Exceptions: Parameterprüfung

```
import java.io.IOException;

public class Parameters
{

public static String Methode(String str,Exception obj,int index,
                             String[] array)
{
    if(str==null)
        throw new NullPointerException("Par. str darf nicht null sein");
    if(str.length()==0)
        throw new IllegalArgumentException(
                           "Parameter str darf kein leerer String sein");
    if(obj==null)
        throw new NullPointerException("Par. obj darf nicht null sein");
    if(!(obj instanceof IOException))
        throw new IllegalArgumentException("Parameter obj muß ein String
                                    oder eine Subklasse von String sein");
    if(index<0 || index>99)
        throw new IllegalArgumentException("Parameter index Muß im
                                           Bereich von 0 bis 99 liegen");
    if(array==null)
        throw new NullPointerException("Par. array darf nicht null sein");
    if(array.length==0)
       throw new IllegalArgumentException("Par. array muß El. enthalten");
```

```
    for(int i=0;i<array.length;i++)
        if(array[i]==null)
            throw new NullPointerException("Das Element am Index "+i+
                                " des Parameter array darf nicht null sein");
    if(array.length<=index)
        throw new IllegalArgumentException("Parameter index referenziert
                                ein Element außerhalb des Parameters array");
    // Alle Parameter geprüft
    return "Ergebnis: "+str+" "+obj.toString()+" "+index*5+
                        " "+array[index];
}

static public void main(String args[])
{
    String ergebnis=null;
    // True solange Berechnung dauert (z. B. für Multithreading)
    boolean inBerechnung=false;
    String[] arr={"Element1","Element2","Element3"};
    System.out.println("Programmstart\n\n");
    try
    {
        inBerechnung=true;
        ergebnis=Methode("",null,-5,null);
        inBerechnung=false;
    }
```

```
    catch(Exception e)
    {
        ergebnis="Fehler: "+e.getMessage();
    }
    if(inBerechnung)
        System.out.println("Noch in Berechnung");
    System.out.println("Ergebnis 1: "+ergebnis+"\n");
    ergebnis=null;
    inBerechnung=false;
    try
    {
        inBerechnung=true;
        ergebnis=Methode("String",new java.io.EOFException(),3,arr);
    }
    catch(Exception e)
    {
        ergebnis="Fehler: "+e.getMessage();
    }
    finally
    {
        inBerechnung=false;
    }
    if(inBerechnung)
        System.out.println("Noch in Berechnung");
    System.out.println("Ergebnis 2: "+ergebnis+"\n");
```

```
    ergebnis=null;
    inBerechnung=false;
    try {
        inBerechnung=true;
        ergebnis=Methode("String",new java.io.EOFException(),2,arr);
    }
    catch(Exception e) {
        ergebnis="Fehler: "+e.getMessage();
    }
    finally {
        inBerechnung=false;
    }
    if(inBerechnung)
        System.out.println("Noch in Berechnung");
    System.out.println("Ergebnis 3: "+ergebnis+"\n");
    System.out.println("\nProgrammende");
}

}
```

Testausdruck:

```
Programmstart


Noch in Berechnung
Ergebnis 1: Fehler: Parameter str darf kein leerer String sein

Ergebnis 2: Fehler: Parameter index referenziert Element außerhalb des
Parameters array

Ergebnis 3: Ergebnis: String java.io.EOFException 10 Element3


Programmende
```

Exception: ReRead

```
import java.io.*;

public class ReRead
{

public static void berechnen(long wert) throws WertAusserBereichException
{
    if(wert<0)
        throw new WertAusserBereichException("Der Wert ist zu klein.
                                              Erlaubter Bereich: 0 - 99");
    if(wert>99)
        throw new WertAusserBereichException("Der Wert ist zu groß.
                                              Erlaubter Bereich: 0 - 99");
    // Hier Wert verarbeiten
}

static public void main(String args[])
{
    System.out.println("Programmstart \n");
    boolean erfolgreich=false;
    BufferedReader in=new BufferedReader(
                                          new InputStreamReader(System.in));
```

```
    do{
        try {
            System.out.print("Bitte Wert eingeben: ");
            // Wert einlesen, umwandeln und gleich verarbeiten
            berechnen(Long.parseLong(in.readLine()));
            erfolgreich=true;
        }
        catch(WertAusserBereichException e) {
            System.out.println(e.getMessage());
        }
        catch(NumberFormatException e) {
            System.out.println("Sie haben keine Zahl eingegeben!");
        }
        catch(IOException e) {
            System.out.println("Fehler beim lesen: "+e.getMessage());
            // Würde erlauben, weiter außen noch aufzuräumen
            // (z. B. in finally)
            throw new RuntimeException("Nicht behebbarer Fehler!
                                        Programm wird abgebrochen.");
            // Oder anders: System.exit(1);
        }
    }while(!erfolgreich);
    System.out.println("\nProgrammende");
}

}
```

```
class WertAusserBereichException extends Exception
{
    public WertAusserBereichException(String msg)
    {
        super(msg);
    }

    public WertAusserBereichException()
    {
        super();
    }
}
```

Testausdruck:

```
Programmstart

Bitte Wert eingeben: Wert
Sie haben keine Zahl eingegeben!
Bitte Wert eingeben: -5
Der Wert ist zu klein. Erlaubter Bereich: 0 - 99
Bitte Wert eingeben: 100
Der Wert ist zu groß. Erlaubter Bereich: 0 - 99
Bitte Wert eingeben: 13

Programmende
```

Programm-Argumente lesen und prüfen

```
import java.util.Vector;
import java.util.Enumeration;

public class Arguments
{

static String dateiname;
static int startIndex;
static boolean binaer=false;
static boolean trace=false;
static Vector suchstrings=null;

/* "throws IllegalArgumentException" ist nicht nötig, da es sich hierbei
um eine Subklasse von RuntimeException handelt -> unchecked */
static protected void parseParameter(String args[])
{
    int nextIndex=0;
    if(args.length<2)
        throw new IllegalArgumentException("Zu wenige Parameter!");
    dateiname=args[nextIndex++];
    try
    {
        startIndex=Integer.parseInt(args[nextIndex++]);
    }
```

```
    catch(NumberFormatException e) {
        // Hier nötig, da wir im Hauptprogramm nur
        // IllegalArgumentException abfangen.
        // Alternative: Dort auch NumberFormatException prüfen
        throw new IllegalArgumentException("Startindex ungültig: "+
                                           e.toString());
    }
    // Vorsicht: Es müssen folgende Kombinationen akzeptiert werden:
    // -b -t, -t -b, -tb, -bt, -b, -t sowie die selben mit / statt -
    while(nextIndex<args.length && (args[nextIndex].startsWith("-")
          || args[nextIndex].startsWith("/")))
    {
        String param=args[nextIndex].substring(1);
        while(param.length()>0) {
            String option=param.substring(0,1);
            if(option.equalsIgnoreCase("b"))
                binaer=true;
            else if(option.equalsIgnoreCase("t"))
                trace=true;
            else
                throw new IllegalArgumentException("Option "+option+
                                                   " ist unbekannt!");
            param=param.substring(1);
        }
        nextIndex++;
    }
```

```
    // Die restlichen Argumente sind die Suchstrings.
    // Gibt es keine, soll der Vector nicht angelegt werden,
    // sondern null sein
    while(nextIndex<args.length)
    {
        if(suchstrings==null)
            suchstrings=new Vector();
        suchstrings.addElement(args[nextIndex++]);
    }
}

static public void main(String args[])
{
    System.out.print("Übergebene Parameter als Gesamt-String:");
    for(int i=0;i<args.length;i++)
        System.out.print(" "+args[i]);
    System.out.println();
    try
    {
        parseParameter(args);
    }
    catch(IllegalArgumentException e)
    {
        System.out.println(e.getMessage());
        Usage();
    }
```

```
    System.out.println("Verarbeitung der Daten mit folgenden Werten:");
    System.out.println("Dateiname: "+dateiname);
    System.out.println("Startindex: "+startIndex);
    if(binaer)
        System.out.println("Binaer verarbeiten");
    if(trace)
        System.out.println("Trace ausgeben");
    if(suchstrings==null)
        System.out.println("Keine Suchstrings angegeben");
    else
    {
        int pos=1;
        for(Enumeration e=suchstrings.elements();e.hasMoreElements();)
            System.out.println("Suchstring "+(pos++)+": "+
                               (String)e.nextElement());
    }
}

private static void Usage()
{
    System.out.println("Aufruf: java Arguments dateiname startindex [-b]
                        [-t] {suchstring}");
    System.exit(1);
}

}
```

Testausdruck:

```
Übergebene Parameter als Gesamt-String: test.txt 23 -bt Wann Wo
Verarbeitung der Daten mit folgenden Werten:
Dateiname: test.txt
Startindex: 23
Binaer verarbeiten
Trace ausgeben
Suchstring 1: Wann
Suchstring 2: Wo
```

Die Klasse File

```
import java.io.*;
import java.util.Date;

public class Datei
{

static public void main(String args[])
{
    if(args.length!=1 || args[0].length()==0)
    {
        System.out.println("Aufruf: java Datei Dateiname");
        System.exit(1);
    }
    File f=new File(args[0]);
    if(!f.exists())
    {   // datei/Verzeichnis existiert nicht
        System.out.println("Die Datei "+args[0]+" exitsiert nicht.");
        System.exit(1);
    }
    // Vorsicht: Kann nicht immer in 1:1 in ein Datum umgewandelt
    // werden (keine Garantie!)
    System.out.println("Zuletzt verändert: "+new Date(f.lastModified()));
```

```
/* Diese beiden Funktionen sind mit Vorsicht zu genießen!
   Werden / und \ gemischt verwendet, ergeben sich falsche Werte.
   Der Name ist nur dann richtig, wenn der letzte Pfad-Separator
   ein \ ist (genauer: der Variable separatorChar entspricht) */
// Pfad: Directories+Dateiname
// Name: Nur Dateiname
System.out.println("Pfad: "+f.getPath()+"\nName: "+f.getName());
// Die Länge eines Verzeichnisses ist 0!
System.out.println("Länge: "+f.length());
if(f.isFile())
{
    // Der kanonische Pfad kann einen Zugriff bedingen
    // und daher fehlschlagen!
    try
    {
        System.out.println("Der kanonische Name der Datei ist "+
                           f.getCanonicalPath());
    }
    catch(IOException e)
    {
        System.out.println("Exception bei getCanonicalPath: "+
                           e.getMessage());
        System.exit(1);
    }
    if(!f.canRead())
        System.out.println("Die Datei kann nicht gelesen werden");
```

```
        if(!f.canWrite())
            System.out.println("Die Datei kann nicht geschrieben werden");
    }
    else if(f.isDirectory())
    {
        try
        {
            System.out.println("Der kanonische Name des Verzeichnisses
                                ist "+f.getCanonicalPath());
        }
        catch(IOException e)
        {
            System.out.println("Exception bei getCanonicalPath: "+
                                e.getMessage());
            System.exit(1);
        }
        if(!f.canRead())
            System.out.println("Das Verzeichnis kann nicht
                                gelesen werden");
        if(!f.canWrite())
            System.out.println("Das Verzeichnis kann nicht
                                geschrieben werden");
```

```
        // Verzeichnis-Inhalt ausgeben: Wird als Array von
        // Strings (Dateiname) geliefert
        System.out.println("Inhalt des Verzeichnisses: ");
        String[] dateiliste=f.list();
        for(int i=0;i<dateiliste.length-1;i++)
            System.out.print(dateiliste[i]+", ");
        System.out.println(dateiliste[dateiliste.length-1]);
    }
    else
    {
        // Beispiel: Unter Windows NT wenn auf Verzeichnis und Datei
        // alle außer Read-Rechten bestehen
        System.out.println("Seltsam: Weder Datei noch Verzeichnis???");
    }
}

}
```

Testausdruck:

```
Zuletzt verändert: Thu Jul 02 14:41:22 CEST 1998
Pfad: ..\..\Strings/License.txt
Name: Strings/License.txt
Länge: 3781
Der kanonische Name der Datei ist C:\Java\lva\Strings\License.txt


Zuletzt verändert: Thu Feb 25 14:33:49 CET 1999
Pfad: ../Datei
Name: ../Datei
Länge: 0
Der kanonische Name des Verzeichnisses ist C:\Java\lva\Exceptions\Datei
Inhalt des Verzeichnisses:
Datei.cdb, Datei.class, Datei.java, Datei.ve2, Datei.vep, Datei.vpj,
License.txt
```

Datei License.txt und Verzeichnis SplitFile: Rechte "WXDPO"

```
Zuletzt verändert: Thu Jan 01 01:00:00 CET 1970
Pfad: ../SplitFile\License.txt
Name: License.txt
Länge: 0
Seltsam: Weder Datei noch Verzeichnis???
```

Verzeichnisbaum

```
import java.io.*;

public class VerzeichnisBaum
{

private static long verzeichnisAnzahl=0;

// Ein Verzeichnis mit allen seinen Unterverzeichnissen ausgeben
private static void listDirectory(File verz,int tiefe)
{
    verzeichnisAnzahl++;
    // Richtig eingerückt den Namen ausgeben
    for(int i=0;i<tiefe;i++)
        System.out.print("  ");
    System.out.println(verz.getName());
    // Alle Verzeichnisinhalte durchgehen
    String[] subVerz=verz.list();
    if(subVerz!=null && subVerz.length>0)
    {
```

```
        for(int i=0;i<subVerz.length;i++)
        {
            // Pfad muß zusammengehängt werden!
            File sub=new File(verz.getPath()+
                              File.separatorChar+subVerz[i]);

            if(sub.exists() && sub.canRead())
            {
                // Nur rekursiv aufrufen, wenn es ein Verzeichnis ist
                if(sub.isDirectory())
                    listDirectory(sub,tiefe+1);
            }
            else
                System.out.println("Unterverzeichnis "+subVerz[i]+
                                   " kann nicht gelesen werden!");
        }
    }
}

public static void main(String args[]) {
    // Parameter prüfen
    if(args.length!=1 || args[0].length()==0) {
        System.out.println("Aufruf: java VerzeichnisBaum
Startverzeichnis");
        System.exit(1);
    }
```

```
    // Sicherheitshalber / durch \ ersetzen
    File verz=new File(args[0].replace('/','\\'));
    // Prüfungen für Startverzeichnis müssen hier gemacht werden
    if(!verz.exists() || !verz.canRead())
    {
        System.out.println("Verzeichnis "+args[0]+" existiert nicht
                           oder kann nicht gelesen werden!");
        System.exit(1);
    }
    if(!verz.isDirectory())
    {
        System.out.println(args[0]+" ist kein Verzeichnis!");
        System.exit(1);
    }
    // Ausgeben
    System.out.println("Verzeichnisbaum:");
    listDirectory(verz,0);
    // Statistik ausgeben
    System.out.println("\nInsgesamt "+verzeichnisAnzahl+"
Verzeichnisse.\n");
}

}
```

Testausdruck:

```
Verzeichnisbaum:
Klassen
  Abenteuer
    Daten
      Objekte
      Orte
    Programm
      Kommandos
      Parser
  doc

Insgesamt 9 Verzeichnisse.
```

SplitFile

```
import java.io.*;
import java.lang.*;
import java.math.*;


/** SplitFile: Aufsplitten einer Datei in mehrere kleinere fixer Länge
Achtung: Hier wird alles und jedes gedondert mit try abgesichert:
Manchmal ist es besser, nicht so detaillierte Fehlermeldungen zu haben,
dafür aber einen einfacheren un übersichtlichen Code zu programmieren
-> Dies bedeutet NICHT, weniger zu prüfen, sondern NUR die try-Blöcke
weiter außen zusammenzufassen und als Detail die Exception zu verwenden!
@author Dipl.-Ing. Michael Sonntag
@version 1.0
*/
public class SplitFile {

public static void main(String[] args) {
    File quelle=null;  // Quelldatei
    File teil;         // Zieldatei
    BufferedInputStream von=null;     // Stream zum Einlesen
    BufferedOutputStream nach=null;   // Stream zum Ausgeben
    long blockgroesse=2048;   // Größe der einzelnen Dateien
    long restlaenge;          // Noch zu schreibende Bytes
    String quellpfad=null;    // Pfad und Name der Quelldatei (kan.)
    int teilNr;               // Nummer des aktuellen Teils
```

```
    System.out.println("SplitFile: Aufteilen von Dateien in kleinere
                        Stücke\n(c) 1999 Dipl.-Ing. Michael Sonntag\n");
    // Parameter prüfen
    if (args.length<1 || args.length>2 || args[0].length()<1)
    {
        System.out.println("Benutzung: java SplitFile dateiname
                            [dateigroesse]\nDie Zieldateinamen sind der
                            Quelldatiename mit .1, .2 angehängt. Die
                            Dateigröße der Zeildateien ist in Bytes
                            anzugeben (Standard: "+blockgroesse+
                            " Bytes).");
        System.exit(1);
    }
    // Quelldatei prüfen
    try
    {
        quelle=new File(args[0]);
        quellpfad=quelle.getCanonicalPath();
        if(args.length>1)
            blockgroesse=Long.parseLong(args[1]);
        else
            System.out.println("Standardwert von "+blockgroesse+
                               " als Blockgröße angenommen.");
    }
```

```
    catch(NumberFormatException e) {
        System.out.println("Ungültige Blockgröße!\nBenutzung: ……
        System.exit(1);
    }
    catch(IOException e) {
        System.out.println("Vollständiger Pfad kann nicht festgestellt
                            werden: "+args[0]+"!");
        System.exit(1);
    }
    /* Ist Quelldatei vorhanden, lesbar, eine Datei? */
    if (!quelle.exists())
    {
        System.out.println("Datei "+quellpfad+" existiert nicht!");
        System.exit(1);
    }
    if (!quelle.isFile())
    {
        System.out.println(quellpfad+" ist keine Datei!");
        System.exit(1);
    }
    if (!quelle.canRead())
    {
        System.out.println("Datei "+quellpfad+" kann nicht gelesen
                            werden!");
        System.exit(1);
    }
```

```
    teilNr=0;
    // Prüfen, ob überhaupt geteilt werden muß
    restlaenge=quelle.length();
    if(restlaenge<blockgroesse)
    {
        System.out.println("Die Datei ist kleiner als die Blockgröße
                            und wird daher nicht aufgeteilt.");
        System.exit(1);
    }
    try
    {
        von=new BufferedInputStream(new FileInputStream(quelle));
        while(restlaenge>0)
        {
            teilNr++;
            teil=new File(quellpfad+"."+teilNr);
            /* Ist Zieldatei noch nicht vorhanden und beschreibbar? */
            if (teil.exists())
            {
                System.out.println("Datei "+quellpfad+"."+teilNr+
                                   " existiert bereits!");
                System.exit(1);
            }
            /* Keine Prüfung teil.isFile() -> wenn sie noch nicht
               existiert (was sie ja hier sicher nicht tut), wird hier
               immer false zurückgeliefert */
```

```
            if (!quelle.canWrite())
            {
                System.out.println("Datei "+quellpfad+"."+teilNr+
                                   " kann nicht geschrieben werden!");
                System.exit(1);
            }
            try
            {
                nach=new BufferedOutputStream(new FileOutputStream(teil));
                System.out.println("Erstelle Teildatei "+teilNr+
                                   ": "+quellpfad+"."+teilNr);
                long count=0;
                // Eigentliche Kopierschleife
                while(count<blockgroesse && restlaenge>0)
                {
                    int nextByte=-1;
                    try
                    {   // Ein Byte einlesen
                        nextByte=von.read();
                    }
                    catch(IOException e)
                    {
                        System.out.println("Fehler beim Lesen der
                              Quelldatei "+quellpfad+": "+e.getMessage());
                        System.exit(1);
                    }
```

```
                if(nextByte<0)
                {   // Unerwartetes Dateiende
                    System.out.println("Unerwartetes Dateiende in "+
                                       quellpfad+"."+teilNr+".");
                    System.exit(1);
                }
                try
                {   // Ein Byte schreiben
                    nach.write(nextByte);
                }
                catch(IOException e)
                {
                    System.out.println("Fehler beim Schreiben der
        Teildatei "+teilNr+" "+quellpfad+"."+teilNr+": "+e.getMessage());
                    System.exit(1);
                }
                count++;
                restlaenge--;
            }
        }
        catch(IOException e)
        {   // Sollte nicht auftreten
            System.out.println("Zieldatei "+teilNr+": "+quellpfad+
                      " kann nicht erstellt werden: "+e.getMessage());
            System.exit(0);
        }
```

```
            finally
            {   // Zieldatei schließen
                try
                {
                    nach.close();
                }
                catch(IOException e)
                {   // Hier kann nichts mehr getan werden
                    System.out.println("Fehler beim Schließen der Datei
                                "+quellpfad+"."+teilNr+": "+e.getMessage());
                }
            }
        }
        System.out.println("Erfolgreich aufgeteilt: "+teilNr+" Teile\n");
    }
    catch(FileNotFoundException e)
    {   // Sollte nicht auftreten
        System.out.println("Quelldatei "+quellpfad+" nicht gefunden: "+
                           e.getMessage());
    }
```

```
    finally
    {   // Quelldatei schließen
        try
        {
            if(von!=null)
                von.close();
        }
        catch(IOException e)
        {   // Hier kann nichts mehr getan werden
            System.out.println("Fehler beim Schließen der Datei "+
                               quellpfad+": "+e.getMessage());
        }
    }
}

}
```

Testausdruck:

```
SplitFile: Aufteilen von Dateien in kleinere Stücke
(c) 1999 Dipl.-Ing. Michael Sonntag

Standardwert von 2048 als Blockgröße angenommen.
Erstelle Teildatei 1: C:\Java\lva\Exceptions\SplitFile\License.txt.1
Erstelle Teildatei 2: C:\Java\lva\Exceptions\SplitFile\License.txt.2
Erfolgreich aufgeteilt: 2 Teile
```

Mit 1024 als Parameter sofort danach

```
SplitFile: Aufteilen von Dateien in kleinere Stücke
(c) 1999 Dipl.-Ing. Michael Sonntag

Datei C:\Java\lva\Exceptions\SplitFile\License.txt.1 existiert bereits!
```

Mit 1024 als Parameter nach löschen der Dateien

```
SplitFile: Aufteilen von Dateien in kleinere Stücke
(c) 1999 Dipl.-Ing. Michael Sonntag

Erstelle Teildatei 1: C:\Java\lva\Exceptions\SplitFile\License.txt.1
Erstelle Teildatei 2: C:\Java\lva\Exceptions\SplitFile\License.txt.2
Erstelle Teildatei 3: C:\Java\lva\Exceptions\SplitFile\License.txt.3
Erstelle Teildatei 4: C:\Java\lva\Exceptions\SplitFile\License.txt.4
Erfolgreich aufgeteilt: 4 Teile
```