Java Crash Course

Grundlagen der Programmierung

Sönke Schmidt
(Soenke.Schmidt@fu-berlin.de)

Meine Webseite: http://www.soenke-berlin.de
Mentoring Webseite: www.mi.fu-berlin.de/stud/mentoring

Java – kann man das essen?

- objektorientierte Programmiersprache
  - Quellcode + Compiler -> Bytecode (maschinenverständlich)
  - Bytecode von der Java virtuell Maschine (JVM) interpretiert
- Java-Entwicklungswerkzeug (JDK) zum Erstellen
- Java-Laufzeitumgebung (JRE) zur Ausführung
  - JVM& Bibliotheken
- Plattformunabhängig
  - Durch Virtualisierung (siehe JVM)
  - Just-in-time (JIT) – Kompilierung
- Es ist kein Javascript und sollte damit nicht verwechselt werden

Java Virtual Machine

http://cdn.cs50.net/2010/fall/seminars/CrashCourseInJava/CrashCourseInJava.pdf

Wieso Java???

- Plattformunabhängig
  - Bsp.: Code des Studenten (Win) ist ausführbar beim Tutor (Linux) und beim Prof. (Mac)
- Viel einfacher als andere Programmiersprachen
  - Höhere Abstraktion
  - Viele Bibliotheken verfügbar (bsp. Für Protokolle wie HTTP)
  - Nicht direkt auf Hardware ausgeführt - Sandbox
- Keine Pointers! (Garbage Colection)
- Einfaches und schönes Error Handling
- Ähnlich zu C/C++
  - Aber NICHT identisch!!!
  - Ähnliche Syntax und ähnliche primitive Datentypen (bool ist hier kein int)

Tafel

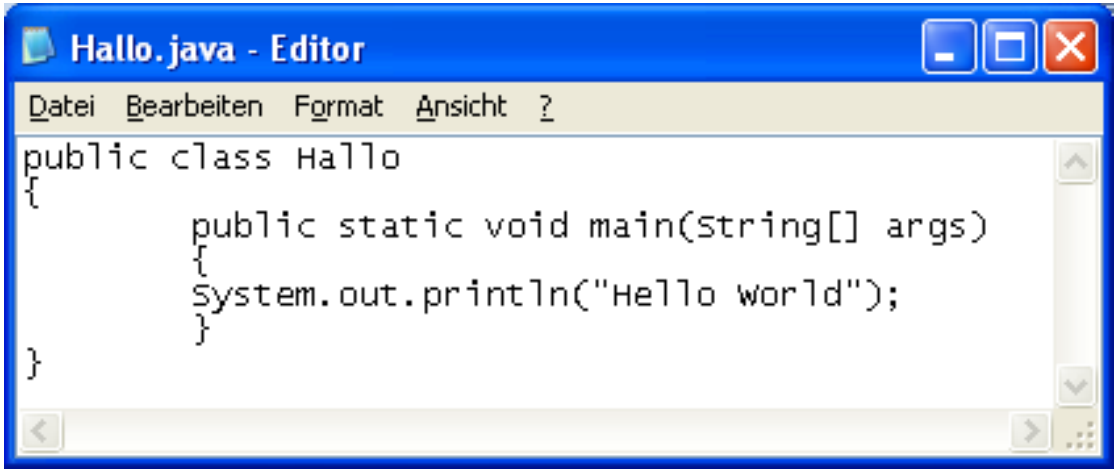
Kurzes Tafelprogramm

Unser erstes Programm (Hallo) – Hello World

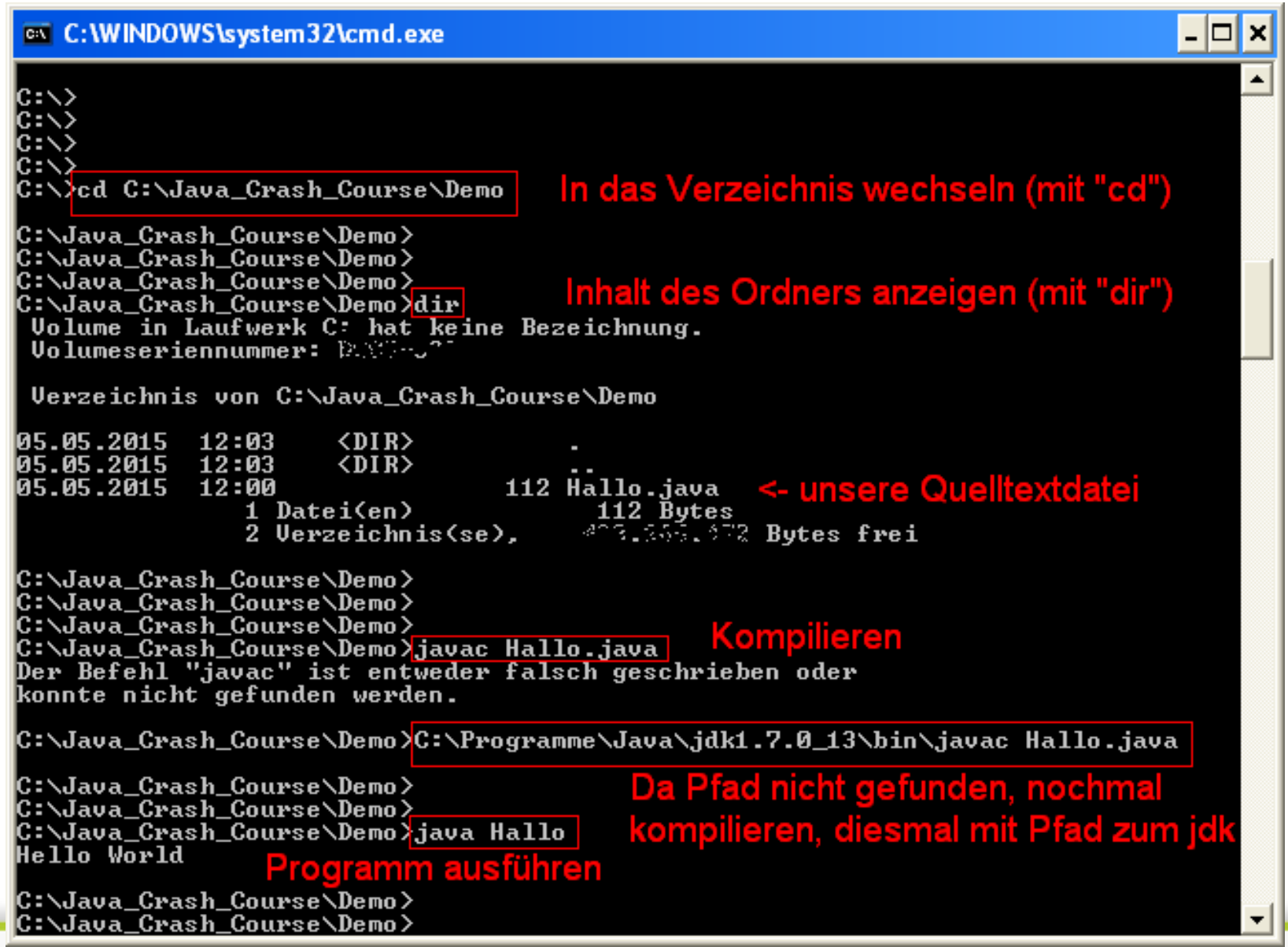
```
public class Hallo
{
        public static void main(String[] args)
        {
                System.out.println("Hello World");
        }
}
```

Kompilieren und ausführen

- Quellcode speichern in Hallo.java
- Javac Hallo.java
  - Es wird nun kompiliert
  - Es entsteht Hallo.class
  - *.class ist der Bytecode
- Ausführen mit: java Hallo



Via Kommandozeile



IDE

IDE = integrated development environment
= Integrierte Entwicklungsumgebung

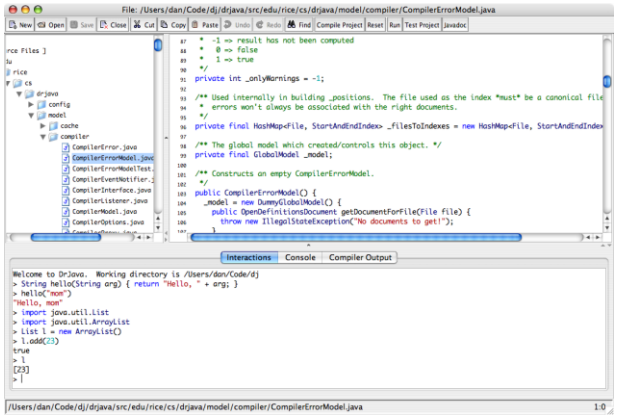
Einige bekannte sind:

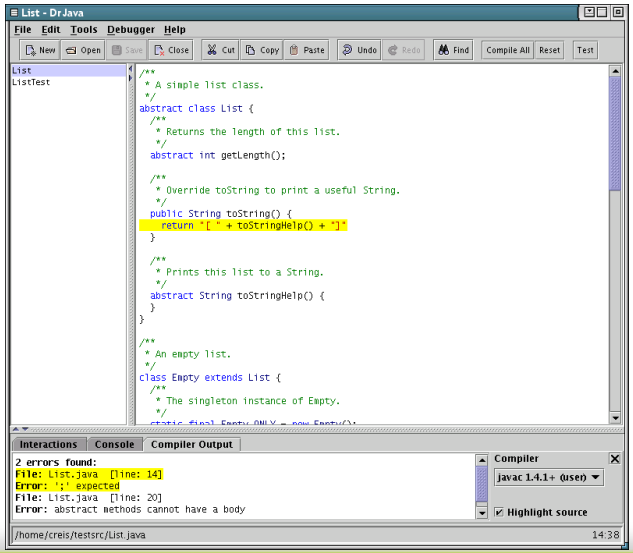
- Eclipse
  - https://de.wikipedia.org/wiki/Eclipse_(IDE)
- IntelliJ IDEA
  - https://de.wikipedia.org/wiki/IntelliJ_IDEA
- NetBeans DIE
  - https://de.wikipedia.org/wiki/NetBeans_IDE
- DrJava
  - https://en.wikipedia.org/wiki/DrJava
- Geany
  - https://de.wikipedia.org/wiki/Geany

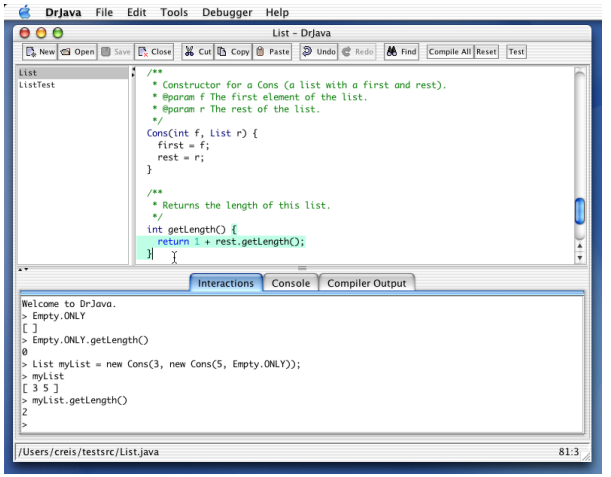
IDE - DrJava

www.drjava.org *bzw.* http://drjava.sourceforge.net/

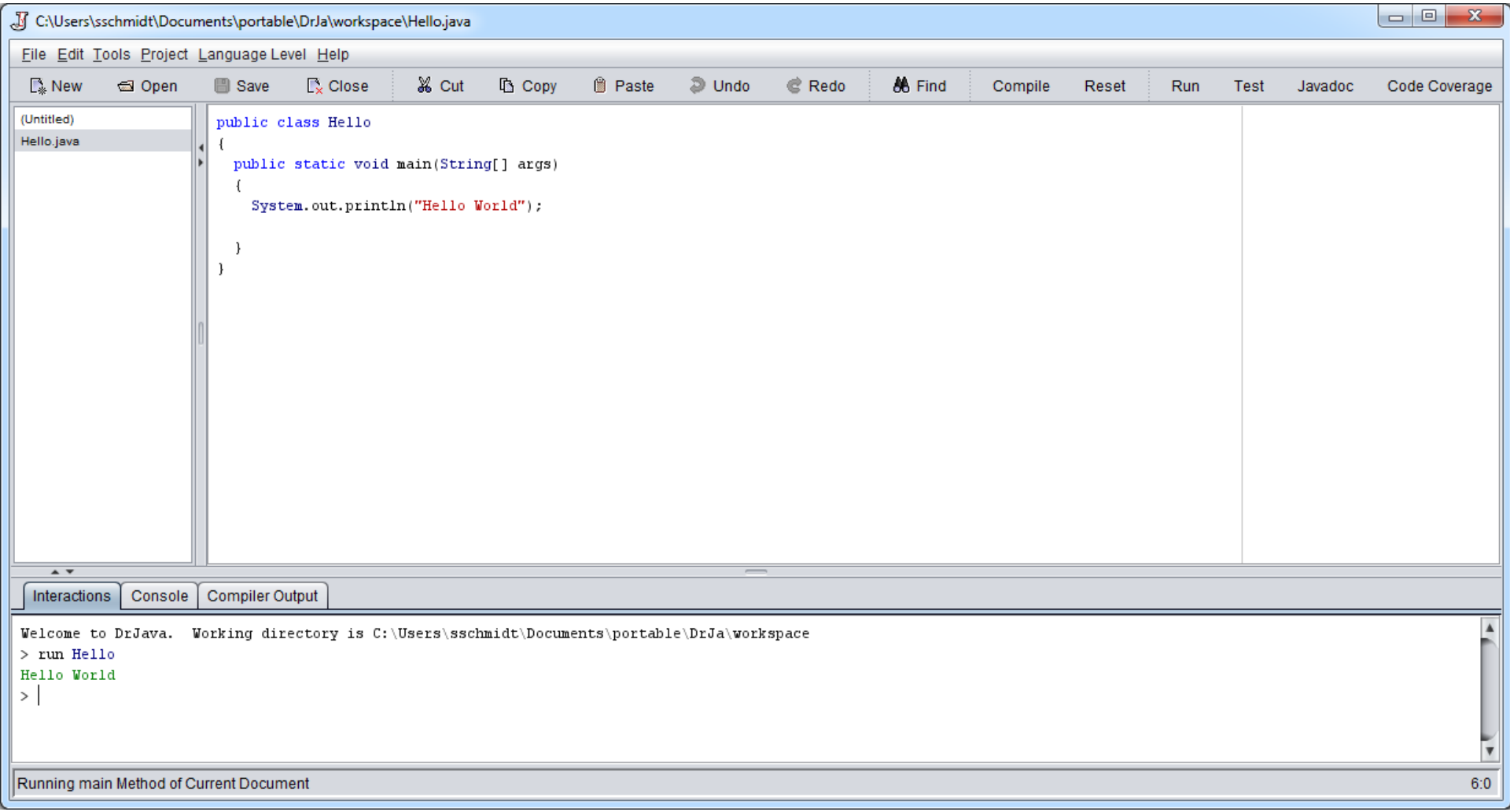
- Für Studenten und Anfänger designed, hat aber auch „advanced“ zeug
- Übersichtliches Design
- Open Source License
- Für Win, Linux, Mac und überall wo Java läuft



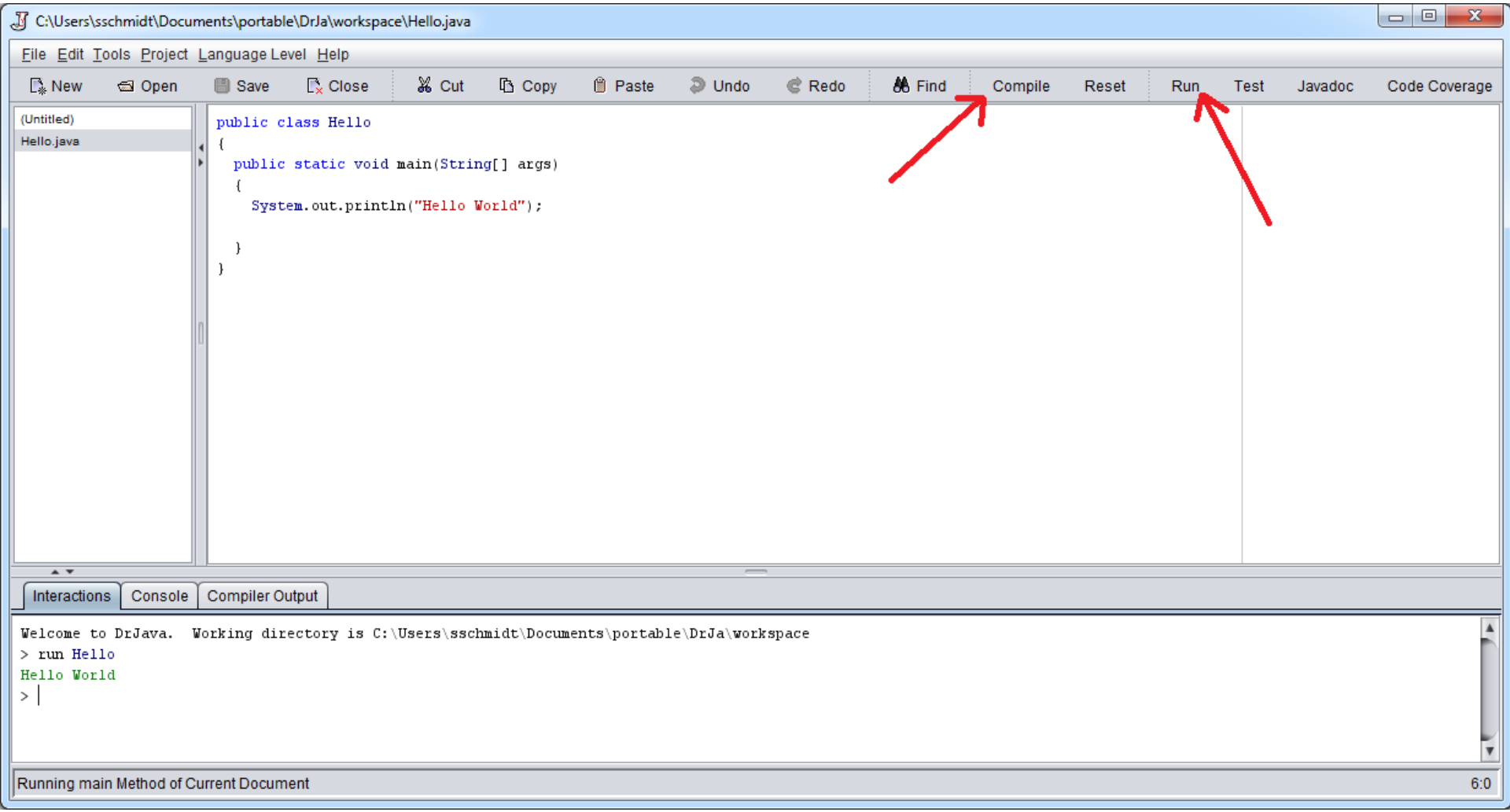




IDE - DrJava


C:\Users\sschmidt\Documents\portable\DrJa\workspace\Hello.java
File Edit Tools Project Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset Run Test Javadoc Code Coverage
(Untitled)
Hello.java
public class Hello
{
public static void main(String[] args)
{
System.out.println("Hello World");
}
}
Interactions Console Compiler Output
Welcome to DrJava. Working directory is C:\Users\sschmidt\Documents\portable\DrJa\workspace
> run Hello
Hello World
>
Running main Method of Current Document
6:0

IDE - DrJava


C:\Users\sschmidt\Documents\portable\DrJa\workspace\Hello.java
File Edit Tools Project Language Level Help
New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset Run Test Javadoc Code Coverage
(Untitled)
Hello.java
public class Hello
{
public static void main(String[] args)
{
System.out.println("Hello World");
}
}
Interactions Console Compiler Output
Welcome to DrJava. Working directory is C:\Users\sschmidt\Documents\portable\DrJa\workspace
> run Hello
Hello World
>
Running main Method of Current Document
6:0

Java - Was ist wichtig? Wiederholung von Tafel

- Alles muss in einer Klasse sein
- Wir brauchen eine „main()“ zum starten
- Ein Ausdruck endet mit einem Semikolon („ ; “)
- Variablen müssen (mit Datentyp) deklariert werden (vor dem benutzen!)
  - Format: Typ Variablen-Name
    - Bsp.: int quotient;
      float taschengeld
- Kommentare mit **//** oder im Block von **/*** über Zeilen bis zum Ende ***/**
- Blockstruktur anders
  - Anweisungsblöcke sind durch geschweifte Klammern { } gekennzeichnet
- Name Conventions ; Namens Zusammenhänge mit Großbuchstaben verknüpft oder Unterstrich
  - Variablen beginnen klein ; Bsp.: maxSpeed, zeitDerDauerVonX
  - Klassen beginnen groß; Bsp.: MyClass, MeinErstesProgramm, Hallo (s.o.)

Primitive Datentypen in Java

Datentyp	Größe[1]	Wrapper-Klasse	Wertebereich	Beschreibung
boolean	JVM-Spezifisch	java.lang.Boolean	true / false	Boolescher Wahrheitswert
char	16 bit	java.lang.Character	0 ... 65.535 (z. B. 'A')	Unicode-Zeichen (UTF-16)
byte	8 bit	java.lang.Byte	-128 ... 127	Zweierkomplement-Wert
short	16 bit	java.lang.Short	-32.768 ... 32.767	Zweierkomplement-Wert
int	32 bit	java.lang.Integer	-2.147.483.648 ... 2.147.483.647	Zweierkomplement-Wert
long	64 bit	java.lang.Long	-9.223.372.036.854.775.808 ... 9.223.372.036.854.775.807	Zweierkomplement-Wert
float	32 bit	java.lang.Float	+/-1,4E-45 ... +/-3,4E+38	Gleitkommazahl (IEEE 754)
double	64 bit	java.lang.Double	+/-4,9E-324 ... +/-1,7E+308	Gleitkommazahl doppelter Genauigkeit (IEEE 754)

http://de.wikibooks.org/wiki/Java_Standard:_Primitive_Datentypen

Operatoren

- **Zuweisungs Operatoren:** =, +=, -=, *=, /=, %=
  - Bsp.: meinGeldAufmBankkonto += 10;
    entspricht:´ meinGeldAufmBankkonto = meinGeldAufmBankkonto + 10;

- **Numerische Operatoren:** +, -, *, /, %, ++, --

- **Relationale Operatoren:** ==, !=, <, >, <=, >=
  - Bsp.: if (meinGeldAufmBankkonto <= 0) System.*out.print(*"P*LEITE*");*

- **Boolische Operatoren:** &&, ||, !
  - Bsp.: if ((meinGeldAufmBankkonto <= 0) && (geld_im_Portmonai <=0))
    System.*out.print(*"P*LEITE*");*

- **Bitweise Operatoren :** &, |, ^, ~, <<, >>

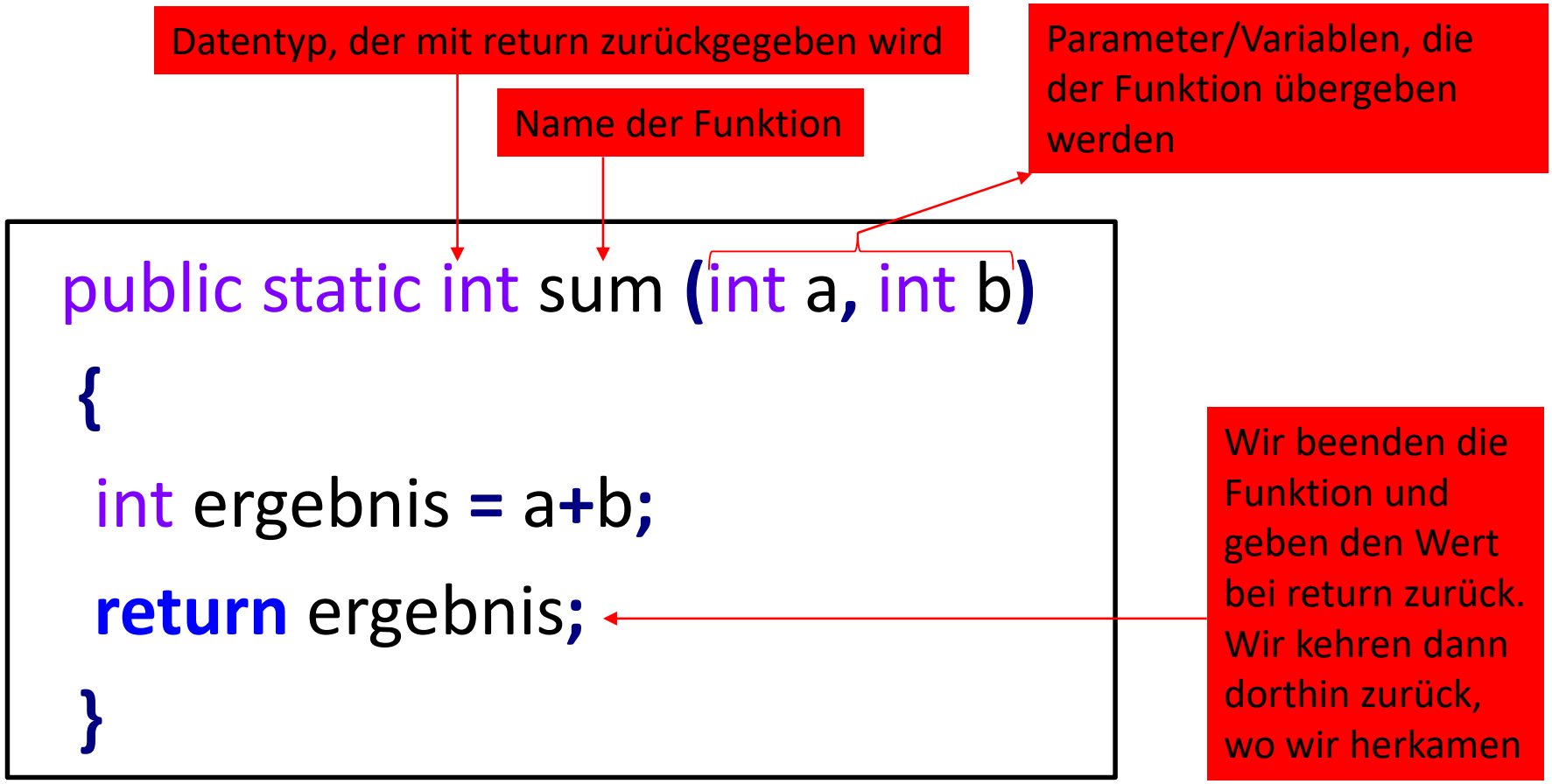
Tafel

Kurzes Tafelprogramm

Funktionen

```
public static int sum (int a, int b)
 {
  int ergebnis = a+b;
  return ergebnis;
 }
```

Funktionen


Datentyp, der mit return zurückgegeben wird
Name der Funktion
Parameter/Variablen, die der Funktion übergeben werden
public static int sum (int a, int b)
{
int ergebnis = a+b;
return ergebnis;
}
Wir beenden die Funktion und geben den Wert bei return zurück. Wir kehren dann dorthin zurück, wo wir herkamen

Funktionen aufrufen

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println( sum(3,4) );  //Funktion aufrufen und direkt ausgeben
        int erg = sum(7,3); //Das Ergebnis aus der Funktion sum in die Variable erg speichern
        System.out.println( erg );  //Ergebnis ausgeben
    }

    //Die Funktion sum berechnet die Summe zweier natürlicher Zahlen
    public static int sum (int a, int b)
    {
        int ergebnis = a+b;
        return ergebnis;
    }

}
```

Funktionen aufrufen

```
public class Hello
{
    public static void main(String[] args)
    {
        System.out.println( sum(3,4) );  //Funktion aufrufen und direkt ausgeben
        int erg = sum(7,3); //Das Ergebnis aus der Funktion sum in die Variable erg speichern
        System.out.println( erg );  //Ergebnis ausgeben
    }

    //Die Funktion sum berechnet die Summe zweier natürlicher Zahlen
    public static int sum (int a, int b)
    {
        int ergebnis = a+b;
        return ergebnis;
    }

}
```

Auf der Konsole erscheint als Ausgabe:
7
10

Strings

- Initialisierung mit:

```
String name = "Peter ";
String satz  = "Es war einmal vor einer langen Zeit…";
```

- String Konkatenation (zusammenfügen) mit +
  - Bsp.: satz + name;

    //wird zu: Es war einmal vor einer langen Zeit…Peter
  - Oder: name + " hoer zu. " + satz;

    //wird zu: Peter hoer zu. Es war einmal vor einer langen Zeit…

String Methoden

- String länge überprüfen mit .length()

  Bsp.: String name = "Peter ";

  int zaehler = name.length(); //zaehler == 5

- Strings miteinander vergleichen
  - NICHT mit ==
  - Benötigt wird für Strings: .equals

  Bsp.: String name = "Peter";

  String name2 = "Pe";

  name2 = name2 + "ter";

  if (name.equals(name2)) System.out.println("Beide Namen identisch");

- Alle Zeichen im String zu klein Buchstaben machen, mit .toLowerCase()

  Bsp.: String text= "Dies ist ein Test!";

  Nach text.toLowerCase(); wird daraus: "dies ist ein test!"

String Methoden

- Äquivalent, um alles zu Großbuchstaben umzuwandeln: .toUpperCase

  Bsp.: String text= "Dies ist ein Test!";

  mit text.toUpperCase();

  wird daraus: DIES IST EIN TEST!

- Trim – Entfernt die Leertasten sowie Tabulatoren und Zeilenumbruche, die am Anfang und Ende von einem String stehen

  Bsp.: String test = " Platz vorne und hinten ";

  test.trim();

  //Daraus wird nun „Platz vorne und hinten“

  //Alle whitespaces wurden entfernt

String Methoden

- Zeichen aus String holen mit .charAt(Position)

  Bsp.:

```
String test = "Hallo";
char neu;
neu = test.charAt(0);
//Variable neu hat nun das ‚H‘
```

- Teilstringe nehmen mit substring(beginIndex, endIndex)

  Bsp.:

```
String satz = "Es war einmal vor einer langen Zeit…";
System.out.println("Teilstring: "+ satz.substring (7,13) );
//Ausgabe: Teilstring: einmal
```

String - Escapen

- Um in einer Zeichenkette / String / Ausgabe spezielle Zeichen benutzen zu können, muss man sie vorher escapen:

\" Doppel Anführungsstriche
\' Einfache Anführungsstrich
\\ Backslash
\n New line (Geh zum Beginn der nächsten Linie)
\r Carriage return (Geh zum Beginn der aktuellen Linie)
\t Tabulator (macht white spaces bis zum nächsten Tab stop)

Exceptions

```
try {
        //Code der eine Exception werfen könnte
} catch (ExceptionType e)
{
        //Code um mit der Exception umzugehen
} //Mehrere catches mit unterschiedlichen ExceptionTypes möglich
```

Beispiel:

String zum Integer umwandeln. Problem: String könnte gar keine natürliche Zahl sein! Abfangen im Notfall:

```
String zahlString = "42";   //Zum Testen der Exception den String ändern (bsp. ändern zu "42asd";   )
int zahl = 0;
try{
    zahl = Integer.parseInt(zahlString);
    System.out.println("Casten vom String "+zahlString +", der zum integer wird: "+zahl);
}catch(Exception e){
     //... falls zahlString keine Zahl ist – also die Umwandlung nicht funktioniert
   System.out.println("Es ist ein Problem aufgetreten! "+
                "Der String scheint keine natürliche Zahl zu sein und konnte nicht umgewandelt werden! \n"+
                "Die Fehlermeldung lautet: "+e);
}
```

Skript

Jetzt dürft ihr etwas rumprobieren!

Ihr bekommt gleich den ersten Teil vom Skript

Kontroll Strukturen

- Entscheidungen: if, if-else, switch
- Schleifen: for, while, do-while

- Kontrollstrukturen werten Boolische Ausdrücke aus
  - Bsp.:
    - **if (** x **==** 0 **)** //x ist eine integer Variable
    - **if (**Bedingung**)** //Bedingung ist eine boolesche Variable
    - **if (true)**
    - **if (**x **>** 3**)**
    - **while(**geld **>** 0**)**

Kontroll Strukturen – if -else

```
if (Bedingung)
{
   Anweisung
}
else
{
   Anweisung
}
```

Kontroll Strukturen – While

```
while(Bedingung)
{
   Anweisung
}
```

Kontroll Strukturen – While Beispiel

```
while(geld > 0)
{
   SpendeGeld(10);
   geld = geld – 10;
}
```

Kontroll Strukturen – For Schleife

for (Initialisierung; Boolescher Test bzw Bedingung; Modifikation)

{

Anweisung

}

Bsp.:

```
for ( int i = 1; i < 11; i++)        // i++  entspricht i = i +1
{
  System.out.println("Nr. " + i);
}
```

Kontroll Strukturen – For Schleife (Rückwärts)

Was gibt das folgende aus?

```
for ( int i = 10; i > 0; i -= 2)  // i -= 2  entspricht i = i - 2
{
  System.out.println( i );
}
```

Kontroll Strukturen – do-While

```
do {
Anweisung
} while (Bedingung);
```

- Der Körper der do-while Schleife wird immer zum Schluss ausgeführt…
  - Bedeutet, dass alles was hinter dem „do“ kommt, mindestens einmal ausgeführt wird, erst danach wird die Bedingung überprüft und ggf erneut ausgeführt.

Break & continue

- Break und continue können in while, do-while und for Schleifen verwendet werden, um den Rest des durchlaufes zu überspringen

- Break: Bricht die Schleife ab
- Continue: Führt den nächsten Schleifen durchlauf aus

Break Beispiel

Example break in a for Loop

```
…
int i ;
for (i = 1; i < 10; i = i + 1) {
 if (i == 5) {
     break;
 }
 System.out.println(i);
}
System.out.println("\nBroke out of loop at i = " + i);
```

- **OUTPUT:**
- **1 2 3 4**
- **Broke out of loop at i = 5.**

http://cs.brynmawr.edu/Courses/cs206/spring2012/slides/01_IntroToJavaProgramming.pdf

Continue Beispiel

Example continue in a for Loop

```
…
int i;
for (i = 1; i < 10; i = i + 1) {
 if (i == 5) {
     continue;
 }
 System.out.println(i);
}
System.out.println("Done");
```

OUTPUT:

1 2 3 4 6 7 8 9

Done.

http://cs.brynmawr.edu/Courses/cs206/spring2012/slides/01_IntroToJavaProgramming.pdf

Skript

Jetzt dürft ihr etwas rumprobieren!

Ihr bekommt gleich den zweiten Teil vom Skript

Switch

```
switch (Ausdruck)  //meist integer Ausdruck
{
        case Konstante1 : Anweisung(en);
                          break;
        case Konstante2:  Anweisung(en);
                          break;
        …

        case KonstanteX:  Anweisung(en);
                          break;
        default:          Anweisung(en);
                          break;
}
```

Switch Beispiel

```
//[..]
Int day = 2;
//[..]
switch (day){
        case 1:     System.out.println("Montag");
                    break;
        case 2:     System.out.println("Dienstag");
                    break;
        //[..]
        case 6:     System.out.println("Samstag");
                    break;
        case 0:
        case 7:     System.out.println("Sonntag");
                    break;
        default:    System.out.println("Error – kein gültiger Tag");
                    break;
}
```

Switch Beispiel

```
//[..]
Int day = 2;
//[..]
switch (day){
        case 1:     System.out.println("Montag");
                    break;
        case 2:     System.out.println("Dienstag");
                    break;
        //[..]
        case 6:     System.out.println("Samstag");
                    break;
        case 0:
        case 7:     System.out.println("Sonntag");
                    break;
        default:    System.out.println("Error – kein gültiger Tag");
                    break;
}
```

Zu beachten ist hier Tag 0 und Tag 7.
Da es identisch ist, wird kein break bei 0 mit eingebaut.
Somit fällt es im case 0 automatisch in case 7 rein!
Bedeutet ein Aufruf von case 0 führt zum darunterliegenden Aufruf von case 7.

Arrays

- Index startet mit 0 und ist fix (größe nicht änderbar)

- Deklarieren:

```
double[] score = new double[5];
```

Oder in zwei Ausdrücken:

```
double[] score;
score = new double[5];
```

- Arrays mit Initialisierung erzeugen:

```
int[] messungen = {132, 22, 73, 98}
```

- Länge des Arrays:

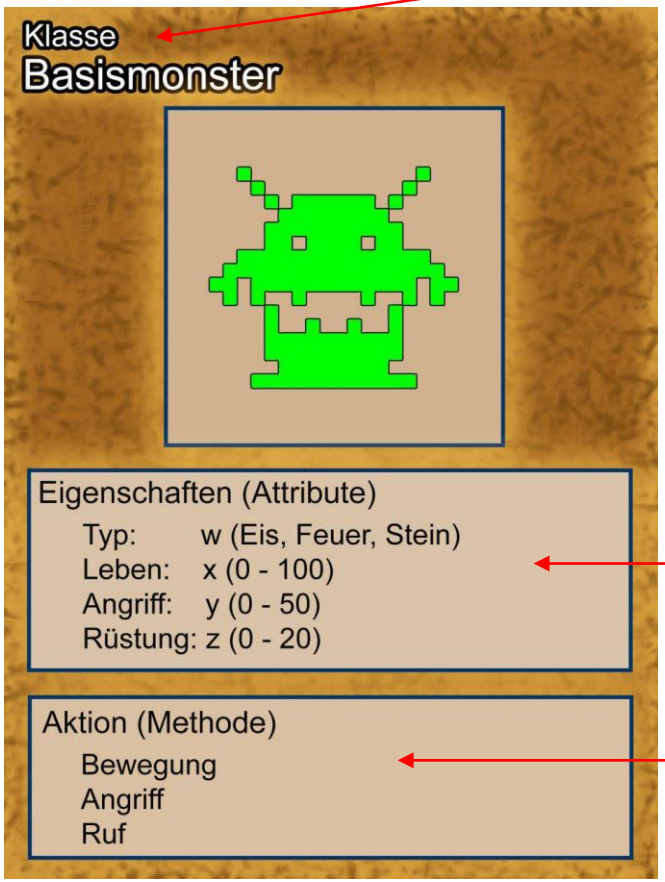
```
int zahl = score.length;  //zahl == 5
```

Pause

Erstmal eine Pause…

Objekte aus Klassen erzeugen

Objekte aus Klassen erzeugen

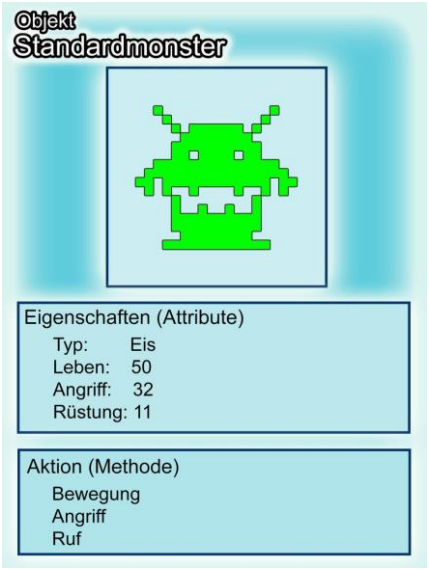


Unsere Klasse, die wir uns einmal programmieren

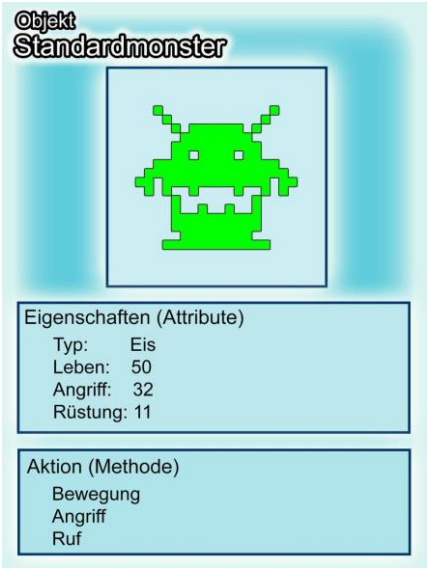
Können von einem Datentypen sein. Beispiel Leben könnte so aussehen: int health;

Können einfach Funktionen sein, Beispiel wäre: public static void walkAround()

Objekte aus Klassen erzeugen



Objekte aus Klassen erzeugen



Wir „ziehen“ uns von der Klasse die Objekte ab.
Zu beachten ist hier, dass wir Die Eigenschaften (Attribute) der Monster mit unterschiedlichen Werten gefüllt haben!
Die Funktionen (bzw. die Aktionen) der Monster, bleiben gleich.
Wir können von der Klasse nun beliebig viele Monster Objekte erstellen.

Objekte aus Klassen erzeugen

Ihr findet ein Beispiel zu Objekten, wie man damit in Java umgeht und sie benutzen kann, im Handout
(auf meiner Webseite: http://www.soenke-berlin.de)
neben dieser Präsentation.

Das Beispiel behandelt Flugzeuge und gibt euch zudem einen Einblick in die IDE von IntelliJ.

Vielen Dank für Ihre Aufmerksamkeit.

Haben Sie

Fragen?