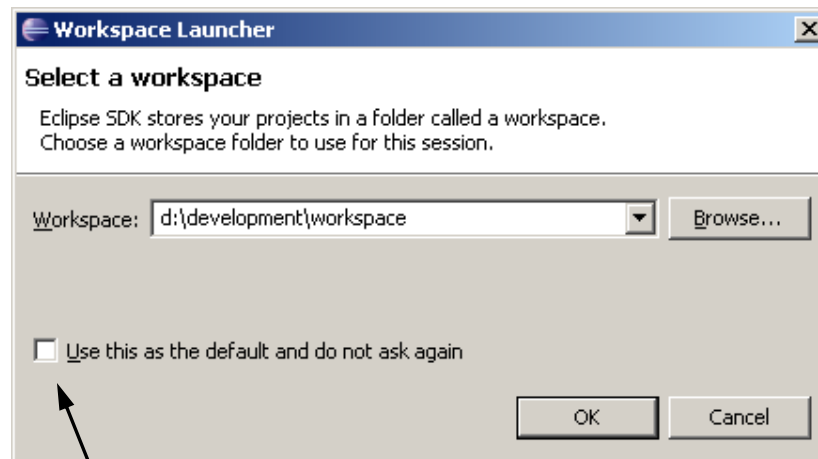
 **Eclipse IDE for Java and Report Developers**, 242 MB
Downloaded 127,803 Times [Details](#)  [Windows 32 Bit](#)
[Windows 64 Bit](#)

Eclipse for RCP and RAP Developers, 104 MB  [Windows 32 Bit](#)

- Download:
<http://www.eclipse.org/downloads>
- Eclipse ist ein Open-Source Framework und wird primär als Integrated Development Environment (IDE) verwendet
- Eclipse ist modular und durch Plugins erweiterbar
- Man kann Eclipse in verschiedenen Konfigurationen herunterladen

Eclipse – Workspaces

- Workspace: Arbeitsplatz enthält Projekte
- Wo ist mein Workspace? Wird beim Start von Eclipse angezeigt.
 - Mehrere Workspaces möglich

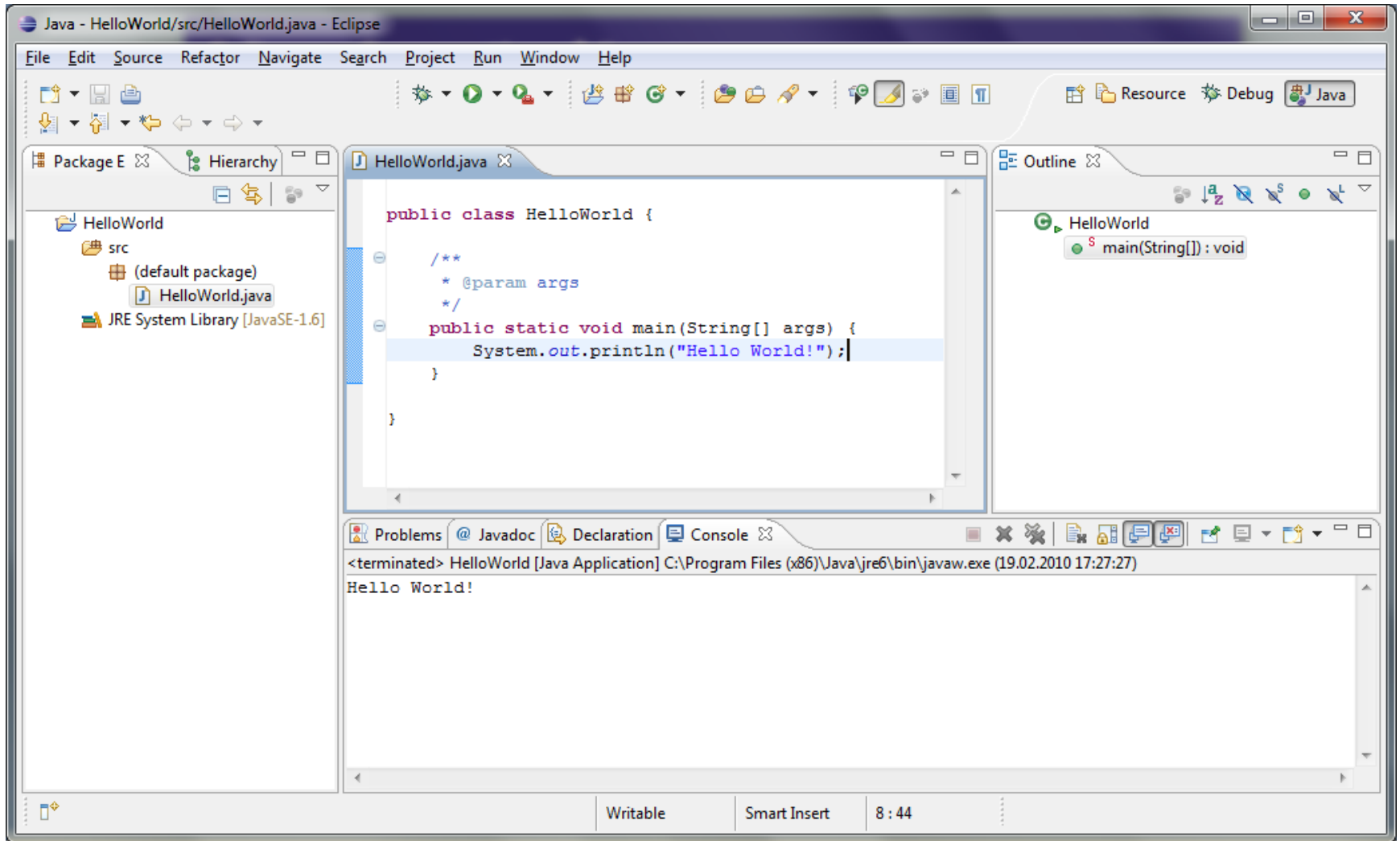


Diesen Workspace als Standard definieren. Dann wird Dialog ausgeblendet.

Später ändern unter:

Window → Preferences → General → Startup and Shutdown → Workspaces
→ Prompt for workspace on startup

Eclipse



■ Programm ausführen

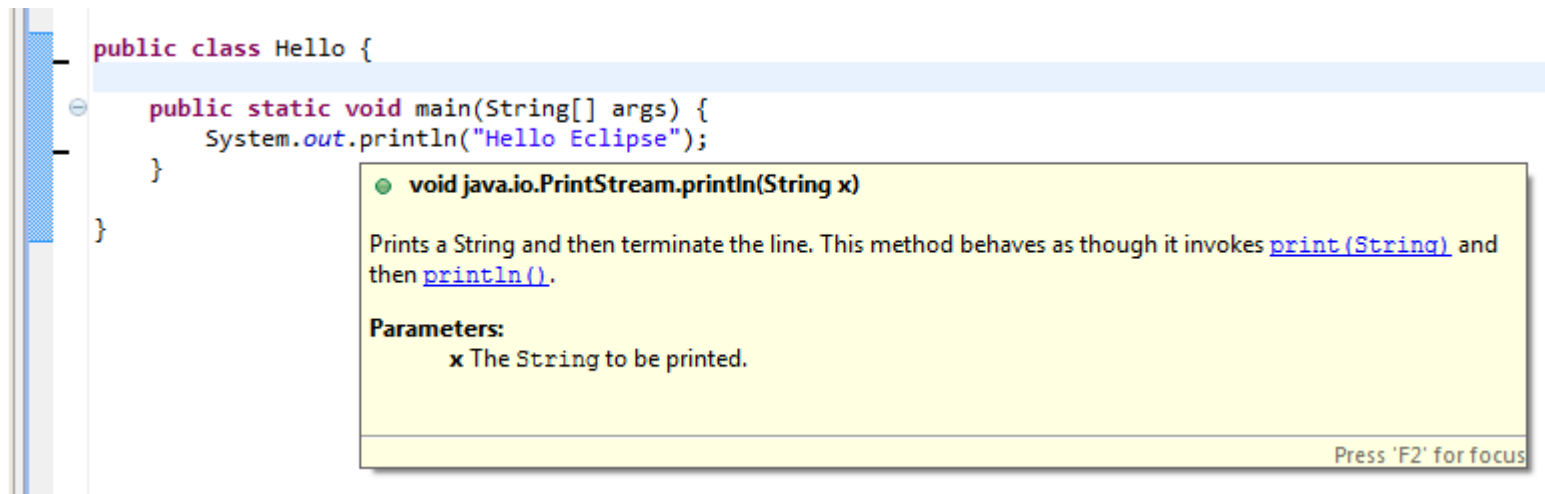
- Icon  oder:
- *Run As => Java Application* im Contextmenü von Hello.java

■ Hilfe beim Programmieren

- Code formatieren: CTRL+SHIFT+F
- Code Completion: automatisch oder explizit: CTRL+SPACE
- Refactoring: Umbenennen von Methoden/Klassen/Felder, inkl. anpassen aller Referenzen
- und vieles mehr ...

Dokumentation

- Komplette Java Dokumentation:
 - <http://docs.oracle.com/javase/7/docs/>
- Dokumentation aller Klassen die mit Java mitgeliefert werden:
 - <http://docs.oracle.com/javase/7/docs/api/>
- Tooltips in Eclipse:



■ Richtlinie für Java-Programmierstil

- Erhöht Lesbarkeit des Codes
- 80% der Softwarekosten durch Code-Wartung
- Code wird selten vom ursprünglichen Autor gewartet
- Erleichtert mir das Korrigieren 😊

■ „Code Conventions for the Java Programming Language“

<http://www.oracle.com/technetwork/java/codeconv-138413.html>

■ Automatisch in Eclipse : Source → Format

■ Source Formatter konfigurieren:

Window → Preferences → Java → Code Style → Formatter

■ Eclipse[built-in], Java Conventions[built in] oder benutzerdefiniert

Einige Code Conventions

- Paket-Namen werden klein geschrieben (z.B. `ch.ethz.inf.sd.math`)
- Klassen beginnen mit Grossbuchstabe (z.B. `Complex`)
- Java Dateien heissen gleich wie die (public) Klassen, die sie enthalten
Vorsicht: UNIX-oide Systeme unterscheiden Gross/Kleinschreibung
- Statische Konstanten werden gross geschrieben
- Felder/Methoden beginnen mit Kleinbuchstaben. Trennung von Worten durch Grossschreibung:

~~`get_first_element()`~~

`getFirstElement()` ✓

- Klammerung:

~~```
void foo()
{
 if (x>y)
 {
 }
 else
 {
 }
}
```~~

```
void foo() {
 if (x>y) {
 } else {
 }
}
```



- Code Blöcke Einrücken

# Ausgeben von Objekten

- Ausgabe in Java von Primitivtypen (int, float, double, usw.)

```
int a, b;
```

```
...
```

```
System.out.println(a+b*3.0);
```

↗  
Gibt Ergebnis auf  
der Konsole aus

- Ausgabe von Objekten durch Konvertierung in String

```
Foo foo = new Foo();
```

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

↗  
Ausgabe:  
Foo@82ba41  
(Klasse+Hashwert)

# Eine Klasse für Vektor-Operationen

- Encapsulation: Klasse “Vector” kapselt Vektor-Operationen in  $\mathbb{R}^3$
- Mögliche Operationen (nicht abschliessend):
  - Konstruktoren: Konstruktor aus drei Komponenten
  - $\mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3$  : Addition, Subtraktion
  - $\mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^3$  : Skalierung
  - $\mathbb{R}^3 \times \mathbb{R}^{3 \times 3} \rightarrow \mathbb{R}^3$  : Rotation
  - $\mathbb{R}^3 \rightarrow \mathbb{R}$  : Norm
  - Getter und Setter Methoden für die einzelnen Komponenten
  - Normalisieren
  - Ausgabe als String
- Quellcode online nach Übungsstunde

# Grundgerüst

```
public class Vector {
 private double x, y, z;

 public Vector(double x, double y, double z) {
 this.x = x; this.y = y; this.z = z;
 }
```

```
// ...
```

```
 public double getX() {
 return x;
 }
```

```
 public void setX(double x) {
 this.x = x;
 }
```

```
 public void add(Vector v) {
 x+=v.x; y+=v.y; z+=v.z;
 }
```

```
// ...
```

```
}
```

auch möglich

```
{
 public Vector add(Vector v) {
 return new Vector(x+v.x,
 y+v.y, z+v.z);
 }
}
```

## ■ Die Klasse `VectorTest` verwendet `Vector`

```
public class VectorTest {

 public static void main(String args[]) {

 Vector a = new Vector(3.0, 1.0, 5.0);
 Vector b = new Vector(1.0, 0.0, 0.0);

 a.add(b);
 System.out.println(a.getX());
 }
}
```