Exceptions

Quelle: https://michaelkipp.de/java/20B%20exceptions-io.html 11/2023

bearbeitet J. Rau 2023

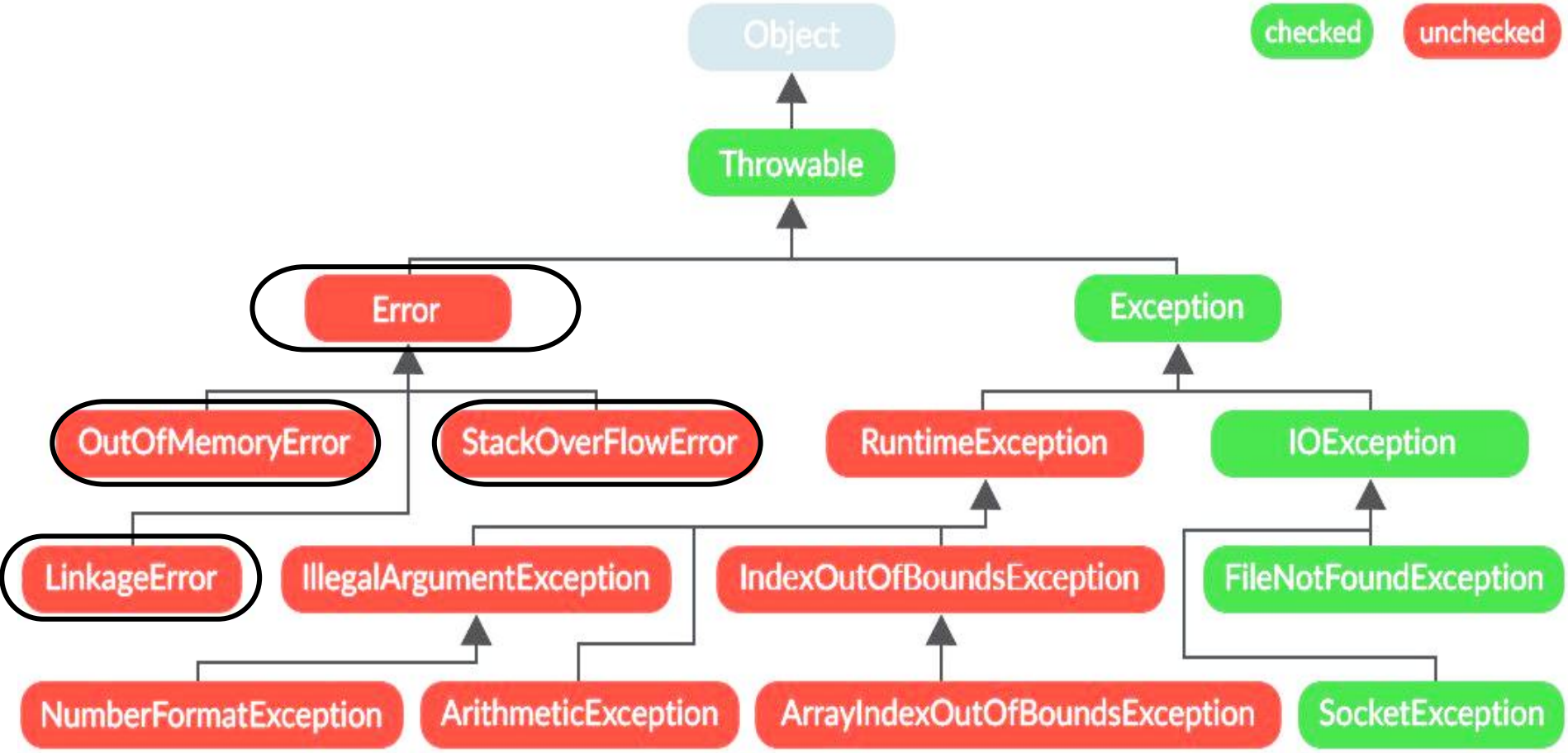
Ausnahmen und Errors

Ausnahmen und Fehler

Fehler (Error) in Java ist ein nicht reparierbarer Laufzeitfehler oder ein Hardware-Problem, das zum Absturz des Programms führt.

NOW WHO'S MADE THE "FATAL ERROR" ?

Ausnahmen (*Exceptions*) sind meistens keine eigentlichen Fehler, sondern unerwartete Fälle, auf die das Programm reagieren muss.



https://codegym.cc/groups/posts/exceptions-in-java

Errors

Sind schwerwiegende Probleme mit der Laufzeitumgebung, die mit ziemlicher Sicherheit nicht wiederherstellbar sind.
z.B. OutOfMemoryError, LinkageError und StackOverflowError

Sie bringen im Allgemeinen das Programm oder einen Teil des Programms zum Absturz.

Checked Exceptions

throws in der Methodensignatur nötig, Zwang zur Behandlung

Unchecked Exceptions

leiten von RuntimeException ab

keine throws in der Methodensignatur nötig, keinen Zwang zur Behandlung

Ausnahmen

Eine Ausnahme (Exception) ist ein Fehler oder ein nicht geplantes Ereignis, das während der Ausführung eines Programms vorkommt und dessen normalen Ablauf stört.

Wenn Programme nicht auf Ausnahmen reagieren können, führt dies zu den von den Anwendern gefürchteten Abstürzen.

In Java wurde die Behandlung von Ausnahmen in die Sprache integriert.
Java ermöglicht nach Fehlern eine "weiche" Landung.

Ausnahmebehandlung

Ausnahmen (***Exceptions***) sind unerwartet auftretende **Laufzeitfehler**.

Eine **Ausnahme** in Java ist ein **Objekt**, das eine Instanz der Klasse ***Throwable*** (oder einer ihrer Unterklassen) ist.

Beispiele:

- Division durch **0**
  (**ArithmeticException**)
- Lesen über Arraygrenzen hinweg
  (**IndexOutOfBoundsException**)
- Lesen über das Ende einer Datei hinaus
  (**EOFException**)

Ausnahmebehandlungsschema

Ausnahmen müssen grundsätzlich abgefangen oder weitergereicht werden.

`auslösen`

throw -Anweisung

Ein Ausnahme-Objekt wird innerhalb einer Methode erzeugt.

`abfangen`

try-catch -Anweisung

In der aufrufenden Methode muss die Ausnahme behandelt oder weitergeleitet werden.

`weiterreichen`

throws-Klausel

Ausnahmebehandlung

Die Ausnahme wird Bestandteil der Methoden-Signatur

Beispiel:

```
public Object suche ( Object x ) throws NotFoundException {
   . . .
}
```

Das bedeutet hier, dass innerhalb des Methodenrumpfs eine Ausnahme mit diesem Namen stattfinden kann.

Ein Ausnahme-Objekt der Klasse **NotFoundException** wird innerhalb der Methode erzeugt, falls das gesuchte Objekt nicht gefunden wird.

Geprüfte Exceptions am Beispiel Kontostand

```
public class Konto {
    double guthaben,betrag;

    public Konto() {
        guthaben= 100;
        betrag = InOut.readDouble("Betrag?...");
    }

    Nichts hindert uns daran, einen negativen Betrag "einzuzahlen":

        public void einzahlen() {
        guthaben =guthaben+betrag;
        ausgabe();
    }

    public void ausgabe() {
        System.out.println("Kontostand: " + guthaben);
    }

}
```

Fehler werfen

```
public void einzahlen()
    if (betrag < 0) {
    throw new Exception("Keinen negativen Betrag einzahlen: "+ betrag);};

    guthaben =guthaben+betrag;
    ausgabe();
}
```

Wir brauchen noch ein Signal, dass innerhalb dieser Methode eine Exception geworfen wird:

```
public void einzahlen() throws Exception{
    if (betrag < 0) {
    throw new Exception("Keinen negativen Betrag einzahlen: "+ betrag);};

    guthaben =guthaben+betrag;
    ausgabe();
}
```

Fehler fangen

Wenn diese Methode aufgerufen wird und einen möglichen Fehler auslöst, muss man entscheiden, wie mit dem Fehler umzugehen ist.

Eine Möglichkeit ist, den Fehler zu **fangen.**

Rufen wir die Methode einzahlen() ohne catch oder throw auf, gibt es eine Fehlermeldung:

```
unreported exception
java.lang.Exception; must be
caught or declared to be thrown
```

Der Compiler erlaubt den Aufruf nicht. → Der Fall, dass bei der Einzahlung etwas schiefläuft muss jetzt abgefangen werden.

Fehler fangen

```
public void foo() {

    ...

    try {

        ...

        boo();

        ...

    } catch (BooException e) {
        System.out.println("Boo!");
    }

    ...

}
```

Fehler!

Sprung in catch-Block

dann normal weiter...

Fehler fangen

```
try {
  CODE
}
catch (Exception e) {
  FEHLERFALL-CODE
}
```

```
public Konto_mit_Exc() {
    guthaben = 100;
    betrag = InOut.readDouble("Betrag?...");

    try {
        einzahlen();
    }
    catch (Exception e) {
        System.out.println("Fehler! " + e.getMessage());}
}
```

Eigene Exceptionklasse erstellen

```
public class Fehler extends Exception {
    static String meldung1="Negativer Betrag ist nicht erlaubt!";
    static String meldung2="Konto überzogen!";

public Fehler(String message) {
        super(message);
}

public static void checkBetrag(double betrag) throws Fehler {
        if (betrag<0) {
            throw new Fehler(meldung1);
        }
}

public static void checkGuthaben(double guthaben) throws Fehler {
        if (guthaben<0) {
            throw new Fehler(meldung2);
        }
}

}
```

```
public void abheben() throws Fehler{
    Fehler.checkBetrag(betrag);
    guthaben = guthaben-betrag;
    Fehler.checkGuthaben(guthaben);    ausgabe();
}

public void einzahlen() throws Fehler{
    Fehler.checkBetrag(betrag);
    guthaben = guthaben + betrag;       ausgabe();
}

public void kontoVerändern()
{
    String frage=InOut.readString ("Abheben? -->a, Einzahlen?-->e...");
    if (frage.equals("e")){
        try {
            einzahlen();
        }
        catch (Fehler e) {
            System.out.println("Achtung! " + e.getMessage());};
    }

    if (frage.equals("a")){
        try {
            abheben();}
        catch (Fehler e) {
            System.out.println("Achtung! " + e.getMessage());};
    }
```

Es gibt auch fertige Exceptions für häufig vorkommende Fehler.

Beispiel arithmetischer Fehler:
Division durch 0

```
try {
    double quotient= a / b;
    System.out.println("Ergebnis "+a+" : "+b+" = "+quotient);
}
catch (ArithmeticException e) {
    System.out.println("ArithmeticException => "+e.getMessage()+" nicht erlaubt!");
}
finally {
    ;}
    System.out.println("Finally block is always executed");
}
```

Beispiel RunTime-Fehler: Index Bound Exception

```
public void doSomething() {
    try
    { System.out.println(name.substring(5));
    }
    catch (IndexOutOfBoundsException e) {
        System.out.print("Die Zeichenkette ist zu kurz: " +e.getMessage());
    }
}
```

Beispiel RunTime-Fehler: NullPointer-Exception

```
try {
    System.out.println("Erster Buchstabe " + name.charAt(0));
}
catch(NullPointerException e) {
    System.out.println("NullPointerException!-Zeichenkette ist leer! -> "+e.getMessage());
}
```

Diese selbst erstellte Exception hast du schon verwendet!

```
public void losGehts( )  // Testmethode
{
    try {//Hier schreibst du zwischen die Klammern dein fertiges Programm
        karol.schneller();
        laufen();
        //treppe( );
        //legenUndsammeln( );
        //ziegelAufhebenUndmarke( ) ;
        //richtungUndWand( ) ;
        //liebesSchnulze()
    }
    catch (RuntimeException e) {
        karol.gibMeldungAus("Kann Auftrag nicht ausführen.");};
}
```

Beispiel Raketen- Exception

```
public class Countdown
{
    //  hier globale Variablen festlegen -->
    int wetter_perfekt;
    String treibstoff;
    boolean technik_OK;
    public Countdown()      // Konstruktor  -->
    {
        this.technik_OK = InOut.readBoolean("Technik ist ok? --> true oder false?..");
        this.treibstoff = InOut.readString( "Treibstoff? --> voll oder leer?..");
        this.wetter_perfekt = (int) (Math.random() * 2);
        eingabe_und_start( );
    }
```

Beispiel Raketen- Exception

```
public void eingabe_und_start( )  // Methoden -->
{
    try{
        Rakete r = new Rakete(technik_OK,treibstoff,wetter_perfekt);
        System.out.println("_____________________________\n");
        System.out.println("Countdown läuft!");
              }
    catch (RocketStartException e){
        System.out.println(e.getMessage());
    }
    finally
    {
        System.out.println("_____________________________\n");
        if (wetter_perfekt==0){System.out.println("Wetter ist prima!");}
        else {System.out.println("Das Wetter ist für den Start ungeeignet!");};
        if (technik_OK==true){System.out.println("Technik: alles ok!");} else
           {System.out.println("Technik: Ein Fehler ist aufgetreten!");}  ;
        System.out.println("Tank: "+treibstoff);
    }
}
```