1.Template Class

<?>
<? super [Type]>
<? extends [Type]>
<? extends [Type1] & [IType2] & [IType3]>

Generische Klasse mit Verwendung eines "Wildcards".
Alles in Ordnung.
Class<?> c = Class.forName("meinpackage.MeineKlasse");
MeineAnnotation a = c.getAnnotation(MeineAnnotation.class);

Raw-Klasse. Nicht kompilierbar.
Class c = Class.forName("meinpackage.MeineKlasse");
MeineAnnotation a = c.getAnnotation(MeineAnnotation.class);

```
public class Example_3_2_GenericMethod {
   static <T> int findIndex(List<T> l, T o) {
...
   }
   public static void main(String[] args) {
      List<Double> dd = new ArrayList<Double>();
      // Autoboxing converts primitive "double" values to "Double".
      dd.add(1.2);
      dd.add(2.3);
      dd.add(3.4);
      dd.add(4.5);
      int i = findIndex(dd, 3.4);
      System.out.println(i);
   }
}
```

```
public class GenerischesPaar <T>
{
 private T l;
 private T r;

 GenerischesPaar (T l, T r)
 {
   this.l = l;
   this.r = r;
 }


 public T get_l ()
 {
   return l;
 }
```

```
 public T get_r ()
 {
   return r;
 }

 public String toString ()
 {
   return "L: " + l + " | R: " + r;
 }
}


public class Template_class
{
 public static void main (String[] args)
 {
   GenerischesPaar<Socke> socken = new GenerischesPaar<>(new
                                   Socke(), new Socke());
   System.out.println(socken);

   GenerischesPaar<Ohrring> ohrringe = new GenerischesPaar<>(new
                                       Ohrring(), new Ohrring());
   System.out.println(ohrringe);

   System.out.println(socken.get_l().toString());

   GenerischesPaar<?> mix = new GenerischesPaar<>(new Socke(), new
                            Ohrring());
   // Wildcard <?> wichtig, weil unterschiedliche Klassen
   System.out.println(mix);
 }
}
```

---

2. Streams

Allgemeine Hinweise:

Sollen Dateien eingelesen / ausgegeben werden, dann immer den Pfad vom “src”-Ordner aus starten!
Bsp.: File f = new File(“src/aufgabe_1/input.txt”);

Geöffnete Streams wieder schließen!

Verschiedene Möglichkeiten zum Einlesen / Schreiben:

// Buffered Varianten sind nicht unbedingt nötig, kommt auf die Aufgabe drauf an

// 1
FileReader fr = new FileReader(“src/input.txt”);
BufferedReader br = new BufferedReader(fr);

FileWriter fw = new FileWriter(“src/output.txt”);
BufferedWriter bw = new BufferedWriter(fw);

// 2
FileInputStream fis = new FileInputStream(“src/input.txt”);
BufferedInputStream bis = new BufferedInputStream(fis);

FileOutputStream fos = new FileOutputStream(“src/output.txt”);
BufferedOutputStream bos = new BufferedOutputStream(fos);

// Noch mehr wie PushbackInputStream… Analog der Aufgabe folgen

Bsp:

Aufgabe hier war, den Wert bestimmter Bytes abzuändern.
Um die Stelle der Bytes (22-25 und 58-60) herauszufinden, eine extra Zählvariable i, die die Durchgänge mitzählt.
b ist das aktuelle Byte (als int-Repräsentation)

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class Streams
{
 public static void main (String[] args)
 {
   // einfach um zu zeigen, dass man auch n File in den Stream
      geben kann
   File f = new File("src/ba_in.bmp");

   try
   {
     FileInputStream fis = new FileInputStream(f);
    FileOutputStream fos = new FileOutputStream("src/ba_out.bmp");

   int b = fis.read();
   int i = 0;

   while (b != -1)
   {
```

```
      if (i == 22) { b = 110; }
      else if (i >= 23 && i <= 25 || i == 58 || i == 59) { b = 0; }
      else if (i == 60) { b = 255; }

      fos.write(b);
      b = fis.read();

      i++;
    }

    fis.close();
    fos.close();
  }
  catch (IOException e)
  {
    System.err.println("Aufgabe e: IOException is aufgetreten.");
  }
}
}
```

---

3. RegEx

Zum Testen + Hilfe (unten rechts): https://regex101.com/

Wichtigste Token:
[abc+#,_-] ← gültige Zeichen in eckige Klammern
[a-zA-Z0-9] ← Reichenweiten für gültige Zeichen
[^abc] ← ungültige Zeichen = in eckige Klammern mit ^ davor
\s ← Whitespaces
\d ← Zahlen
. ← Platzhalter/Wildcard für beliebiges Zeichen
\. ← Punkt mit Backslash escaped, um das Zeichen . zu matchen

Quantifier:
(nichts) ← Zeichen soll/darf GENAU einmal vorkommen
? ← Zeichen soll/darf vorkommen, muss aber nicht
+ ← Zeichen soll/darf MINDESTENS 1-mal (1 bis unendlich) vorkommen
* ← Zeichen soll/darf beliebig oft (0 bis unendlich) vorkommen

RegEx-Beispiele:

Beispiel-String: “Haifisch.”
[aeiou] ← matched beliebigen Umlaut (genau 1) → “a”, ”i”, “i”
[aeiou]+ ← matched beliebige Umlaute (min 1) → “ai”, “i”

.* ← matched beliebiges Zeichen (beliebig oft) → “Haifisch.”
\. ← matched den Punkt (genau 1) → “.”

Beispiel-String: “Jonas Handynummer: 0157 9876 5432 10.”
\d* ← matched Zahlen (beliebig oft) → “0157”, “9876”, “5432”, “10”
[^\d]* ← matched alle Nicht-Zahlen (gleich wie \D*) (beliebig oft) → “Jonas Handynummer: “, 3 Leerzeichen zwischen den Nummern, “.” am Ende
\s ← matched Whitespaces (genau 1) → die ganzen Leerzeichen da drin

String-Methoden:

test_string.split() ← Methode an bestehenden String schreiben, als Parameter wird ein RegEx-String übergeben und zurückgeliefert wird ein String Array
Optionaler zweiter Parameter: Limit (int) wie oft gesplitted werden darf

test_string.replaceAll() ← Methode an bestehenden String schreiben, als Parameter wird ein RegEx-String und ein String zum Ersetzen übergeben, zurückgeliefert wird ein String

test_string.replaceFirst() ← wie .replaceAll(), aber ersetzt nur den ersten Treffer anstatt alle

String.join() ← Methode an String-Klasse schreiben, als Parameter wird ein Delimiter (Zeichen was zwischen die Strings geschrieben werden soll) und eine String-Kollektion (Array, List, ArrayList…) übergeben, zurückgeliefert wird ein String

Konkrete Beispiele:

```
String test_string = “Jonas Handynummer: 0157 9876 5432 10.”;

// split an allen Whitespaces
String[] string_arr = test_string.split(“\s”);
// string_arr → [“Jonas”, “Handynummer:”, “0157”, “9876”, “5432”, “10.”]

// Ersetze eine Zeichenfolge, beginnend mit Nicht-0 und gefolgt von 3 Zahlen durch 4 X
String replaced_string = test_string.replaceAll(“[^0]\d\d\d”, “XXXX”);
// replace_string →  “Jonas Handynummer: 0157 XXXX XXXX 10.”

// [^0]\d{3} würde aufs Gleiche kommen

// Ersetze das erste Vorkommen von “Jonas” durch Steve
String first_replace = test_string.replaceFirst(“Jonas”, “Steve”);
// first_replace → “Steve Handynummer: 0157 9876 5432 10.”

// String-Array zu einem String zusammenfügen mit Leerzeichen dazwischen
String[] names = [“Anne”, “Jonas”, “Steve”];
String joined_names = String.join(“ und “, names); // beachte Leerzeichen vor und nach “und”
// joined_names → “Anne und Jonas und Steve”
```

4. Cloning

Bsp:

```
class Kopie implements Cloneable
{
   int x;
   public Kopie(int x)
{
      this.x = x;
   }
   // Methode zur Ausgabe des Attributes
   public void print()
   {
      System.out.println('x = '+x);
   }
   /* Diese Methode kann eine CloneNotSupportedException werfen */
   public Object clone() throws CloneNotSupportedException
   {
      /* Hier wird die Methode clone der Superklasse(in diesem Falle Object) aufgerufen. */
      return super.clone();
   }
}


public class KopieTest
{
   public static void main (String [ ] args)
   {
      Kopie ref1 = new Kopie(1);
      Kopie ref2 = null;
      try {
         /* Da die Methode clone den Datentyp Object zurückliefert,
            müssen wir eine Typumwandlung zu unserer Klasse Kopie durchführen */
         ref2 = (Kopie) ref1.clone();
      }
      catch(CloneNotSupportedException ex) {
         System.out.println("Das Kopieren dieses Objektes wird nicht unterstützt");
      }
     ...
   }
```

5.Dom4J

Bsp:

```
import org.dom4j.Document;
import org.dom4j.DocumentHelper;
import org.dom4j.Element;
import org.dom4j.Node;
import org.dom4j.io.OutputFormat;
import org.dom4j.io.SAXReader;
import org.dom4j.io.XMLWriter;

import java.io.File;
import java.io.IOException;
import java.util.List;


public class LauncherDOM4J
{
 public static void main (String[] args)
 {
   try
   {
     //Input File
     File inputFile = new File("src/xml/class.xml");

     SAXReader reader = new SAXReader();
     Document document = reader.read(inputFile);

     System.out.println("Wurzel Element : " +
                          document.getRootElement().getName());

     listStudents(document);
     createXML();
   }
   catch (Exception e)
   {
     e.printStackTrace();
   }
 }

 private static void createXML ()
 {
   try
   {
     Document document = DocumentHelper.createDocument();
     Element root = document.addElement("Schule");

     Element class1 = root.addElement("Klasse")
                          .addAttribute("Bezeichnung", "4a");
     Element class2 = root.addElement("Klasse")
                          .addAttribute("Bezeichnung", "3c");
```

```
// Zuweisung der Student-Elemente hier nicht noetig, da nichts
mehr mit den Elementen gemacht wird
Element student1 = class1.addElement("Schüler")
                         .addAttribute("Geschlecht", "Junge")
                         .addText("Christoph Clarsen");
Element student2 = class1.addElement("Schüler")
                         .addAttribute("Geschlecht", "Junge")
                         .addText("Theo Schleißer");
Element student3 = class2.addElement("Schüler")
                         .addAttribute("Geschlecht", "Mädchen")
                         .addText("Tina Langgenke");

     //Pretty print
     OutputFormat format = OutputFormat.createPrettyPrint();
     XMLWriter writer = new XMLWriter(System.out, format);
     writer.write(document);
   }
   catch (IOException e)
   {
     e.printStackTrace();
   }
 }

 public static void listStudents (Document doc)
 {
   try
   {
     List<Node> nodes = doc.selectNodes("/kurs/student");

     for (Node node : nodes)
     {
       System.out.println("\nCurrent Element : " +
                                    node.getName());

       System.out.println("Matrikelnummer : " +
                                    node.valueOf("@matrikel"));

       System.out.println("First Name : " +
                   node.selectSingleNode("firstname").getText());

       System.out.println("Last Name : " +
                     node.selectSingleNode("lastname").getText());

       System.out.println("Marks : " +
                       node.selectSingleNode("score").getText());
     }
   }
   catch (Exception e)
   {
     e.printStackTrace();
   }
 }
}
```