Exception handling

Bei der Ausführung von Code kann es zu Ausnahmesituationen kommen. Bekannte Beispiele sind der Zugiff auf ein Arrayelement, das außerhalb des Indexbereiches liegt, Nulldivision, Ein-/Ausgabefehler ect..

In c musste man mit bedingten Anweisungen (if-Anweisung) derlei Situationen im Vorfeld abfragen.

```
FILE pf=NULL;
if (argc!=2) {fprintf(stderr, "usage %s <file>\n",argv[0]); exit(-1);}
else
{
  pf=fopen(argv[1]);
  if (pf==NULL){fprintf(stderr, "error opening %s\n",argv[1]); exit(-1);}
  else
  {
    . . .
    fclose(pf);
  }
}
```

- Java bietet dafür, wie auch andere objektorientierte Sprachen, Exceptionhandling
- Exception: Ausnahme- /Fehlersituation
- Dabei wird Code in dem Exceptions auftreten können, in sogenannte try -Blöcke gesetzt.
- Kommt es nun zu einer Ausnahmesituation, z.Bsp. durch eine Indexverletzung bei einem Arrayzugriff, so wird eine Exception geworfen.
- Über eine oder mehere catch-Klauseln (to catch : fangen) kann nun eine geworfene Exception aufgefangen werden.

Try / catch

```
class tryDemo
{
  public static void main(String args[])
  {
    int array[]={2,4,6,8,10,12};
    int i=0;
    try
    {
        while(true)
        {
          System.out.printf("array[%d]: %d\n",i, array[i]);
          i++;
        }
    }catch (IndexOutOfBoundsException ex1)
    {
       System.out.println("Exception ist aufgetreten");
    }
    System.out.println("Das Programm geht weiter, es gibt keinen Absturz");
  }
}
```

catch

- Auf einen try-Block können mehrere catch-Klauseln, die unterschiedliche Exceptions auffangen, folgen.
- Eine Klausel catch(Exception e) fängt alle Exceptions auf und sollte entweder allein oder als letzte catch-Klausel stehen.
- Im einfachsten Falle gibt die catch-Klausel etwas aus oder bleibt leer.

catch

- Das Exceptionobjekt selbst kann auch ausgegeben werden.
- Das wird oft verbunden mit einem Aufruf der Methode printStackTrace(), die die Aufruffolge der Funktionen bis hin zur Exception ausgibt.

```
try
    {
    . . .
    }catch (IndexOutOfBoundsException ex1)
    {
       System.out.println("Exception: "+ ex1);
       Ex1.printStackTrace();
    }
```

Eingebaut in obiges Beispiel wird die angegebene Ausgabe erzeugt. Dabei ist 6 der Index der zur Ausnahmesituation geführt hat.

```java
class TryDemo
{
  public static void main(String args[])
  {
    int array[]={2,4,6,8,10,12};
    int i=0;
    try
    {
        while(true)
        {
          System.out.printf("array[%d]: %d\n",i, array[i]);
          i++;
        }
    }catch (IndexOutOfBoundsException ex1)
    {
       System.out.println("Exception: "+ ex1);
       ex1.printStackTrace();
    }
    System.out.println("Das Programm geht weiter, es gibt keinen Absturz");
  }
}
```

Nebenstehendes Beispiel erzeugt die angegebene Ausgabe. Dabei ist 6 der Index der zur Ausnahmesituation geführt hat.

```
array[0]: 2
array[1]: 4
array[2]: 6
array[3]: 8
array[4]: 10
array[5]: 12
Exception: java.lang.ArrayIndexOutOfBoundsException: 6
java.lang.ArrayIndexOutOfBoundsException: 6
        at TryDemo.main(TryDemo.java:11)
Das Programm geht weiter, es gibt keinen Absturz
```

Exceptions / Funktionen

- In der Java Classlibrary gibt es viele Funktionen, die u. Ust. Exceptions werfen.
- Werden solche Funktionen verwendet, verpflichtet Java den Aufrufer, diese Funktionensaufrufe in try/catch zu kapseln.
- Der Compiler erzeugt ggf. Fehlermeldungen.

Exceptions / Funktionen

- Will oder kann man das Exceptionhandling innerhalb einer Funktion nicht ausprogrammieren, das kann ganz unterschiedliche Gründe haben, können Exceptions weitergeleitet werden.
- throws IOException, angegeben hinter der Parameterliste einer Funktion, bewirkt, dass eine IOException, die in der Funktion auftritt, an den Aufrufer weitergeleitet wird.
- Damit muss der Aufruf dieser Funktion dann in einem try-Block geschehen.
- Die Angabe von throws Exception hinter der main-Funktion leitet alle Exceptions von main an die Virtuelle Maschine weiter.

```
public static void main(String args[]) throws Exception
{
. . .
```

```
class ThrowsDemo1
{
 public static void main(String args[])throws Exception
  {
    int array[]={2,4,6,8,10,12};
    int i=0;
    {
        while(true)
        {
          System.out.printf("array[%d]: %d\n",i, array[i]);
          i++;
        }
    }
  }
}
```

Exception wird an die Virtuellem Maschine (VM) weitergeleitet, diese beendet das Programm mit einer Fehlermeldung.

Exceptions werfen

```
class ThrowsDemo2
{
 public static void main(String args[])throws Exception
  {
    int array[]={2,4,6,8,10,12};
    int i=0;
    {
        while(true)
        {
          System.out.printf("array[%d]: %d\n",i, array[i]);
          i++;
          if (i==array.length) throw new Exception("index exception: "+i+" max allowed: "+array.length);
        }
    }
  }
}
```

Über
`throw (new Exception("message"));`
Kann eine Exception geworfen (signalisiert) werden.

```
array[0]: 2
array[1]: 4
array[2]: 6
array[3]: 8
array[4]: 10
array[5]: 12
Exception in thread "main" java.lang.Exception: index exception: 6 max allowed: 6
        at ThowsDemo.main(ThowsDemo.java:13)
```