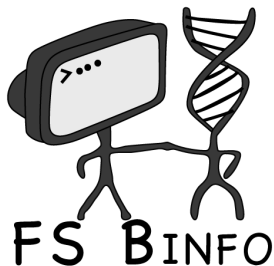


# Einführung in Java und Eclipse



Simon Dirmeier  
simon.dirmeier@campus.lmu.de

Fachschaft Bioinformatik

22. November 2011

# Zeitplan

---

- 25.10 Einführung in Linux und die Shell
- 08.11 Arbeiten mit Dateien auf der Shell
- 15.11 Shell 3 und Editor: vim
- 22.11 **Einführung Java und Eclipse**
- 29.11 Konditionen, Arrays und Schleifen
- 06.12 Rekursion und Iteration
- 13.12 File-I/O
- 20.12 Weihnachtsübung
- 10.01 Debuggen mit Eclipse
- 17.01 Objektorientierte Programmierung
- 24.01 Vererbung und Tipps
- 31.01 Quiz, Fragen und Evaluation

Webseite: <http://www.bioinformatik-muenchen.com/bioinfocom/informatik-tutorium>



# Einleitung : Was ist Java?

---



- Objektorientierte Programmiersprache
- Ende der 80er von Sun zur Steuerung von Haushaltsgeräten entwickelt.
- Wegen "Plattformunabhängigkeit" schnelle Verbreitung im heranwachsenden Netz
- Java-Programme kommunizieren mit JVM.
- JVM übersetzt Javacode in Bytecode und kommuniziert mit OS, CPU. . .
- "Benutzerfreundlich" dank Typüberprüfung, Garbagecollector, etc. . .

# Praxiseinstieg

---



- Wir schreiben unser erstes(?) Java-Programm unter Linux!
- Unser erstes Programm: "Hello World"
- Dafür brauchen wir?



# Praxiseinstieg

---



- Texteditor (zB kate/gedit/nano/vim unter Linux)
- Java Developmentkit (JDK) bereits vorhanden
- Konsole (diese solltet ihr inzwischen "einigermaßen" beherrschen)



# Praxiseinstieg

---

Der Javacode könnte so aussehen:

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

# Praxiseinstieg

---



- Konsole : `vi/vim HelloWorld.java` (`vi/vim` erstellt `HelloWorld.java` und öffnet die Datei)
- Java-Code in Datei
- Schließen und kompilieren : `javac HelloWorld.java`
- Aufruf : `java HelloWorld`
- **Achtung: Main-Klasse muss wie Datei heißen**



# Praxiseinstieg

---



## Übung:

Schreibe ein lauffähiges Java-Programm mit vi/vim/nano, das einen beliebigen Text auf die Konsole ausgibt



# Reflexion

---

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

- Programm = Daten + Befehle
- Main-Methode zur Steuerung:  
**public static void main(String[] args)**
- Alles innerhalb einer Klasse:  
**public class CLASS**
- Viele un-intuitive Zeichen = hohes Fehlerrisiko

## Primitive Datentypen

Typname	Länge	Wertebereich
<code>boolean</code>	1 Byte	Wahrheitswerte <code>{true,false}</code>
<code>char</code>	2 Byte	Alle Unicode-Zeichen
<code>byte</code>	1 Byte	Ganze Zahlen von $-2^7$ bis $2^7 - 1$
<code>short</code>	2 Byte	Ganze Zahlen von $-2^{15}$ bis $2^{15} - 1$
<code>int</code>	4 Byte	Ganze Zahlen von $-2^{31}$ bis $2^{31} - 1$
<code>long</code>	8 Byte	Ganze Zahlen von $-2^{63}$ bis $2^{63} - 1$
<code>float</code>	4 Byte	Gleitkommazahlen (einfache Genauigkeit)
<code>double</code>	8 Byte	Gleitkommazahlen (doppelte Genauigkeit)

z.B. :

```
boolean b = true;
```

```
int i = 1; long l = 11L;
```

```
float f = 2.0f; double d = 2.0;
```

# Primitive Datentypen

---

Der Javacode könnte so aussehen:

```
public class Datentyp {  
    public static void main (String[] args) {  
  
        int i = 1;  
        boolean b = true;  
        char c = 'c';  
        float f = 0.1f;  
        double d = 2.0;  
        long l = 10L;  
        // usw  
  
        System.out.println(i + " " + b + " " + f);  
    }  
}
```

# Strings

---



- Strings sind Behälter, die Texte speichern
- String wird folgendermaßen erzeugt :  
**String meinString = " ein beliebiger Text!";**
- //Schreibt auf Stdout , was in 'meinString' steht:  
**System.out.println(meinString);**



# Strings

---

Der Javacode könnte so aussehen:

```
public class String {  
  
    public static void main(String[] args) {  
        String s = " i like java ";  
        System.out.println( s );  
    }  
}
```

# Reflexion

---



- String ist ein Datentyp für beliebige Zeichen
- Erste Wertzuweisung heißt Initialisierung
- Variablenname  $\neq$  Wert der Variablen  
**int i = 0**
- Kommentare im Code mit //



## Arbeiten mit IDE : Eclipse

---



- Eclipse ist eine Entwicklungsumgebung, die versucht dem Programmierer Denk-,Layout-, und Tipparbeit zu erleichtern
- Weitere IDEs (Netbeans, JBuilder,....)
- Das könnte dann so aussehen :



# Arbeiten mit IDE : Eclipse

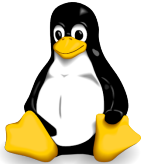
The screenshot displays the Eclipse IDE interface with the following components:

- Package Explorer:** Shows the project structure for 'org.eclipse.ui.examples.javaedit'. The 'JavaDocumentSetupParticipant.java' file is selected.
- Source Editor:** Displays the code for 'ContentProvider' which implements 'ITreeContentProvider'. The code includes:

```
protected class ContentProvider implements ITreeContentProvider {  
  
    protected final static String SEGMENTS= "___java_segments"; //NON-NLS  
    protected IPositionUpdater fPositionUpdater= new DefaultPositionUpdat  
    protected List fContent= new ArrayList(10);  
  
    protected void parse(IDocument document) {  
  
        int lines= document.getNumberOfLines();  
        int increment= Math.max(Math.round(lines / 10), 10);  
  
        for (int line= 0; line < lines; line += increment) {  
  
            int length= increment;  
            if (line + increment > lines)  
                length= lines - line;  
  
            try {  
  
                int offset= document.getLineOffset(line);  
                int end= document.getLineOffset(line + length);  
                length= end - offset;  
                Position p= new Position(offset, length);  
                document.addPosition(SEGMENTS, p);  
                fContent.add(new Segment (MessageFormat.format(JavaEditorM  
  
            } catch (BadPositionCategoryException x) {  
            } catch (BadLocationException x) {
```
- Outline:** Shows a tree view of the code structure, including 'org.eclipse.ui.examples', 'JavaContentOutlinePag', 'Segment', 'ContentProvider', and 'fInput: Object'.
- Console:** Shows 'SVN' as the current output.
- Bottom Status Bar:** Displays 'Writable', 'Smart Insert', and '2:1'.

# Arbeiten mit IDE : Eclipse

---



## Übung:

- Initialisiere 5 Variablen von verschiedenem Datentyp und gib sie unter Eclipse aus
- Vertausche den Wert zweier Variablen vom gleichen Typ
- Was passiert, wenn man :
  1. Int und Double addiert?
  2. Double und Int addiert?
  3. String und Int addiert?



# If-Statement und Bools

---



- Das if-Statement kann man sich als eine Verzweigung oder "Pattermatching" vorstellen
- Syntax : if (Bedingung(wahr)) {Anweisung}

```
if (2+3==5) {  
    System.out.println("2 + 3 ergibt 5");  
}  
else {  
    System.out.println("2 + 3 ergibt nicht 5");  
}
```

# If-Clause

---

Der Javacode könnte so aussehen:

```
public class IF{  
    public static void main(String[] args) {  
        int low    = 1;  
        int high   = 10;  
        if ( low < high ) {  
            System.out.println( low + " ist kleiner als " + high);  
        }  
        else if ( low == high ) {  
            System.out.println( low + " ist so groß wie " + high);  
        }  
        else {  
            System.out.println( low + " ist größer als " + high);  
        }  
    }  
}
```

# If-Statement und Bools

---



- Man kann auch mehrere if-Statements verschachteln

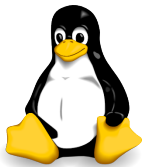
- Syntax:

```
if (Bedingung1) {  
    if (Bedingung2) {  
        Anweisung;  
    }  
    else {  
        Anweisung;  
    }  
}
```



# If-Statement und Bools

---



- "else" fängt alternative Fälle ab
- Bedingung ist vom Typ boolean, also entweder true or false
- Mit "!" kann eine Aussage negiert werden (!true == false)
- Wir vergleichen Aussagen mit "==" ("=" ist eine Zuweisung)
- Mit "&&" wird das logische "und"; mit "||" das logische "oder" dargestellt

# If-Statement und Bools

---



## Übung:

Überprüfe mit if-Statements und den Zeichen

$<$  ,  $>$  ,  $<=$  ,  $>=$

welche von 3 Zahlen die größte/kleinste ist und gib sie in der richtigen Reihenfolge aus



# For-Schleife

---



- Eine Schleife dient dazu, eine Anweisung zu wiederholen, solange bis eine bestimmte Bedingung (nicht mehr) erfüllt ist
- Syntax :  

```
for (int i=0 ; i<10 ; i++) {  
System.out.println("I ist jetzt : " + i);  
}
```



# For-Schleife

---

Der Javacode könnte so aussehen:

```
public class FOR{  
  
    public static void main(String[] args) {  
  
        for (int i = 0; i <= 10; i++) {  
            System.out.println("i ist jetzt " + i);  
        }  
    }  
}
```



# For-Schleife

---



## Übung:

Schreibe eine for-Schleife, die dir alle Zahlen von 1 bis 1000 addiert. Gib die Zahl dann aus (Zum Überprüfen: 500500)



# Tipp

---



- Die Dokumentation zu Java ist oft einen Blick wert, wenn man ungefähr weiß, was man will  
<http://java.sun.com/j2se/1.5.0/docs/api/>
- Auch mit Google kann nach bestimmten Begriffen gesucht werden



# Übungen für Daheim

---



- Erzeuge eine Funktion zur Multiplikation zweier Zahlen ohne die Verwendung von  $*$  und  $/$
- Berechne die Distanz zweier Punkte im Koordinatensystem
- ...

