Java 7 (2011)

Lesbare Generics - Diamant-Operator

```
List<Integer> Numbers = new ArrayList<>()
Set <? extends Number> numbers = new HashSet<>();
List<?> numbers = new ArrayList<>();
```

Literale - Neue Zahlendarstellungen

```
int meter 10 000 000;
int mneter 10_000_000;
int pi = 3.123_111_121_3223
int bits = 0b_1010_1010_1010_1010;
```

Try-with-Resources

```
// Resourcen werden automatisch geschlossen!!! (Ohne Finally)
// Resourcen müssen das Interface "AutoCloseable" inpementieren

try (InputStream in = new FileInputStream("in.txt");
    OutputStream in = new FileOutputStream("out.txt"))
{
   ....
}
```

Multi-Catch

```
try
{
   //....
} catch (NamingException | IOException | SQLException ex)
{
   log(ex);
}
```

String in Switch

```
switch (monat)
{
   case "Januar"
   case "März"
      return 31;
   case "April"
      return 30;
   default:
      trow new IlligalArgumentException();
```

```
}

// Lieber "enum" verwenden (wg. Typsicherheit))

public enum MonthTyp { Januar, Februar, März, April };

switch (month). // Month ist vom Typ "MonthTyp"
{
   case Januar
   case Februar
      return 31;
   case Februar
      return 30;
   default:
      trow new IlligalArgumentException();
}
```

VarArgs

```
@SafeVarargs // Die Methode ist Typsicher !!!!
void funktion(List<String>... listOfNames)
{
   for (List<String> list : listOfNames)
   {
      //....
   }
}
```

NIO.2 Das neue Dateisystem-API von Java7 (Path & Files)

```
Path source = Paths.get("path", "to", "Source");
Path target = Paths.get("path", "to", "target");

Files.copy(source, target, StandardCopyOption.REPLACE_EXISTING);
Files.move(source, target);
Files.createDirectory(Paths.get("dir", "and", "subdir"));

List<String> lines = Files.readAllLines(Paths.get(filename), Charset.forName("UTF-8"));

Files.write(Paths.get(filename), lines, Charset.forName("UTF-8"));

String javaHome = System.getProperty("java.home");
Path pathToRTJar = Paths.get(javaHome, "lib", "rt.jar"));
try (FileSytsem jarFileSytsem = FileSystem.newFileSystem(pathToRTJar, null))
{
   //...
}
```

Nebenläufigkeit (Cuncurrency, Fork/Join, TranasferQueue, ConcurrentLinkedDeque)

- Fork/Join-Framework (Divide and Conquer - Teile und Herrsche)
- TransferQueue (Erzeuger ----> TransferQueue ----> Verbraucher)

Weitere Neurungen

- Numbus look and Feel (Swing - siehe Programme/Java/jdk1.7.0/demo/jfc/SwingSet3 --> Programm starten)
- javax.swing.JLayer & java.util.Locale.Category
- Objects (wichtige Methoden um keine Null-Afragen zu machen)

```
public void updateOrderJava7(String orderId)
{
   Objects.requireNonNull(orderId);
   Objects.requireNonNull(orderId, "Der Wert darf nicht Null sein!");
}
public String toTextJava7(Objects someObjects)
{
   return Objects.toString(someObjects); //<--- Es muss nicht mehr auf NULL-Abfragen
}
```

Links:

http://openjdk.java.net/projects/jdk7/features/
http://www.angelikalanger.com
https://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html
https://docs.oracle.com/javase/tutorial/index.html