

Lehrstuhl Bioinformatik • Konstantin Pelz

# Erste Java-Programme

(Eclipse und primitive Datentypen)

## Tutorium Bioinformatik

(WS 18/19)

Konstantin: [Konstantin.pelz@campus.lmu.de](mailto:Konstantin.pelz@campus.lmu.de)

Homepage: <https://bioinformatik-muenchen.com/studium/propaedeutikum-programmierung-in-der-bioinformatik/>

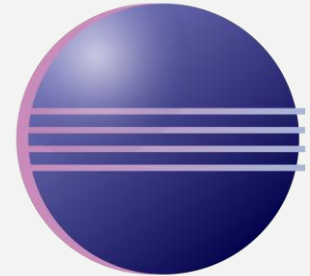




# Eclipse

**IDE: integrated development environment**

- Texteditor
- Compiler bzw. Interpreter
- Debugger
- Quelltextformatierung

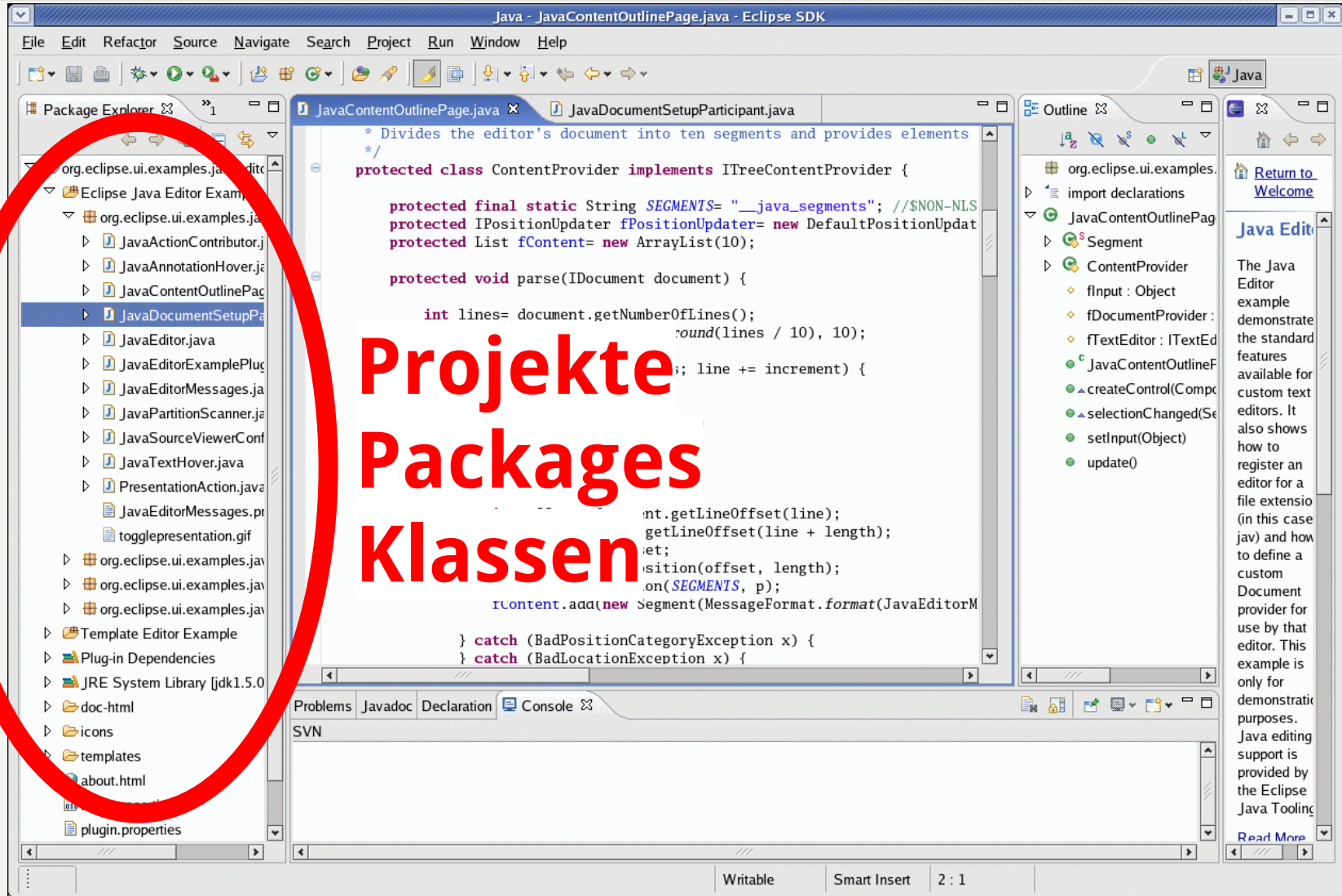


**Download:**

**<https://eclipse.org/downloads/>**

**CIP:**

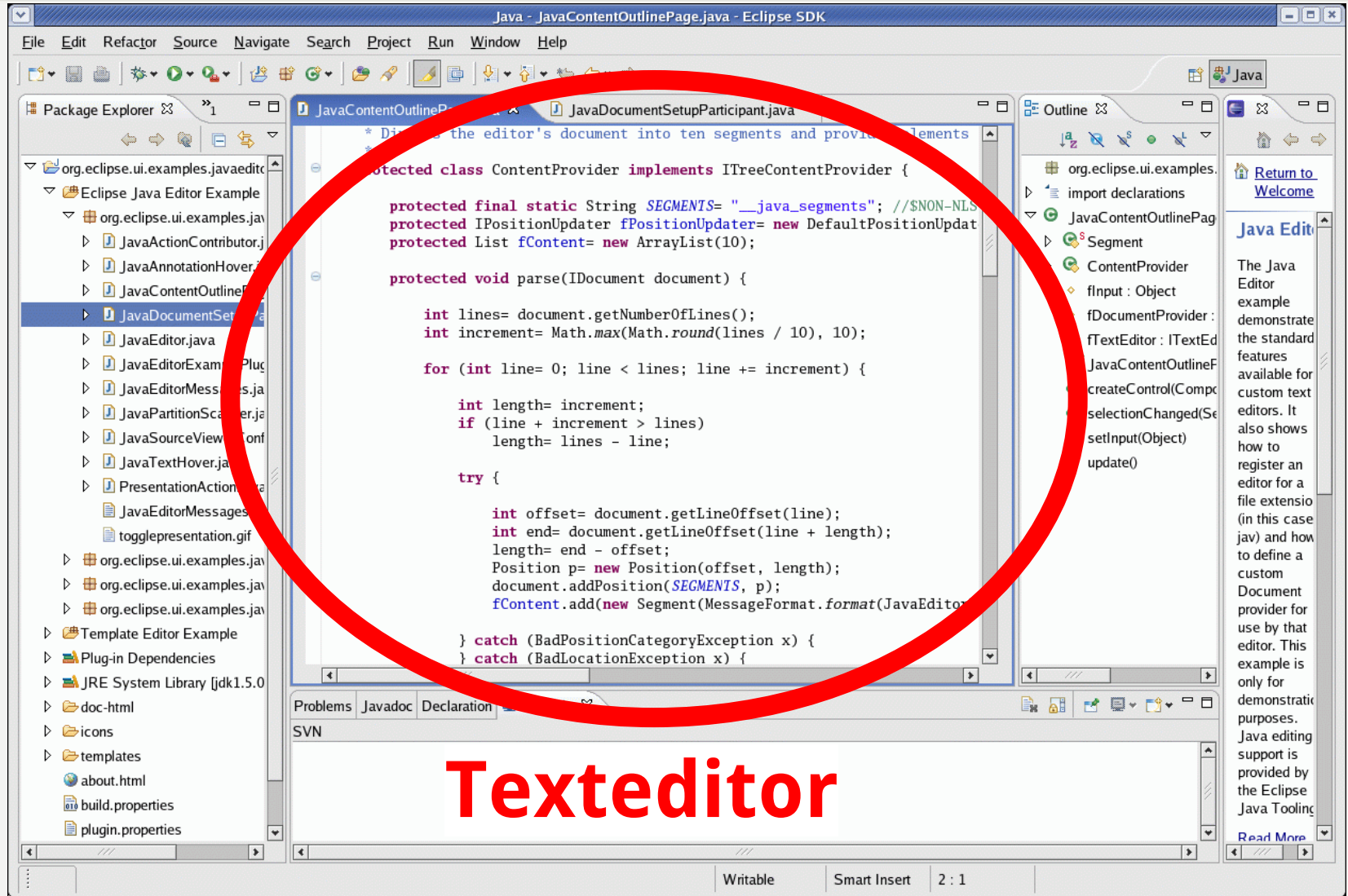
**`/home/proj/software/eclipse/`**



The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays a project structure with a red circle highlighting the 'org.eclipse.ui.examples.java' package. The main editor shows the source code for 'JavaContentOutlinePage.java'. The Outline view on the right shows the class hierarchy. The bottom status bar indicates 'Writable', 'Smart Insert', and '2 : 1'.

```
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= round(lines / 10), 10);  
  
        for (int i= 0; i < lines; i += increment) {  
  
            int start= i;  
            int end= i + increment;  
            int lineOffset= document.getLineOffset(i);  
            int lineLength= document.getLineOffset(i + length);  
            IContentProvider content= new ContentProvider(i, lineOffset, lineLength);  
            fContent.add(new Segment(MessageFormat.format(JavaEditorM  
        } catch (BadPositionCategoryException x) {  
        } catch (BadLocationException x) {  
        }  
    }  
}
```

# Projekte Packages Klassen



Java - JavaContentOutlinePage.java - Eclipse SDK

File Edit Refactor Source Navigate Search Project Run Window Help

Package Explorer

org.eclipse.ui.examples.javaedit

Eclipse Java Editor Example

org.eclipse.ui.examples.javaedit

JavaActionContributor.java

JavaAnnotationHover.java

JavaContentOutlinePage.java

JavaDocumentSetupParticipant.java

JavaEditor.java

JavaEditorExamplePlugin.java

JavaEditorMessages.java

JavaPartitionScanner.java

JavaSourceViewerContent.java

JavaTextHover.java

PresentationActionMessages.java

JavaEditorMessages.java

togglepresentation.gif

org.eclipse.ui.examples.javaedit

org.eclipse.ui.examples.javaedit

org.eclipse.ui.examples.javaedit

Template Editor Example

Plug-in Dependencies

JRE System Library [jdk1.5.0]

doc.html

icons

templates

about.html

build.properties

plugin.properties

Outline

org.eclipse.ui.examples.javaedit

Return to Welcome

Java Editor

The Java Editor example demonstrates the standard features available for custom text editors. It also shows how to register an editor for a file extension (in this case jav) and how to define a custom Document provider for use by that editor. This example is only for demonstrative purposes. Java editing support is provided by the Eclipse Java Tooling

```
protected class ContentProvider implements ITreeContentProvider {  
  
    protected final static String SEGMENTS= "__java_segments"; //NON-NLS  
    protected IPositionUpdater fPositionUpdater= new DefaultPositionUpdater();  
    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(JavaEditorMessages.  
                ) catch (BadPositionCategoryException x) {  
                } catch (BadLocationException x) {  
                }  
            }  
        }  
    }  
}
```

Problems Javadoc Declaration

SVN

Writable Smart Insert 2 : 1

# Texteditor





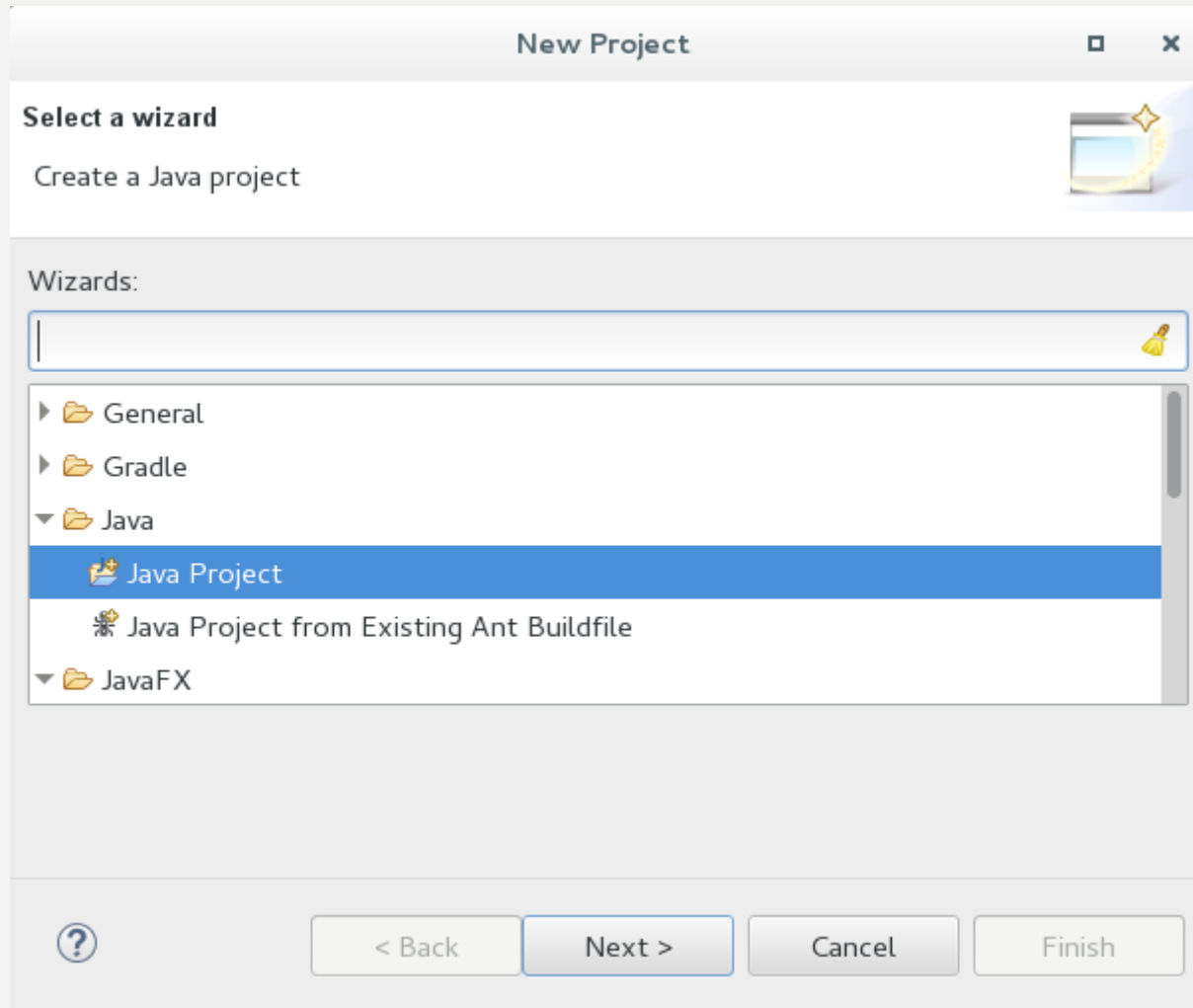
The screenshot shows the Eclipse IDE interface. The main editor displays the code for `ContentProvider` in `JavaContentOutlinePage.java`. The code is as follows:

```
protected class ContentProvider implements ITreeContentProvider {  
  
    protected final static String SEGMENTS= "__java_segments"; //$NON-NLS-1$  
    protected IPositionUpdater fPositionUpdater= new DefaultPositionUpdater();  
    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;  
                document.setSelection(p, length, fContent);  
            } catch (BadPartException e) {}  
        }  
    }  
}
```

The `Console` tab at the bottom of the IDE is circled in red. The `Outline` view on the right shows the class structure, and the `Java Editor` view on the far right provides a tutorial for creating a custom editor.



The screenshot shows the Eclipse IDE interface. The title bar reads "Java - Eclipse". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for file operations, navigation, and development. The "Project Explorer" on the left shows a tree view of projects, with "GOBI" selected. A context menu is open over "GOBI", listing actions such as "New...", "Go Into", "Show In", "Copy", "Paste", "Delete", "Refactor", "Import...", "Export...", "Refresh", "Close Project", "Validate", and "Run As". The "New..." option is highlighted, and a sub-menu is visible with options like "Project...", "File", "Folder", "Annotation", "Class", "Enum", "Interface", "Package", "Source Folder", "Example...", and "Other...". The main editor area is empty, and the "Console" view on the right shows "No consoles to display at this time."





New Java Project

Create a Java Project

Enter a project name.

Project name:

Use default location

Location:

JRE

Use an execution environment JRE:

Use a project specific JRE:

Use default JRE (currently 'jdk1.8.0\_102') [Configure JREs...](#)

Project layout

Use project folder as root for sources and class files

Create separate folders for sources and class files [Configure default...](#)

Working sets

Add project to working sets

Working sets:





The screenshot shows the Eclipse IDE interface. The title bar reads "Java - Eclipse". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for file operations, running, and debugging. The "Project Explorer" on the left shows a project structure with folders like "AlgoBio", "BeklopteTests", and "src". A context menu is open over the "src" folder, with the "New" option selected. The "New" submenu is also open, showing options like "Project...", "Annotation", "Class", "Enum", "Interface", "Package", "Example...", and "Other...". The "Class" option is highlighted. The "Console" view on the right is empty, displaying "No consoles to display at this time."



New Java Class

Java Class  
Create a new Java class.

Source folder:

Package:

Enclosing type:

Name:

Modifiers:  public  package  private  protected  
 abstract  final  static

Superclass:

Interfaces:

Which method stubs would you like to create?

public static void main(String[] args)  
 Constructors from superclass  
 Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

Generate comments



Java - BeklopteTests/src/MyFirstClass.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access Java Debug Team Synchronizing Git Resource

Problems Project Explorer

- AlgoBio
- BeklopteTests
  - src
    - (default package)
      - Brett.java
      - GraphicsTest.java
      - GUI.java
      - MyFirstClass.java**
      - tester.java
      - Tests\_1.java
      - UpperCasedTextFieldTester.java
      - MyFirstClass.java
    - JRE System Library [JavaSE-1.8]
  - BioinformaticTasks [SAMsTOOL master]
  - BioRessourcen
  - Bomberman
  - Breakout
  - FXGL
  - FXGL2
  - GOBI
  - GoBi..WS2016-17
  - GOR [gor master]
  - GOThello
  - Gradius\_X
  - JavaFX
  - LeetCode
  - Othello

MyFirstClass.java

```
1 public class MyFirstClass {  
2  
3  
4  
5  
6 }  
7
```

Debug Call Hier Console Git Stagi

No consoles to display at this time.

Writable Smart Insert 4 : 5



## - Vereinfacht den Ablauf:

- Write a complete program.
- Compile it and fix any errors.
- Run the program.
- Figure out what is wrong with it.
- Edit it.
- Repeat the process.



**Try the code.**

## - Wichtige Befehle:

- `jshell -v`
- `/exit`
- `/help intro`
- `/list`



## Beispiele für:

### Berechnungen

```
File Edit View Search Terminal Help
mano@mano-Inspiron-3542 ~ $ jshell --start DEFAULT --start PRINTING
| Welcome to JShell -- Version 9
| For an introduction type: /help intro

jshell> 2+3
$1 ==> 5

jshell> 2-7*9/3+6
$2 ==> -13

jshell> $1-$2
$3 ==> 18

jshell> $3/7.3
$4 ==> 2.4657534246575343

jshell> /list

 1 : 2+3
 2 : 2-7*9/3+6
 3 : $1-$2
 4 : $3/7.3

jshell>
```

[https://www.developer.com/imagesvr\\_ce/5633/Shell07.jpg](https://www.developer.com/imagesvr_ce/5633/Shell07.jpg)

### Funktionen

```
Administrator: C:\Windows\system32\cmd.exe - jshell

jshell> int sum (int a, int b) <
...> return a+b;
...>
! created method sum(int,int)

jshell> /methods
! int sum(int,int)

jshell> sum(2,2)
$2 ==> 4

jshell> /list sum

 1 : int sum (int a, int b) <
   return a+b;
   >

jshell> int sum (int a, int b) <
...> int c = a+b;
...> return c;
...>
! modified method sum(int,int)

jshell> /list sum

 3 : int sum (int a, int b) <
   int c = a+b;
   return c;
   >

jshell>
```

<https://cdn2.howtodoinjava.com/wp-content/uploads/2017/08/Working-with-Methods-in-JShell.png>



- ... ist der Einstiegspunkt in ein Java-Programm
- ... stellt eine Verbindung zu allen beteiligten Klassen her

```
1 public class HelloWorld
2 {
3     public static void main(String[] args)
4     {
5         int a = 1;
6         int b = a+1;
7
8         System.out.println("a+b = " + (a+b));
9     }
10 }
```





- ... nimmt die Eingaben aus der Konsole entgegen
- ... alle Argumente werden als String übergeben
- ... das erste Argument kann über `args[0]` angesprochen werden

```
public static void main(String[] args) {  
    String s = args[0]; //erster Parameter  
    System.out.println(s);  
    ...  
    System.out.println(args[args.length-1]); //letzter  
        Parameter  
}
```



## Datentypen in Java:

**Buchstaben:** **char (Character)**

**Wörter:** **String**

**Zahlen aus N:** **int (Integer)**

**Kommazahlen:** **float, double**

[https://de.wikibooks.org/wiki/Java\\_Standard:\\_Primitive\\_Datentypen](https://de.wikibooks.org/wiki/Java_Standard:_Primitive_Datentypen)

## Umwandeln von Datentypen:

```
String s1 = "5";  
String s2 = "10.5";
```

```
int f1 = Integer.parseInt(s1);  
float f2 = Float.parseFloat(s2);
```



## Programmabfluss kann durch Fallunterscheidungen verändert werden:

```
1 if (Ausdruck) { // z.B.: a == b, a!=b, a>b s...
2   //Falls der Ausdruck erfuehlt ist
3 }
```

```
1 if (Ausdruck) {
2   //Falls der Ausdruck erfuehlt ist
3 }else{
4   //Falls der Ausdruck nicht erfuehlt ist
5 }
```

```
1 if (Ausdruck) {
2   //Falls der Ausdruck erfuehlt ist
3 }else if (2. Ausdruck) {
4   //Falls der Ausdruck nicht erfuehlt ist
5 }...{
6 }else{
7   //Falls alle anderen Ausdruecke nicht erfuehlt sind
8 }
```



## Der Datentyp String muss mit `.equals(anotherString)` auf Gleichheit geprüft werden:

```
String s1 = "Tutorium";
String s2 = "Bioinformatik";
String s3 = "Tutorium";

int a = 2;
int b = 3;

if (s1.equals(s2)) {
    System.out.println("Wir sind gleich");
} else if (a==b || s1.equals(s3)) {
    System.out.println("Mindestens einer von uns ist gleich");
} else {
    System.out.println("Keiner ist gleich");
}
```



## Zwei einfache Zeilen:

```
javac Klasse.java  
java Klasse Parameter
```

```
MINGW64:/c:/Users/Konstantin/Desktop/Uni/3.Semester/ProPra/test  
Konstantin@KonnyLaptop MINGW64 ~/Desktop/Uni/3.Semester/ProPra/test  
$ javac MSquare.java  
Konstantin@KonnyLaptop MINGW64 ~/Desktop/Uni/3.Semester/ProPra/test  
$ java MSquare matrix  
Congratz, you have a magic square!!  
Magic Number: 34  
Number of cells: 16  
Konstantin@KonnyLaptop MINGW64 ~/Desktop/Uni/3.Semester/ProPra/test  
$ -  
>
```