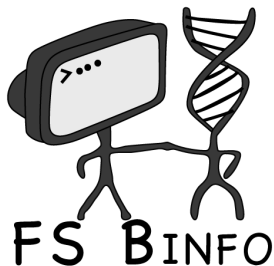


# Einführung in Java und Eclipse



Florian Seifert 2009 und  
Simon Dirmeier 2010

Fachschaft Bioinformatik

16. November 2010

# Zeitplan

---

- 26.10 Einführung in Linux und die Shell
  - 02.11 Shell : Dateimanipulation und weitere Befehle
  - 09.11 Kommandozeileneditor : vi
  - 16.11 **Einführung Java und Eclipse**
  - 23.11 Arrays und Schleifen
  - 30.11 Objektorientierte Programmierung
  - 07.12 Rekursion und Iteration
  - 14.12 File-I/O und Wiederholung
  - 21.12 Weihnachtsaufgabe
- 
- 11.01 Debuggen mit Eclipse
  - 18.01 Vererbung, Tipps und Fragen
  - 25.01 Wiederholung

Webseite: [http://www.bioinformatik-muenchen.com/  
bioinfocom/informatik-tutorium](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!
- Dafür brauchen wir?
- Eine Idee, was für ein Programm wir wollen :  
"Hello World"



# Praxiseinstieg

---



- Texteditor (zB Kate unter Linux)
- Java Developmentkit (JDK) bereits vorhanden
- Konsole



# Praxiseinstieg

---



- Der Javacode könnte so aussehen :
- ```
public class Test {  
    public static void main(String[] args) {  
        System.out.println (" Hello, world!" );  
    }  
}
```
- Wo gebe ich meinen Code nun ein?



# Praxiseinstieg

---



- Konsole : `kate meineDatei.java`
- Ist der Code fertig, wird die Datei kompiliert :  
`javac meineDatei.java`
- Sobald sie kompiliert ist, kann sie aufgerufen werden : `java meineDatei`



# Praxiseinstieg

---



- **Übung** : Schreibe ein lauffähiges Java-Programm, das einen beliebigen Text auf die Konsole ausgibt



# Reflexion

---



- Programm = Daten + Befehle
- Main-Methode zur Steuerung der weiteren Kommandos
- Alles innerhalb einer Klasse
- 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 x = 1; long l = 11L;
```

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

# Strings

---



- Strings sind Behälter, die Texte speichern können
- Ein String wird folgendermaßen erzeugt :  
`String meinString = "Hello, World";`
- `//` Schreibt auf die Ausgabe , was in `meinString` steht  
`System.out.println(meinString);`



# Reflexion

---

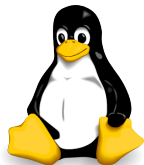


- String ist ein Datentyp für beliebige Zeichen
- Erste Wertzuweisung heißt Initialisierung
- Man unterscheidet zwischen Variablenname und Wert der Variablen
- Kommentare im Code mit //



## 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. The main editor window shows the source code for `ContentProvider` in `JavaContentOutlinePage.java`. The code is as follows:

```
protected class ContentProvider implements ITreeContentProvider {  
  
    protected final static String SEGMENTS= "___java_segments"; //SNON-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) {  
  
        }  
    }  
}
```

The Package Explorer on the left shows the project structure, with `JavaContentOutlinePage.java` selected. The Outline view on the right shows the class hierarchy, including `ContentProvider` and its methods. The Console view at the bottom is empty, showing the SVN status.

# 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-Statement und Bools

---



- Man kann auch mehrere if-Statements verschachteln

- Syntax:

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



# If-Statement und Bools

---



- "else" dient dazu alle nicht explizit abgefragten Fälle abzufangen
- 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



# Die 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);  
}
```



# Tipp

---

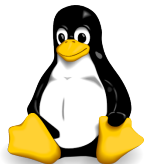


- Die Dokumentation zu Java ist oft einen Blick wert, wenn man ungefähr weiß, was man will  
<http://java.sun.com/j2se/1.5.0/docs/api/>
- Auch mit Google kann nach bestimmten Begriffen gesucht werden



# Übungen für Daheim

---



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

