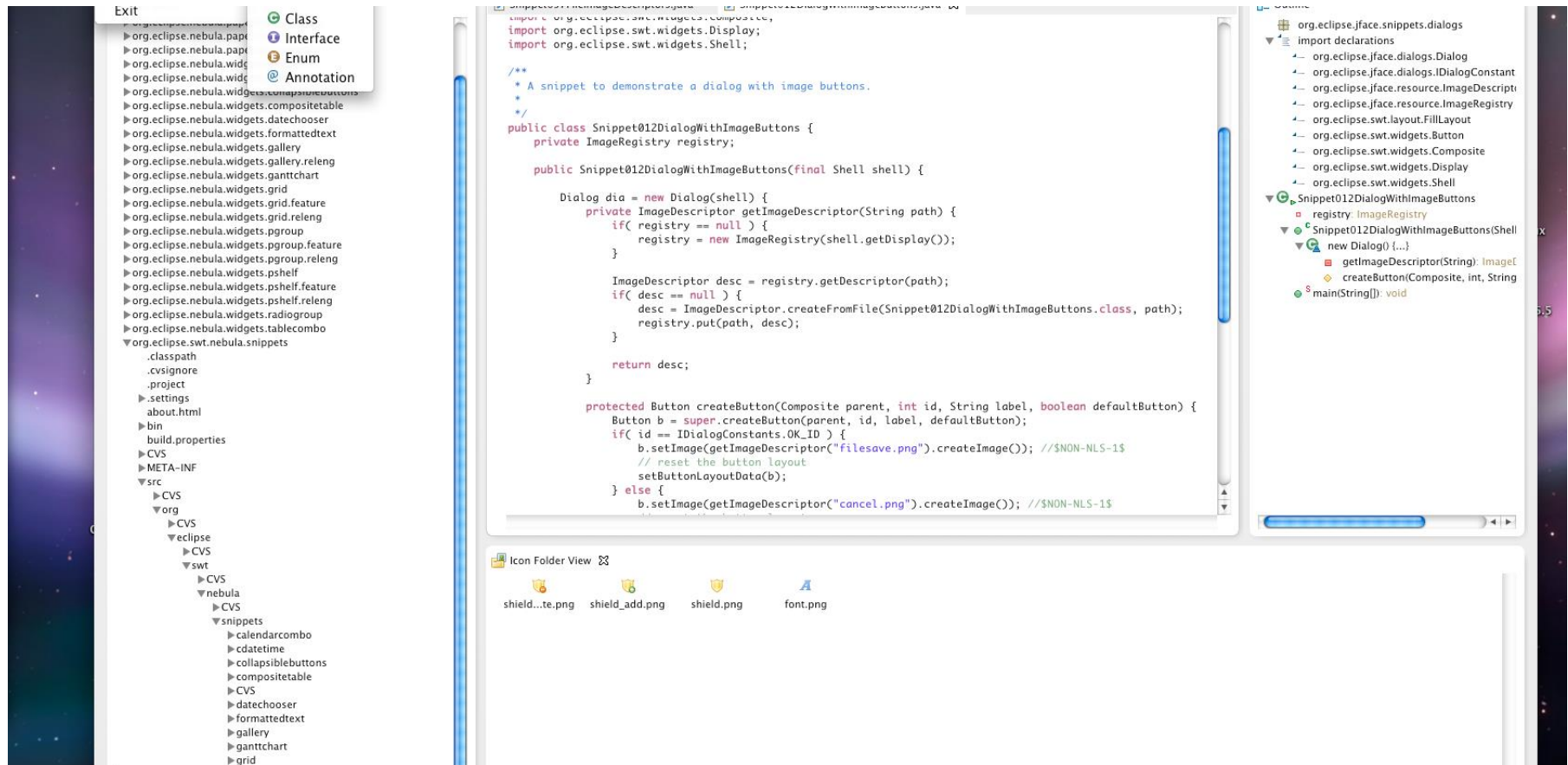


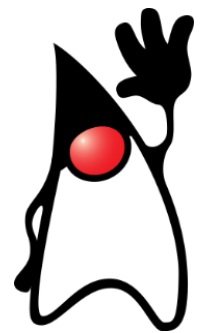
## Java Grundlagen und Entwicklungsumgebung



### 1. Java Grundlagen: Entwicklungszyklus, Entwicklungsumgebung

2. Datentypen, Kodierung, Binärzahlen, Variablen, Arrays
3. Ausdrücke, Operatoren, Schleifen und Verzweigungen
4. Blöcke, Sichtbarkeit und Methoden (Teil 1)
5. Grundkonzepte der Objektorientierung
6. Objektorientierung: Sichtbarkeit, Vererbung, Methoden (Teil 2), Konstruktor
7. Packages, lokale Klassen, abstrakte Klassen und Methoden, Interfaces, enum
8. Arbeiten mit Objekten: Identität, Listen, Komparatoren, Kopien, Wrapper, Iterator
9. Fehlerbehandlung: Exceptions und Logging
10. Utilities: Math, Date, Calendar, System, Random
11. Rekursion, Sortieralgorithmen und Collections
12. Nebenläufigkeit: Arbeiten mit Threads
13. Benutzeroberflächen mit Swing
14. Streams: Auf Dateien und auf das Netzwerk zugreifen

- 1991: Erste Version in 18 Monaten entwickelt (The Green Project, SUN Microsystems)
- Ursprüngliches Entwicklungsziel: Sprache und Systemumgebung zur Steuerung von Alltagsgeräten
- Ende 1992: Ausrichtung auf Internet-Anwendungen
- 23.05.1995: Erste öffentliche Vorstellung von Java
- 1995: Integration in den Netscape Navigator (Applets)



Duke

## Wichtige Versionsschritte:

- 1996 Java Version **1**
- 1998 Java 1.**2** mit Just-In-Time Compiler (JIT) Add-On
  - Unterschied zu Ahead-of-Time Compiler
  - Siehe: [http://en.wikipedia.org/wiki/Just-in-time\\_compilation](http://en.wikipedia.org/wiki/Just-in-time_compilation)
- 2000 Java 1.**3** mit HotSpot-Compiler
- 2002 Java 1.**4** mit Assertions
- 2004 Java **5** mit Generics und Annotationen
- 2006 Java **6** mit erweiterten Verwaltungstools
- 2011 Java **7** mit NIO2.0
- 2014 Java **8 (Update 40)**

Siehe: [http://de.wikipedia.org/wiki/Java\\_%28Technik%29#Versionen](http://de.wikipedia.org/wiki/Java_%28Technik%29#Versionen)

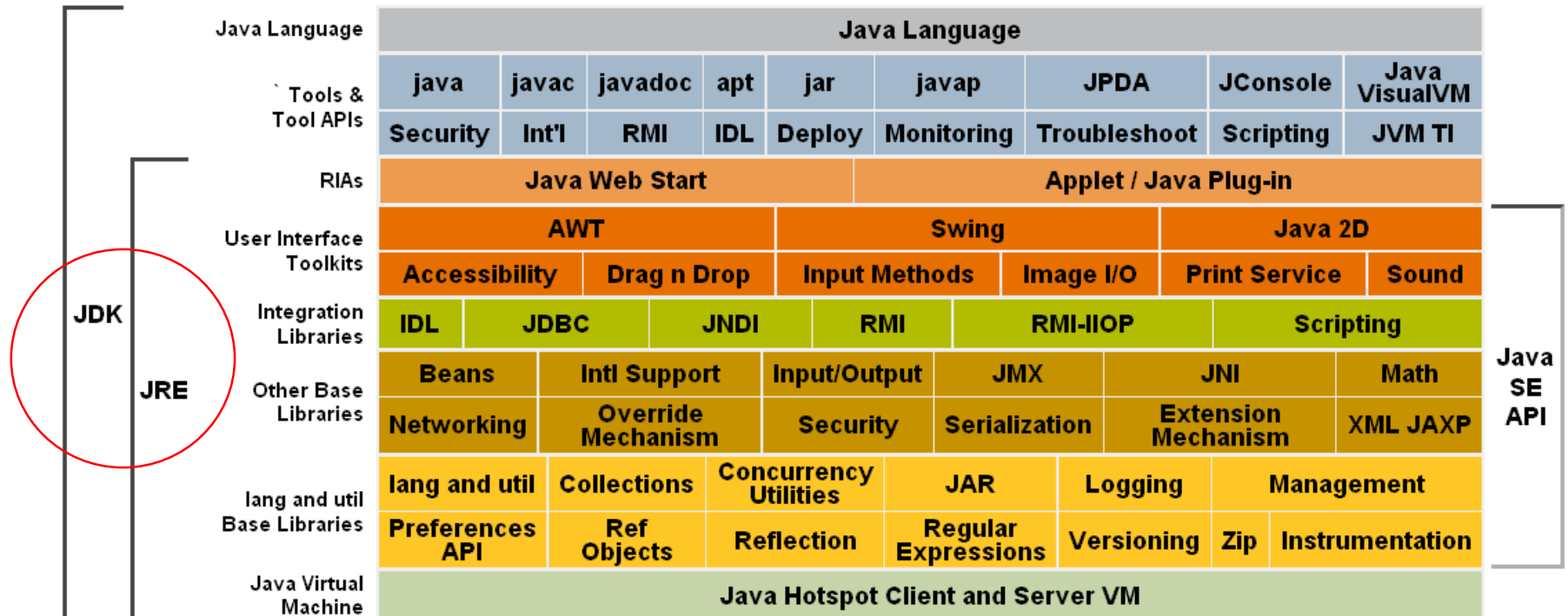
- Java Standard Edition (Java SE)
  - Fundamentale Java Klassen (String, System, I/O ...)
  - Toolkits für das Benutzerinterface (AWT, Swing, ...)
  - ...
- Java Enterprise Edition (Java EE)
  - EJB – Enterprise Java Beans
  - JPA – Java Persistence API

“The platform was known as *Java 2 Platform, Enterprise Edition* or *J2EE* until the name was changed to *Java EE* in version 5. The current version is called *Java EE 6*.”

- Java Micro Edition (Java ME)

## JDK: Java Developer Kit

## JRE: Java Runtime Environment / Java Virtual Machine



Quelle: <http://download.oracle.com/javase/6/docs/>

## Download bei Oracle:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



Here are the Java SE downloads in detail:

Java Platform, Standard Edition		
<b>Java SE 7u3</b> This release includes security fixes. <a href="#">Learn more</a> ▶  <b>"What Java Do I Need?"</b> You must have a copy of the <b>JRE</b> (Java Runtime Environment) on your system to <b>run</b> Java applications and applets. To <b>develop</b> Java applications and applets, you need the <b>JDK</b> (Java Development Kit), which includes	<b>JDK</b> <a href="#">Download</a>  <b>JDK 7 Docs</b> <ul style="list-style-type: none"> <li><a href="#">Installation Instructions</a></li> </ul>	<b>JRE</b> <a href="#">Download</a>  <b>JRE 7 Docs</b> <ul style="list-style-type: none"> <li><a href="#">Installation Instructions</a></li> </ul>



Plattform-  
unabhängig

```

demoClass.java X
1 public class demoClass {
2
3     public static void main(String[] args) {
4
5         System.out.println("Hello World");
6
7     }
8 }
  
```

.java – Datei

```

6a 61 76 51 2f 6c 61 5a 67 2f 4f 02 6a 65 63 74
01 00 06 3e 69 6e 69 74 3e 01 00 03 28 29 56 01
00 04 43 6f 64 65 0a 00 03 00 09 0c 00 05 00 06
01 00 0f 4c 69 6e 65 4e 75 6d 62 65 72 54 61 62
6c 65 01 00 12 4c 6f 63 61 6c 5a 63 72 69 61 62
6c 65 54 61 62 6c 65 01 00 04 74 68 69 73 01 00
0c 4c 48 65 6c 6c 6f 6f 72 6c 64 3b 01 00 04
6d 61 69 6e 01 00 16 28 5b 4c 6a 61 76 61 2f 6c
61 6e 67 2f 53 74 72 69 6e 67 3b 29 56 09 00 11
00 13 07 00 12 01 00 10 6a 61 76 61 2f 6c 61 6e
67 2f 53 79 73 74 65 6d 0c 00 14 00 15 01 00 03
ef 75 74 01 00 15 4c 6a 61 76 61 2f 69 6f 2f 50
72 69 6e 74 53 74 72 65 61 6d 3b 08 00 17 01 00
0b 48 65 6c 6c 6f 20 77 6f 72 6c 64 0a 00 19 00
1b 07 00 1a 01 00 13 6a 61 76 61 2f 69 6f 2f 50
72 69 6e 74 53 74 72 65 61 6d 0c 00 1c 00 1d 01
00 07 70 72 69 6e 74 6c 6e 01 00 15 28 4c 6a 61
76 61 2f 6c 61 6e 67 2f 53 74 72 69 6e 67 3b 29
56 01 00 04 61 72 67 73 01 00 13 5b 4c 6a 61 76
61 2f 6c 61 6e 67 2f 53 74 72 69 6e 67 3b 01 00
0a 53 6f 75 72 69 65 4e 69 6c 65 01 00 0f 48 65
6c 6c 6f 6f 6f 72 6c 64 2a 6a 61 76 61 00 21 00
01 00 03 00 00 00 00 00 02 00 01 00 05 00 06 00
01 00 07 00 00 2f 00 01 00 01 00 00 00 05 2a
b7 00 0b b1 00 00 02 00 0a 00 00 00 06 00 01
00 00 02 00 0b 00 00 00 0c 00 01 00 00 00 05
00 0c 00 04 00 00 09 00 0a 00 0f 00 01 00 07
00 00 00 37 00 02 00 01 00 00 09 b2 00 1d 12
16 b6 00 18 b1 00 00 00 02 00 0a 00 00 00 0a 00
02 00 00 00 05 00 08 00 06 00 0b 00 00 0c 00
01 00 00 00 09 00 1e 00 1f 00 00 01 00 20 00
00 00 02 00 21
  
```

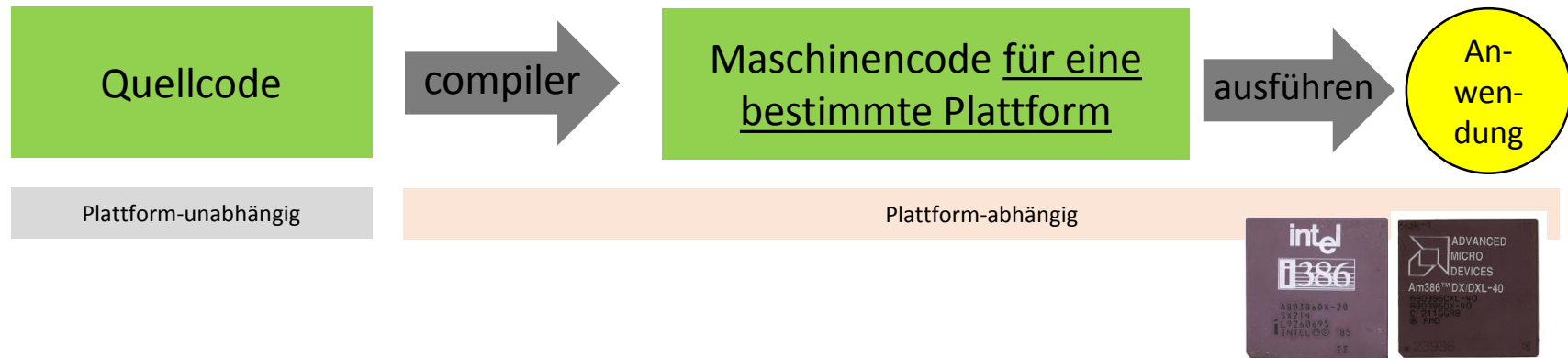
.class – Datei

```

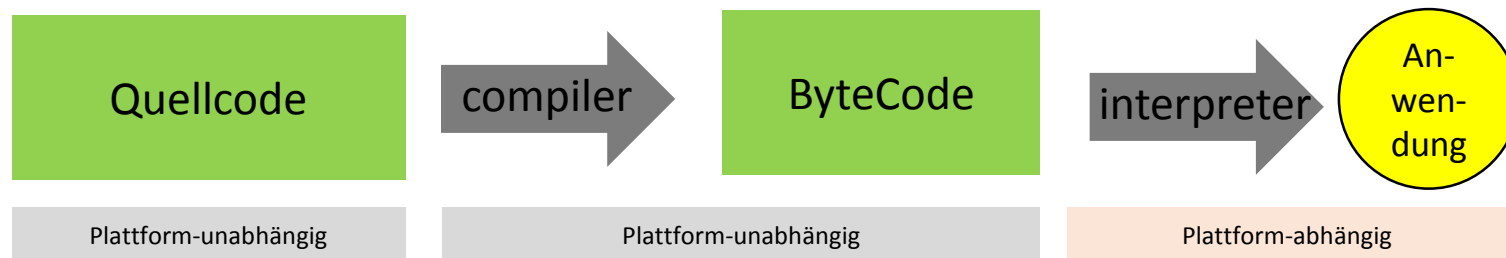
C:\>java demoClass
Hello World
C:\>
  
```



## Kompilieren zu Maschinen-Code (z. B. C / C++):

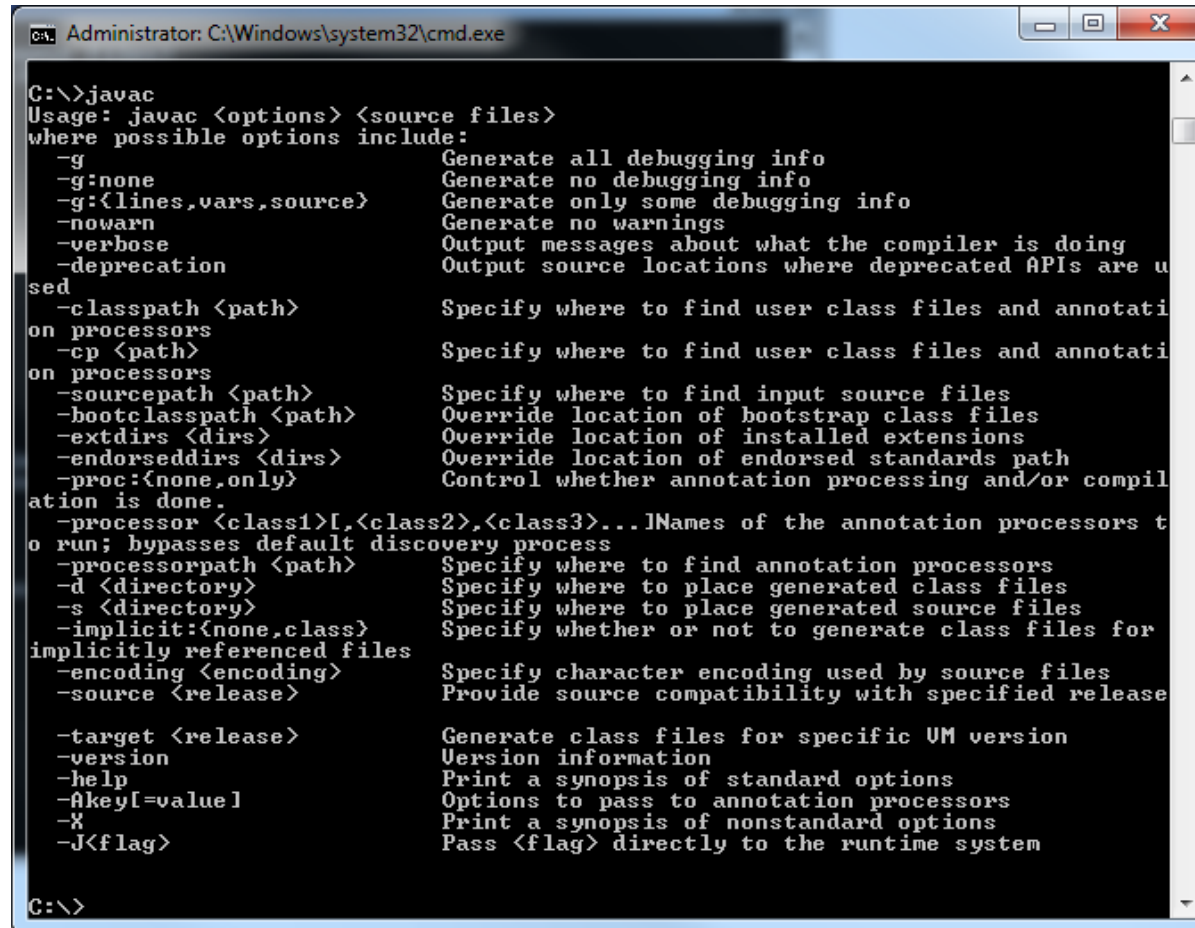


## Kompilieren zu ByteCode (z.B. Java):



„Write Once, Run Anywhere“

## javac

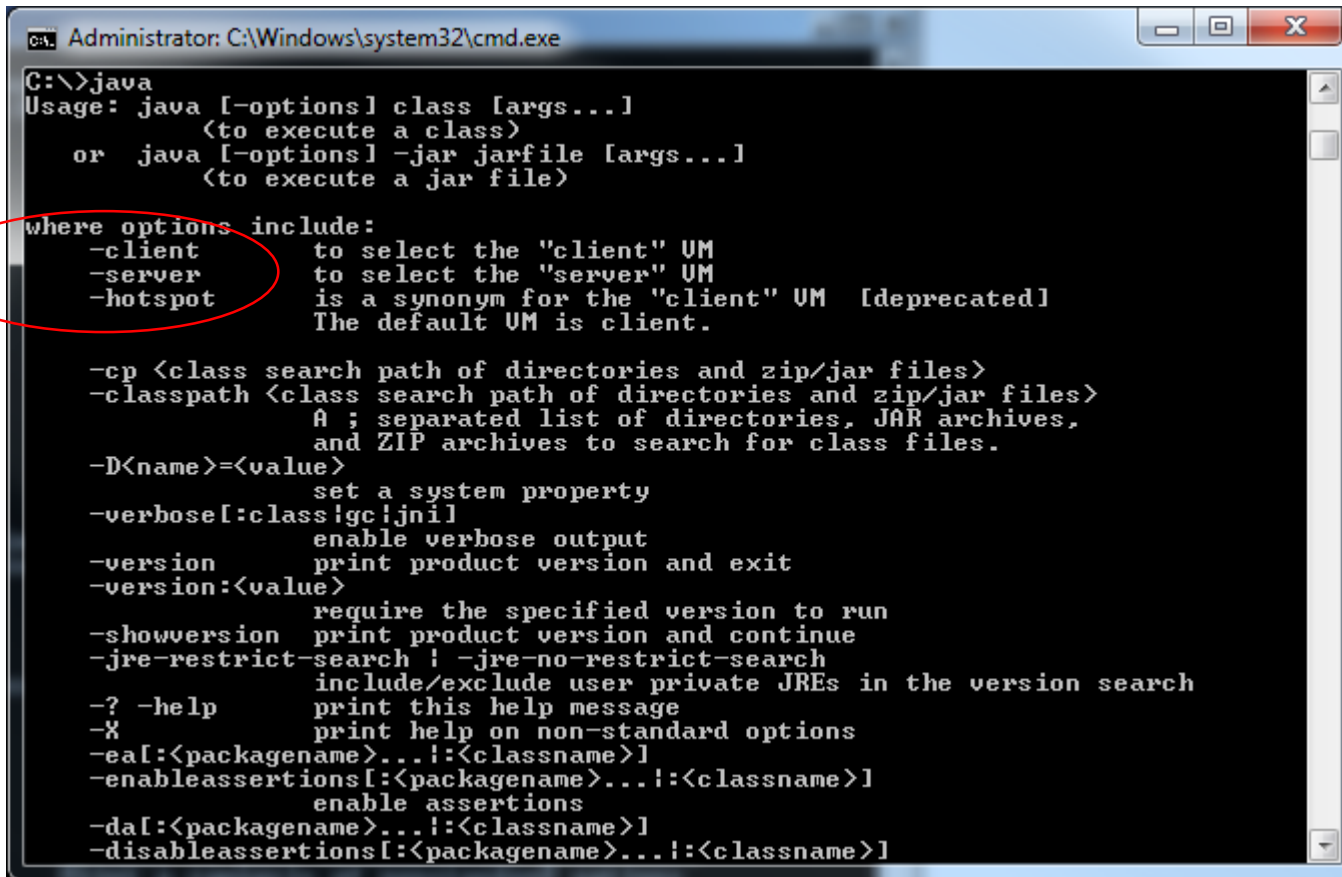


```
Administrator: C:\Windows\system32\cmd.exe

C:\>javac
Usage: javac <options> <source files>
where possible options include:
  -g                      Generate all debugging info
  -g:none                 Generate no debugging info
  -g:<lines,vars,source>  Generate only some debugging info
  -nowarn                 Generate no warnings
  -verbose                Output messages about what the compiler is doing
  -deprecation            Output source locations where deprecated APIs are used
  -classpath <path>      Specify where to find user class files and annotations
  -cp <path>              Specify where to find user class files and annotations
  -sourcepath <path>      Specify where to find input source files
  -bootclasspath <path>  Override location of bootstrap class files
  -extdirs <dirs>         Override location of installed extensions
  -endorseddirs <dirs>   Override location of endorsed standards path
  -proc:<none,only>       Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path>   Specify where to find annotation processors
  -d <directory>          Specify where to place generated class files
  -s <directory>          Specify where to place generated source files
  -implicit:<none,class>  Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding>    Specify character encoding used by source files
  -source <release>        Provide source compatibility with specified release
  -target <release>        Generate class files for specific VM version
  -version                Version information
  -help                  Print a synopsis of standard options
  -Akey[=value]           Options to pass to annotation processors
  -X                     Print a synopsis of nonstandard options
  -J<flag>                Pass <flag> directly to the runtime system

C:\>
```

## java



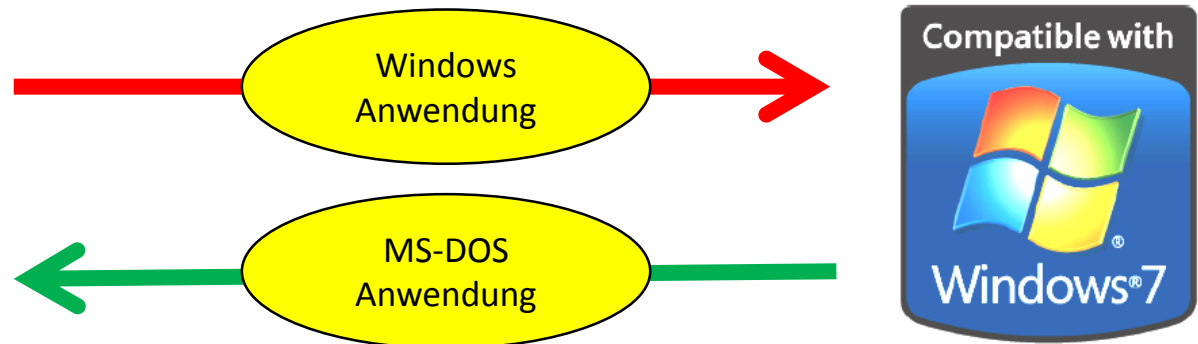
```
C:\>java
Usage: java [-options] class [args...]
        <to execute a class>
    or java [-options] -jar jarfile [args...]
        <to execute a jar file>

where options include:
    -client    to select the "client" VM
    -server    to select the "server" VM
    -hotspot   is a synonym for the "client" VM [deprecated]
               The default VM is client.

    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
               A ; separated list of directories, JAR archives,
               and ZIP archives to search for class files.
    -D<name>=<value> set a system property
    -verbose[:class!gc!jni] enable verbose output
    -version    print product version and exit
    -version:<value> require the specified version to run
    -showversion print product version and continue
    -jre-restrict-search ! -jre-no-restrict-search
               include/exclude user private JREs in the version search
    -? -help    print this help message
    -X          print help on non-standard options
    -ea[:<packagename>...!:<classname>]
               enable assertions
    -da[:<packagename>...!:<classname>]
               disable assertions[:<packagename>...!:<classname>]
```

- **Abwärts- / Rückwärtskompatibel:** Eine neuere (aktuellere) Version unterstützt auch die Anforderungen der älteren Version(en)
- **Aufwärts- / Vorwärtskompatibel:** Die ältere Version erfüllt auch die Anforderung einer neueren Version
- Siehe: [http://de.wikipedia.org/wiki/Kompatibilit%C3%A4t\\_%28Technik%29](http://de.wikipedia.org/wiki/Kompatibilit%C3%A4t_%28Technik%29)

Beispiel: MS-Dos 6.22 und Windows 7 (Abwärtskompatibel)



- Binärkompatibilität (Bytecode)

Class files are forward compatible only.

Beispiel: Ein mit Java 1.4 kompiliertes .class-file läuft auch auf der Java 6 JVM

- Source-Kompatibilität

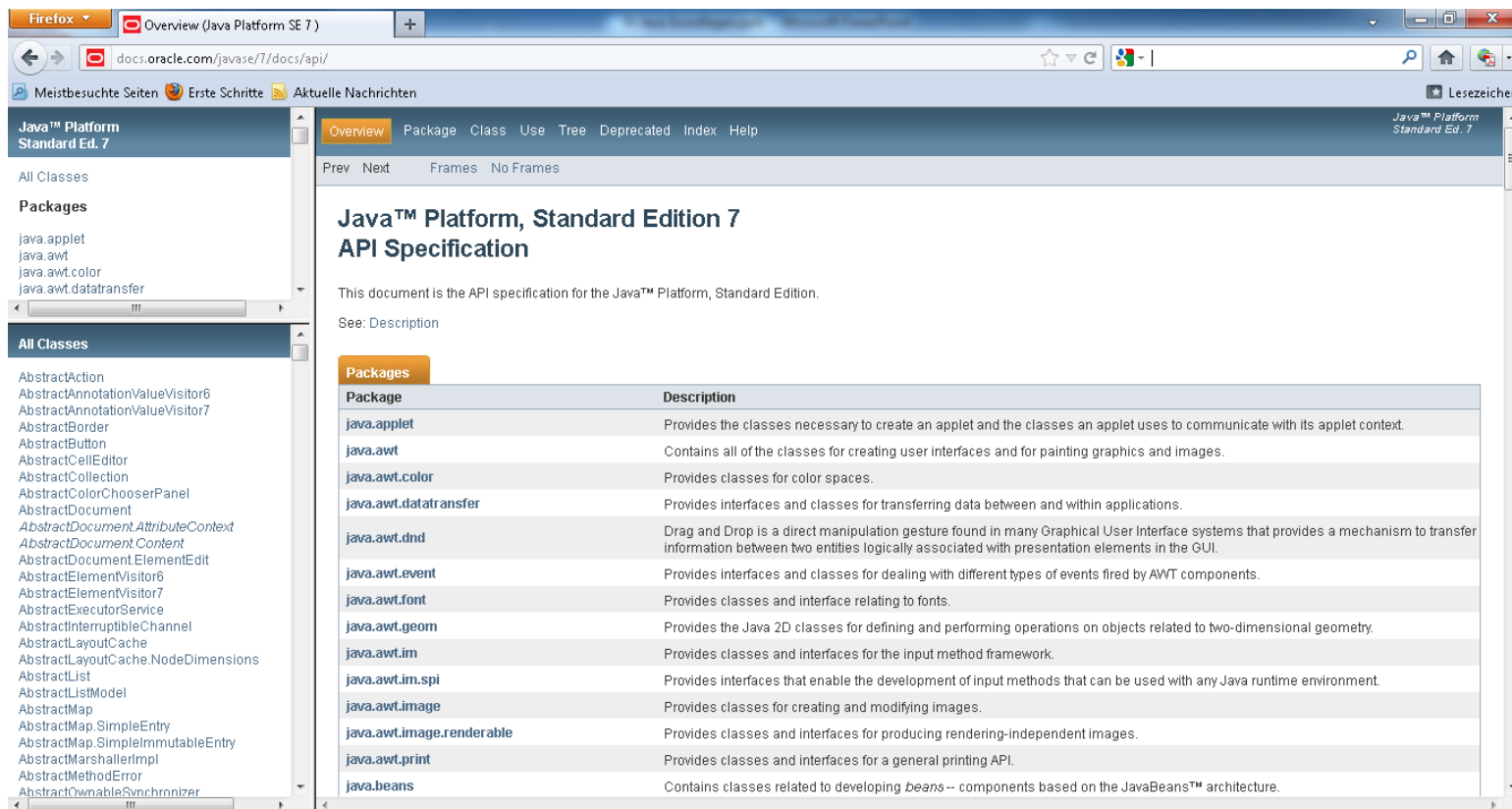
Java SE 6 does not support downward source compatibility. [...] In general, the policy is as follows, except for any [incompatibilities](#) listed further below:

- Maintenance releases (such as 1.4.1, 1.4.2) do not introduce any new language features or APIs. They will maintain source-compatibility with each other.
- Functionality releases and major releases (such as 1.3.0, 1.4.0, 5.0) maintain upwards but not downwards source-compatibility.

**Deprecated APIs** are interfaces that are supported **only** for backwards compatibility.

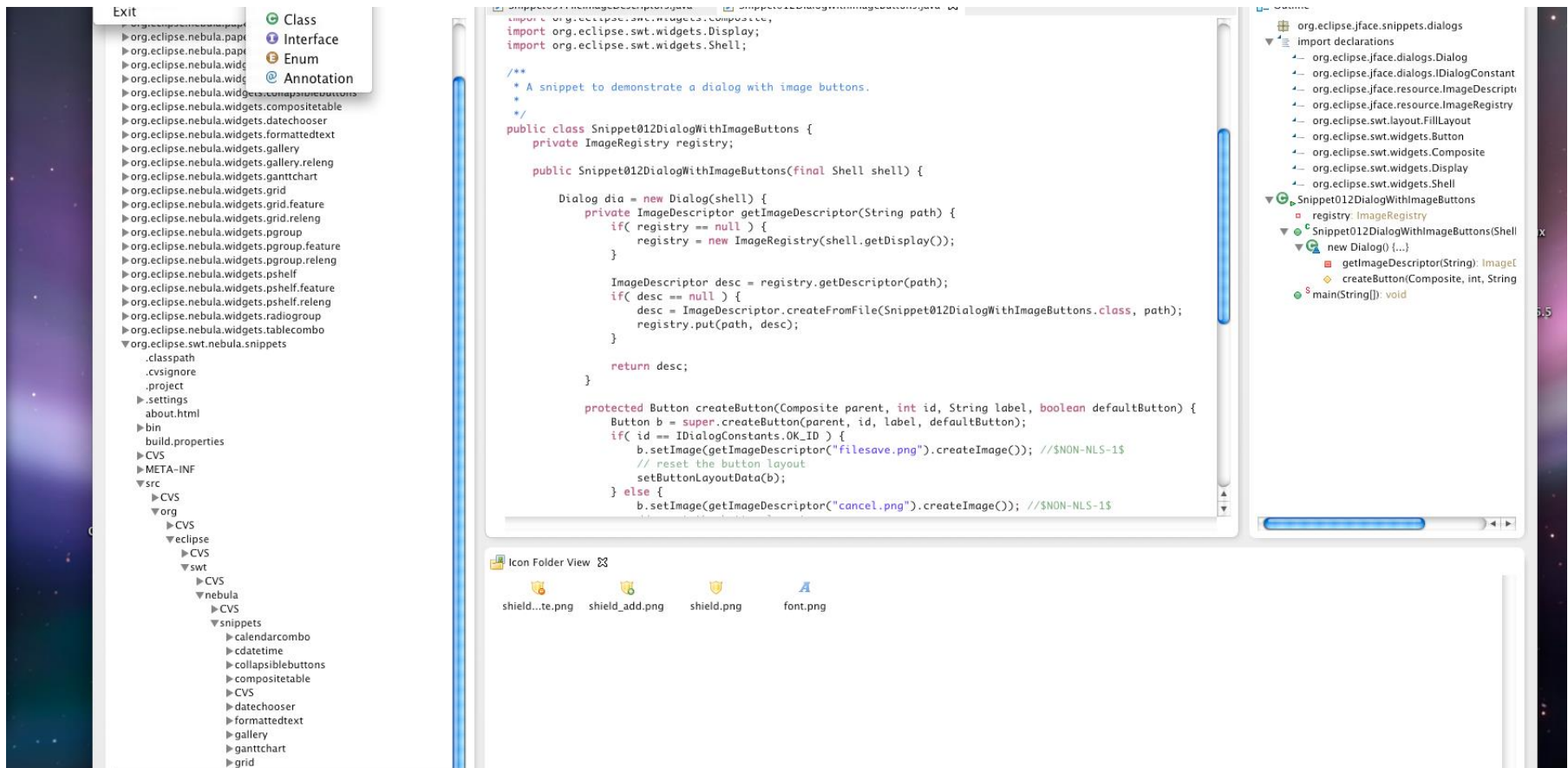
# Application Programming Interface (API)

Java SE 7 API: <http://docs.oracle.com/javase/7/docs/api/>



The screenshot shows the Java™ Platform, Standard Edition 7 API Specification page. The browser is Firefox, and the address bar shows [docs.oracle.com/javase/7/docs/api/](http://docs.oracle.com/javase/7/docs/api/). The page has a navigation bar with tabs: Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. The 'Overview' tab is selected, showing a list of packages and their descriptions.

Package	Description
<code>java.applet</code>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<code>java.awt</code>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<code>java.awt.color</code>	Provides classes for color spaces.
<code>java.awt.datatransfer</code>	Provides interfaces and classes for transferring data between and within applications.
<code>java.awt.dnd</code>	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
<code>java.awt.event</code>	Provides interfaces and classes for dealing with different types of events fired by AWT components.
<code>java.awt.font</code>	Provides classes and interface relating to fonts.
<code>java.awt.geom</code>	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
<code>java.awt.im</code>	Provides classes and interfaces for the input method framework.
<code>java.awt.im.spi</code>	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
<code>java.awt.image</code>	Provides classes for creating and modifying images.
<code>java.awt.image.renderable</code>	Provides classes and interfaces for producing rendering-independent images.
<code>java.awt.print</code>	Provides classes and interfaces for a general printing API.
<code>java.beans</code>	Contains classes related to developing <i>beans</i> -- components based on the JavaBeans™ architecture.



 **eclipse** : <http://www.eclipse.org/downloads/>

Java IDE, tools for Java EE, JPA, JSF, Mylyn...



## Enterprise-Class BIRT Report Server (Free, Not a Trial)

Instantly enhance your Eclipse BIRT reports with interactive, real-time analytics that can be seamlessly embedded into web and mobile apps.



## Eclipse IDE for Java Developers

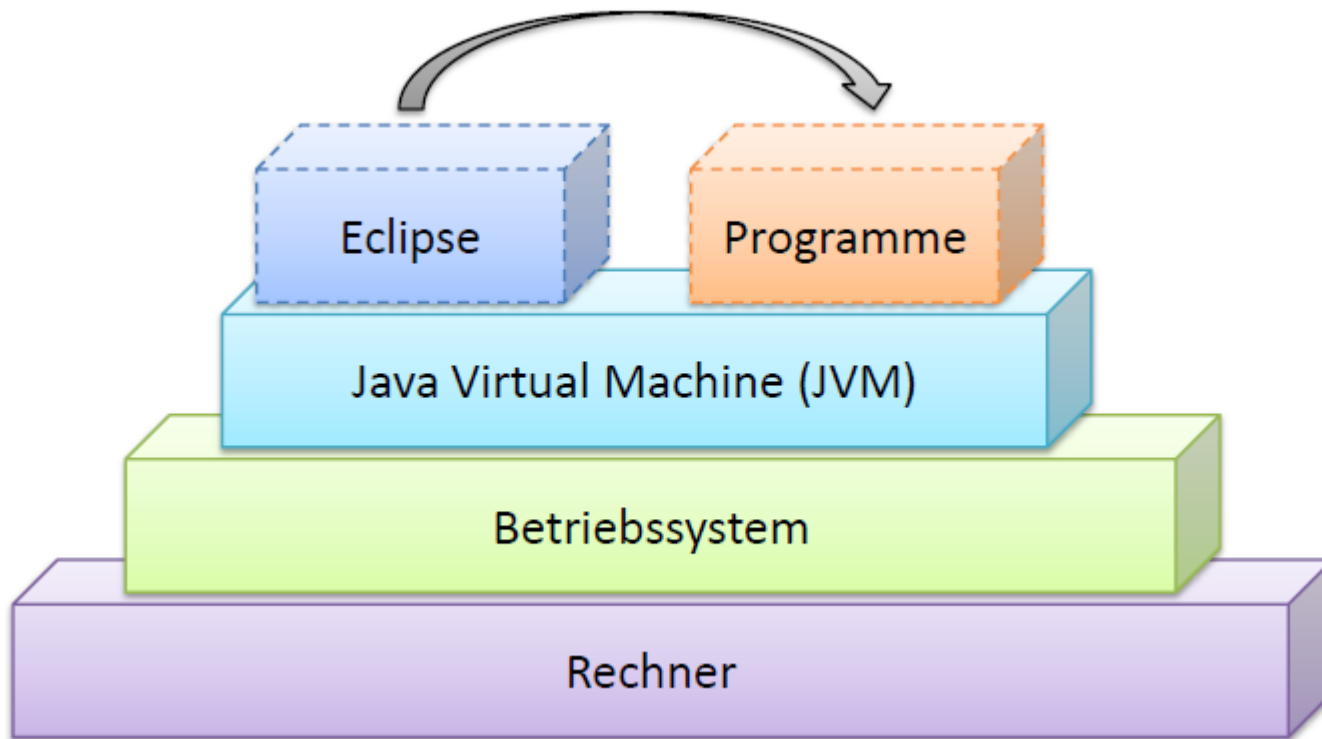
165 MB 200,597 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder...





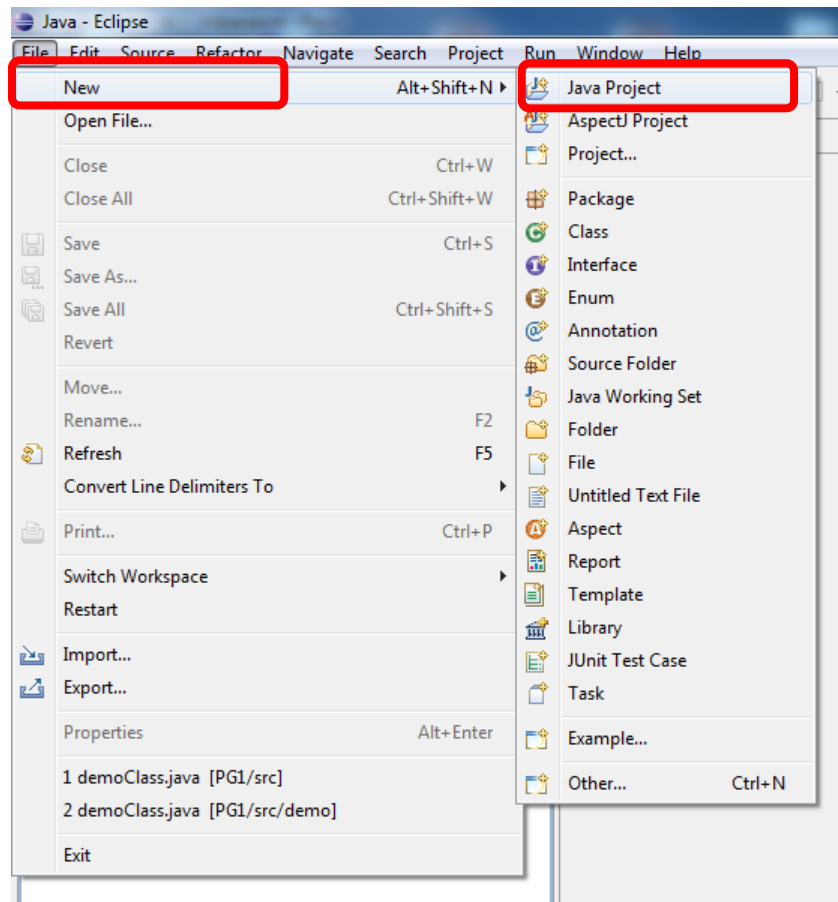
## Eclipse ist selbst ein Programm



## Eclipse starten



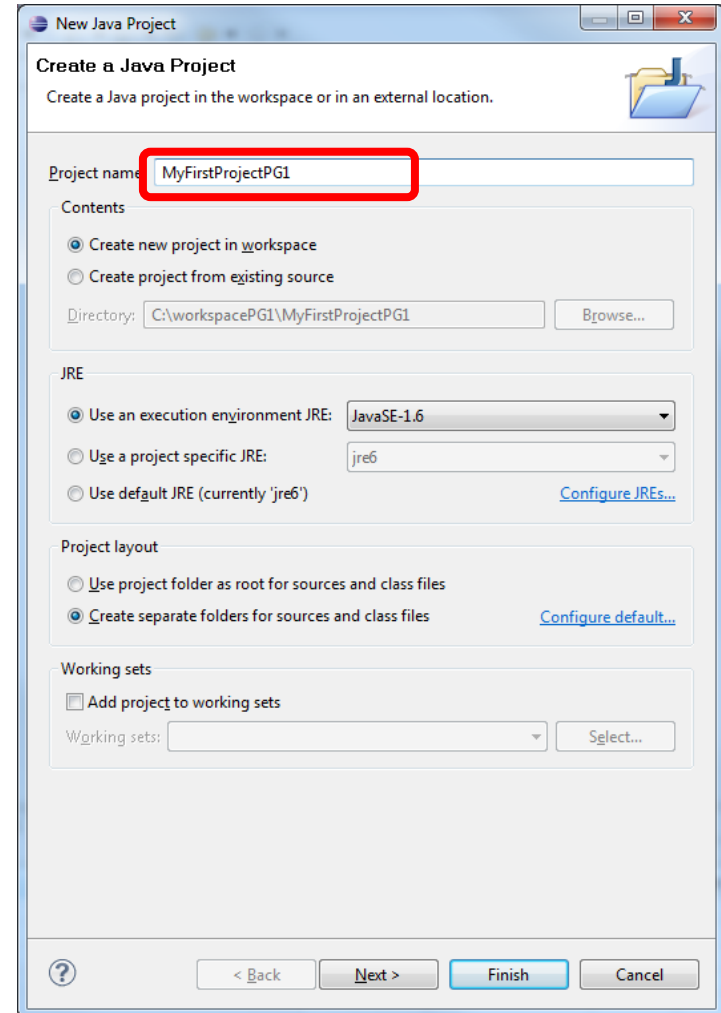
## File > New > Java Project



## Dem Projekt einen Namen geben

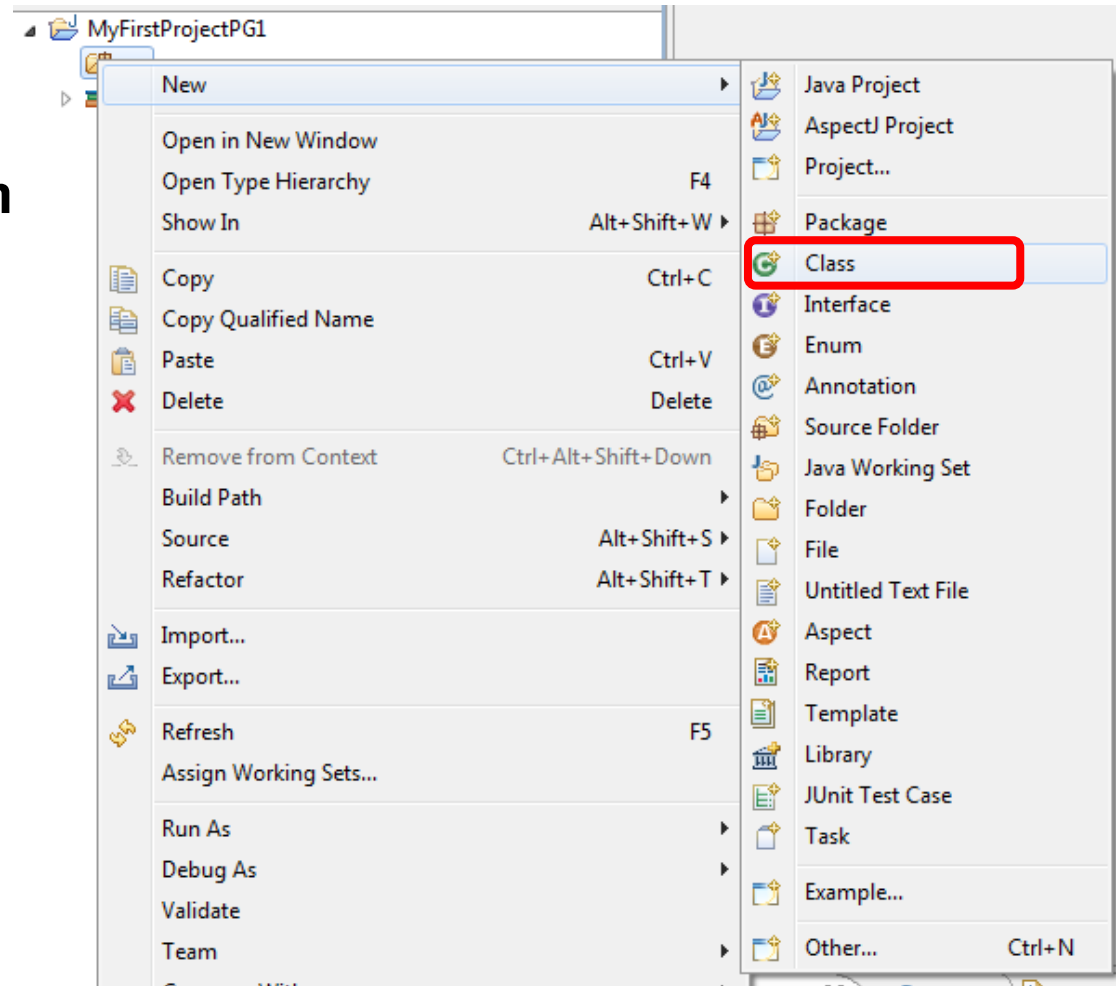
- Hier: „MyFirstProjectPG1“

Jetzt sollte ein Projekt mit diesem Namen im Package-Explorer angezeigt werden



## Den Wizzard für die Erstellung einer neuen Klasse starten

- Rechtsklick auf „src“ im Projekt
- New > Class

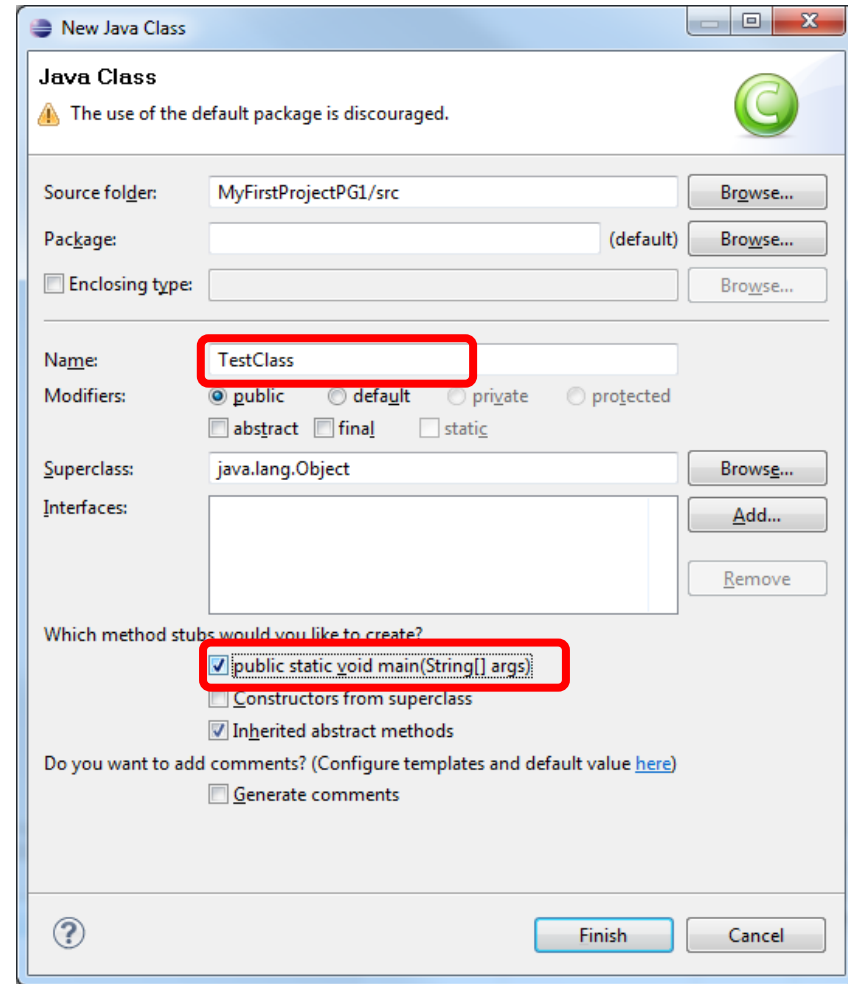


## 1. Klassennamen angeben

>> Hier: TestClass

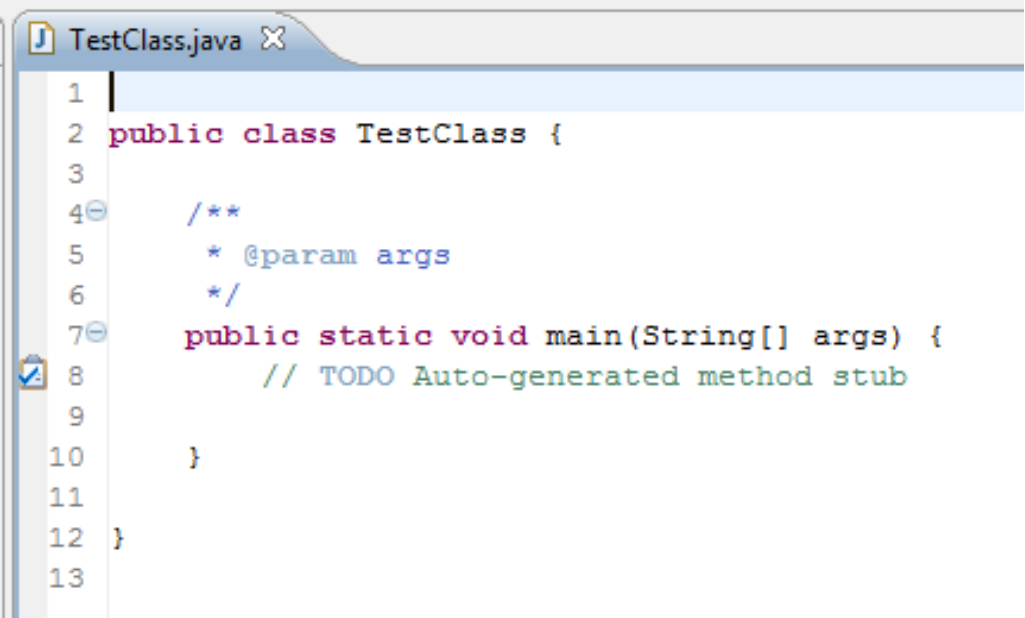
## 2. Main-Methode erzeugen

## 3. Finish



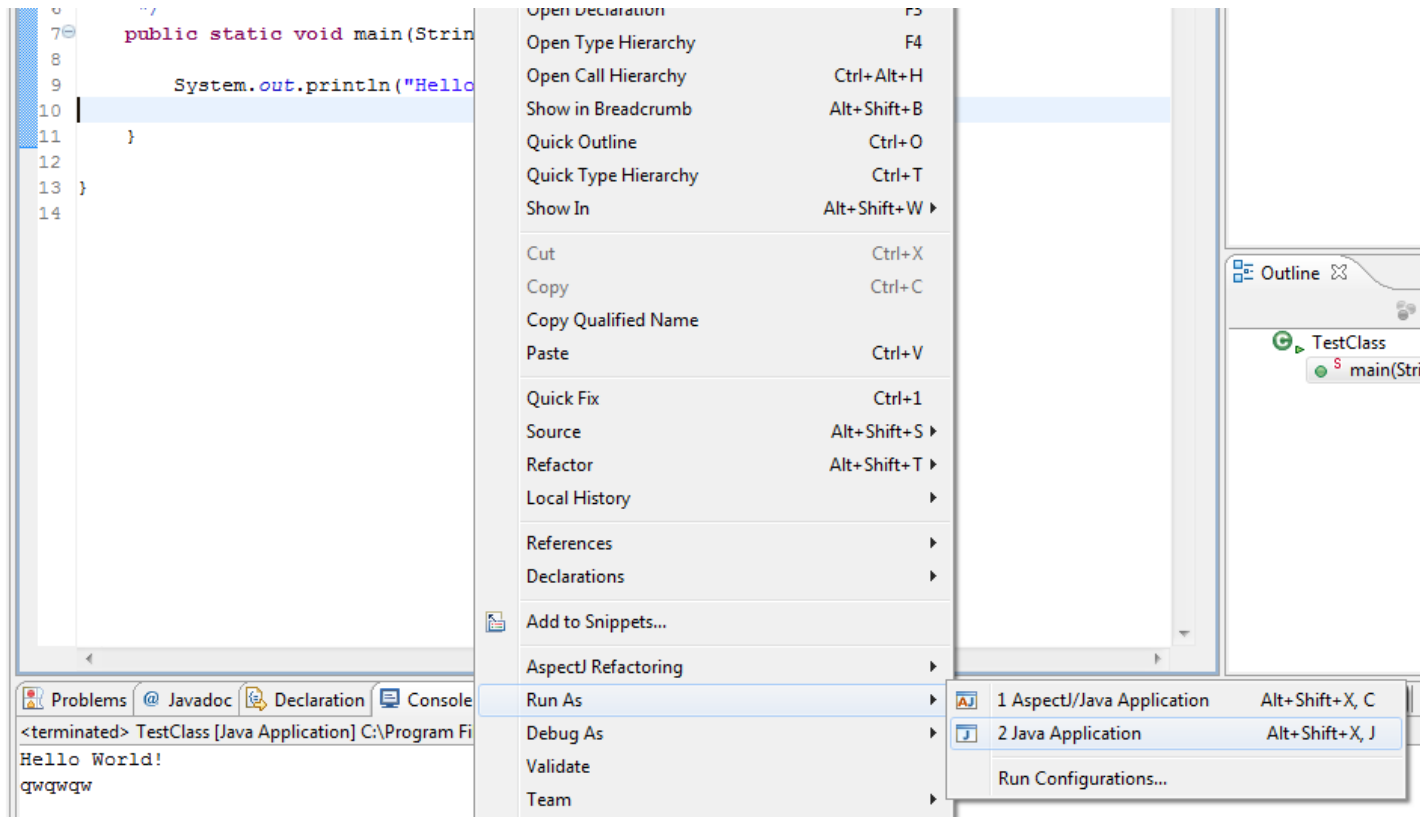
## Quellcode der erstellten Klasse

- Klassenname
- main-Methode, die automatisch erstellt wurde



```
1 |
2 | public class TestClass {
3 |
4 |     /**
5 |      * @param args
6 |      */
7 |     public static void main(String[] args) {
8 |         // TODO Auto-generated method stub
9 |
10 |    }
11 |
12 | }
13 |
```

- Java-Programme können auch direkt aus Eclipse gestartet werden (im Vergleich zum Command-Line-Aufruf)
- Rechtsklick auf die Klasse mit der Main-Methode, Run As > Java Application





## Code Conventions for the Java Programming Language

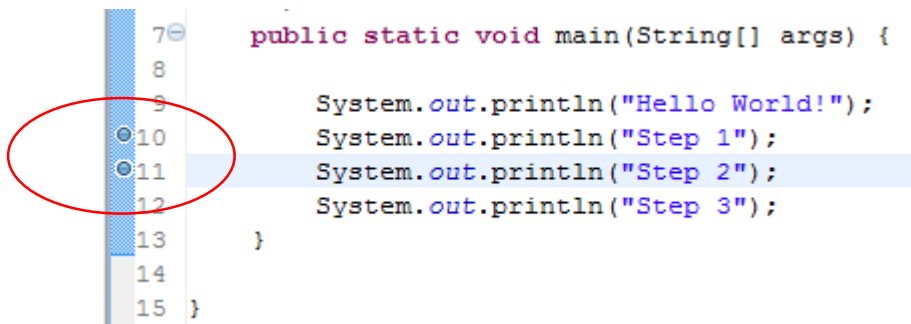
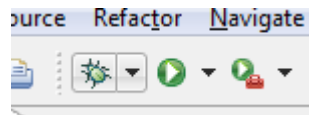
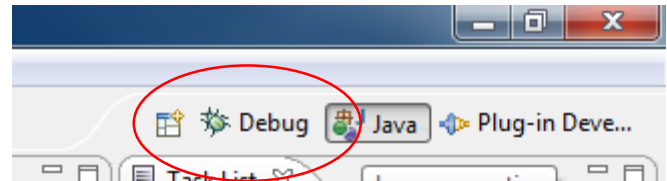
This *Code Conventions for the Java Programming Language* document contains the standard conventions that we at Sun follow and recommend that others follow. It covers filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices and includes a code example.

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

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

## Eclipse Programm debuggen

- „Schritt für Schritt“ durch ein Programm laufen und den Zustand des Programms verfolgen
- Debug-Perspektive
- Breakpoints



- Das „Hello World“ Programm in einem Editor programmieren (nicht in eclipse)
  - Programm in der Command-Line mit javac kompilieren
  - Programm in der Command-Line mit java ausführen
  - Das „Hello World“-Programm mit eclipse erstellen und ausführen
  - Eclipse:
    - Grundeinstellungen (UTF8 encoding setzen, line numbers)
    - Package Explorer
    - Problems-, Konsolen- und JavaDoc-View
    - run configurations (programm arguments), package explorer
    - Perspektiven: Java, Debug
  - Aufbau der Java API-Dokumentation
- 
- Lesen: Java-Buch Kapitel 2 und 3; Links aus der Vorlesung
  - Vorbereiten: Was versteht man unter Zeichencodierung bzw. Encoding:
    - <http://www.w3.org/International/questions/qa-what-is-encoding.de.php>
    - <http://www.w3.org/International/articles/definitions-characters/#unicode>