II.4. Erweiterungen von Klassen und fortgeschrittene Konzepte

- 1. Unterklassen und Vererbung
- 2. Abstrakte Klassen und Interfaces
- 3. Modularität und Pakete
- **4. Ausnahmen (Exceptions)**

Ausnahmen (Exceptions)

Treten auf, wenn zur Laufzeit semantische Restriktionen nicht erfüllt werden, z.B.

- **`Arithmetische Ausnahmen:`**

  z.B. Division durch 0, Wurzel aus negativer Zahl, Overflow

- **Unzulässiger Zugriff auf Datenstrukturen:**

  z.B. Zugriff auf array-element mit negativem Index, oder index größer als length()-1.

  z.B. Zugriff auf ein Objekt über null-Verweis.

- **`Infrastrukturelle Ausnahmen:`**

  z.B. Lesen aus einer Datei, die nicht existiert

  z.B. Einlesen eines Stringes, der nicht in ein gewünschtes Datenformat konvertiert werden kann.

Exception Handling

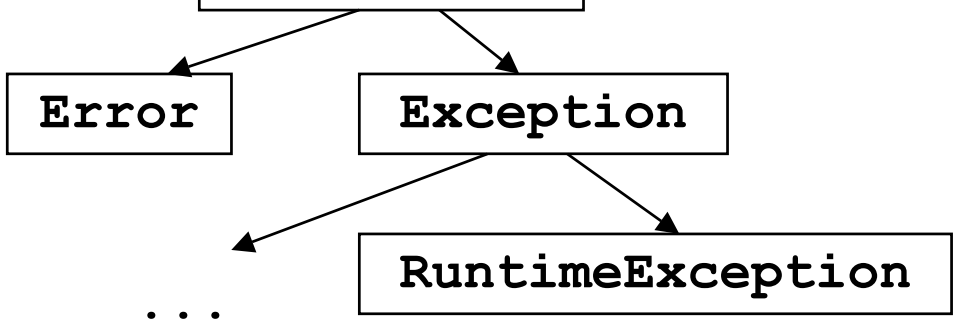
- ***Wenn in einem Programmblock Ausnahmen auftreten, wird ein exception handler aufgerufen***

```
try { ... Normalblock ...}
catch (AusnahmeArt1 Parameter1)
  { .. Ausnahmehandler1 .. }
catch (AusnahmeArt2 Parameter2)
  { .. Ausnahmehandler2 .. }
...
finally { .. Abschliessende Anweisungen ..}
```

- Bei Auftreten einer Ausnahme im Normalblock wird zu dem entsprechenden exception handler gesprungen.
- Der **`finally`** – Block ist optional und wird auf jeden Fall am Ende ausgeführt.

Exception Objekte

Eine Ausnahme ist ein Objekt, das aus einer Unterklasse der Klasse **`Throwable`** erzeugt wird (z.B. Error, Exception)



- Ausnahmeobjekte werden implizit erzeugt, wenn eine Ausnahme auftritt.
- Ein Ausnahmeobjekt steht dem entsprechenden error handler wie ein aktueller Parameter zur Verfügung.
- **`Throwable`** stellt z.B. die Methoden **`getMessage()`**, **`printStackTrace()`**, **`toString()`** zur Verfügung.

Beispiele von Exception Klassen

- **`IOException`**
  Fehler in Ein- oder Ausgabe
- **`ArithmeticException`**
  z.B. x/0 für int x
- **`ArrayIndexOutOfBoundsException`**
  Überschreiten des Indexbereiches eines arrays
- **`StringIndexOutOfBoundsException`**
  Versuchter Zugriff auf Zeichen eines Strings
- **`NumberFormatException`**
  Versuch, String, der keine gültige Zahl enthält, in Zahl umzuwandeln.
- **`NullPointerException`**
  Versuch, auf Objektvariable über null-Verweis zuzugreifen.

Beispiel für I/O Exception Handling

```
public static Liste LeseListeausDatei () {
    int Zeilennummer = 0; Liste ergebnis = new Liste();
    String inputfile = "d:II44_Exceptions/listeninput", line = null;
    BufferedReader reader = null;
    try { reader = new BufferedReader(new FileReader(inputfile));
      Zeilennummer++; line = reader.readLine();
      while (line != null) {
        ergebnis.sortiere_ein(Integer.parseInt(line));
        Zeilennummer++; line = reader.readLine();}
      reader.close();}
    catch (FileNotFoundException e)
      {System.out.println(e);}
    catch (IOException io) {System.out.println (io);}
    catch (NumberFormatException a){
      System.out.println(" Fehler beim Lesen in Zeile " +
        Zeilennummer + ". \n" + a + "\n" +
        "Weiterer Input aus Datei "+inputfile+" ignoriert");}
    finally { if (line != null) {
        try {reader.close();}
         catch (IOException io){System.out.println(io);}}
    } return ergebnis;}
```

Wo werden Exceptions behandelt

```
int M1()
{ .. M2(); .. // Hier wird eine Exception vom Typ A erzeugt.
..}

int M2()
{ try { .. M3(); ..}
  catch (A a) { .. } }

int M3()
{ .. M4(); ..}

int M4()
{ try { .. M2(); .. // Hier wird eine Exception vom Typ A erzeugt.
 ..}
  catch (B b) { .. } }
```

- **Aufruf von M4 führt zu Exception A. Diese wird im Aufruf von M2 abgehandelt.**
- **Exception A im Aufruf von M1 wird vom Laufzeitsystem abgehandelt.**

Benutzerdefinierte Exceptions

```
class kaputt extends java.lang.Throwable
{...}
```

```
class andereclass
{ ...
  void M1() {
      try { .. M2(); ..}
      catch (kaputt k)
          { .. // Ausnahmehandler für kaputt
           }
  }
  void M2() throws kaputt {
       ... M3(); ..
  }
  void M3() throws kaputt {
       ...
       if ( .. Fehlerbedingung ..) throw new kaputt();
       ...
  }
}
```