

1. Grundbegriffe der Programmierung

2. Einfache Beispielprogramme

3. Datentypen und Variablen

4. Ausdrücke und Operatoren

5. Kontrollstrukturen

6. Blöcke und Methoden

7. Klassen und Objekte

8. Vererbung und Polymorphie

9. Pakete

10. Ausnahmebehandlung

11. Schnittstellen (Interfaces)

12. Geschachtelte Klassen

13. Ein-/Ausgabe und Streams

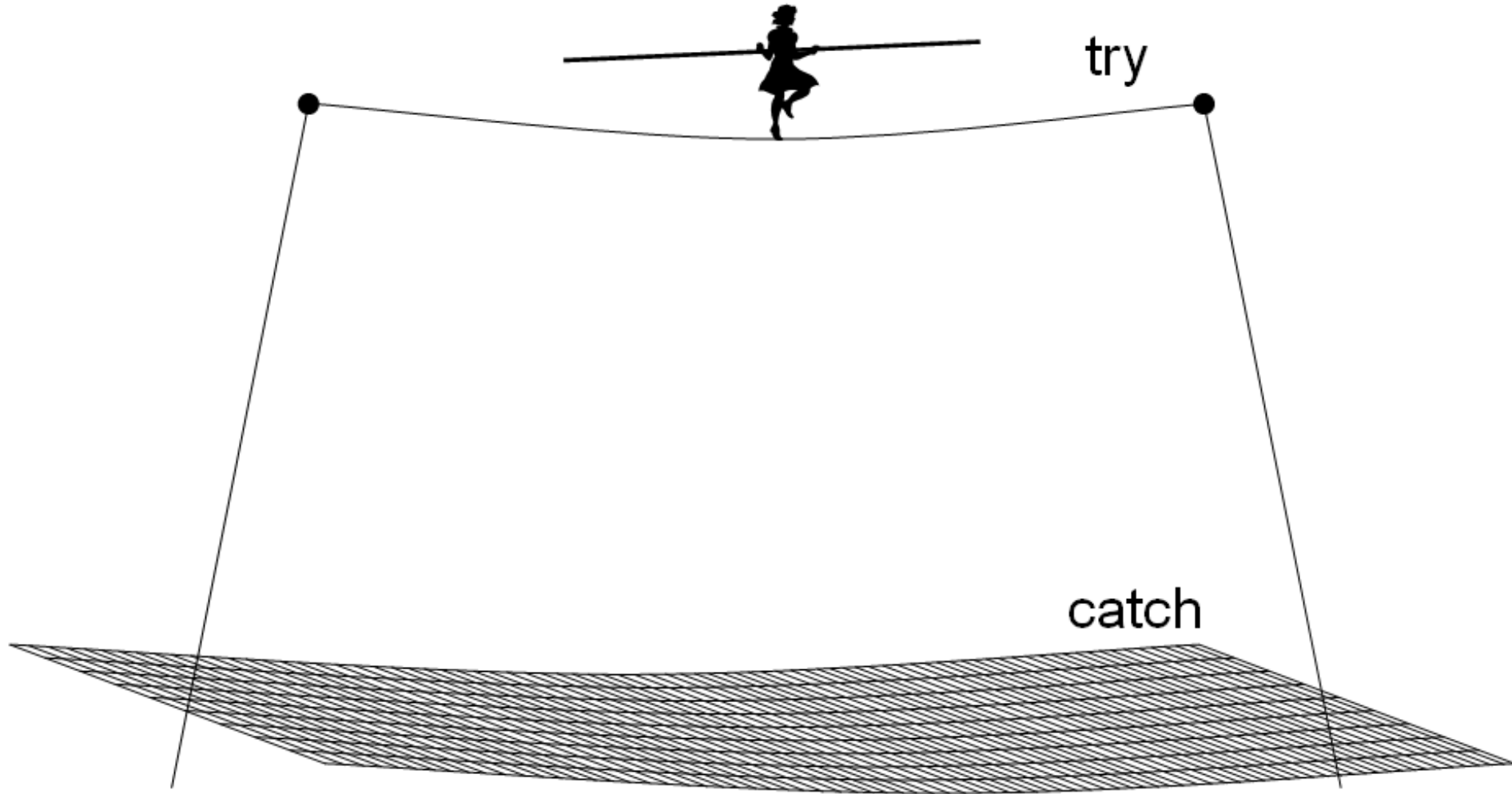
14. Applets / Oberflächenprogrammierung

Inhalte

✓ Konzept des Exception Handling

Implementierung

Ausnahmen



## Konzept des Exception Handlings I

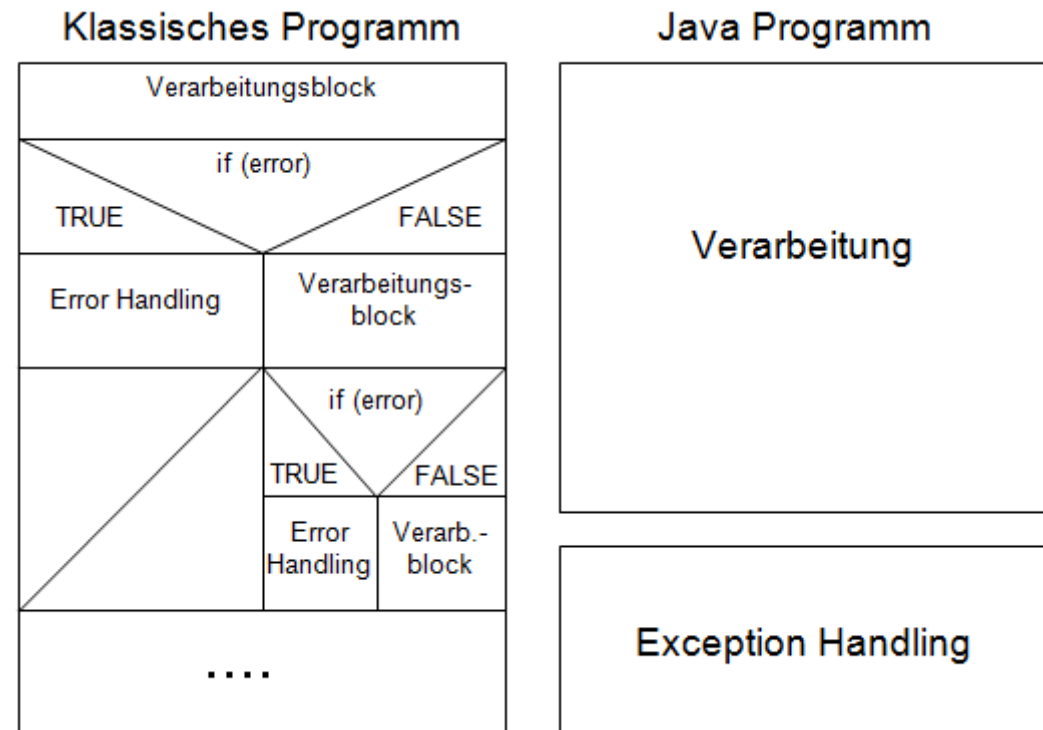
---

- Eine Exception (Ausnahme) ist ein abnormales Ereignis, das zu einem Laufzeitfehler führen kann
  
- Typische Quellen einer Exception:
  - Arithmetischer Überlauf
  - Mangel an Speicherplatz
  - Verletzung von Array Grenzen
  - Probleme bei der Ein und Ausgabe (z.B. Datei nicht vorhanden)
  - Es wird auf ein nichtvorhandenes Objekt zugegriffen (Nullpointer Exception)
  - etc.

## Konzept des Exception Handlings II

➤ Das Ziel des Exception Handling ist es, normalen und fehlerbehandelnden Code übersichtlich zu trennen und Ausnahmesituationen sicher zu trennen:

- Eventuelle Fehler, die vorhersehbar sind, abfangen
- Ein Exception Handler ist der Code, der auf solche Fehler adäquat reagiert.
- Das Programm kann sich von dem Fehler „erholen“, ohne dass es zu einem Programmabsturz kommt



1. Grundbegriffe der Programmierung

2. Einfache Beispielprogramme

3. Datentypen und Variablen

4. Ausdrücke und Operatoren

5. Kontrollstrukturen

6. Blöcke und Methoden

7. Klassen und Objekte

8. Vererbung und Polymorphie

9. Pakete

10. Ausnahmebehandlung

11. Schnittstellen (Interfaces)

12. Geschachtelte Klassen

13. Ein-/Ausgabe und Streams

14. Applets / Oberflächenprogrammierung

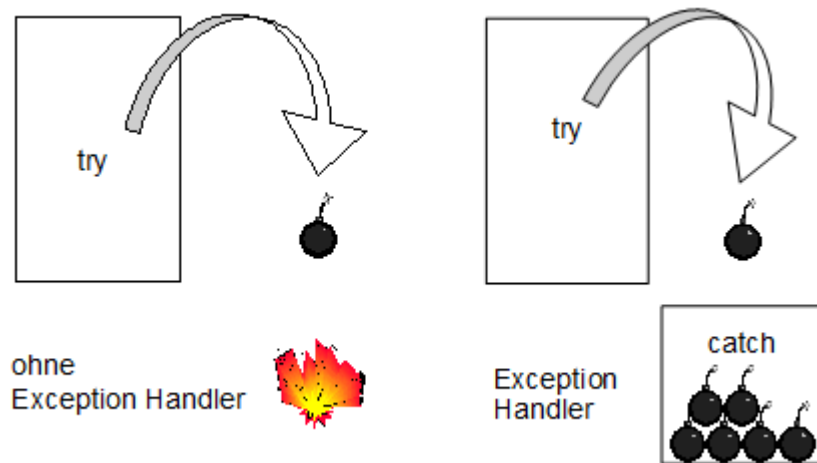
Inhalte

✓ Konzept des Exception Handling

✓ Implementierung

Ausnahmen

- Das Exception-Handling wird in Java durch eine try-Anweisung realisiert.
  - Eine try-Anweisung muss einen try-Block und kann ein oder mehrere catch Konstrukte und ein finally Konstrukt enthalten. Ist mindestens ein catch Konstrukt da, so kann das finally Konstrukt entfallen, ansonsten ist es erforderlich.
  - Falls ein ExceptionHandler gefunden wird, werden die Anweisungen des Handlers als nächstes ausgeführt und das Programm nach Handlern fortgesetzt.



## ➤ try - Anweisung

```
try{
    ...           // try-Block. Das ist der normale Code,
                  // in dem Fehler auftreten können
}

catch (Exceptiontyp1 ex1){
    ...           //catch-Block1. Fängt Fehler der
                  // Klasse Exceptiontyp1 ab
}

catch (Exceptiontyp2 ex2){
    ...           //catch-Block2. Fängt Fehler der
                  // Klasse Exceptiontyp2 ab
}

...

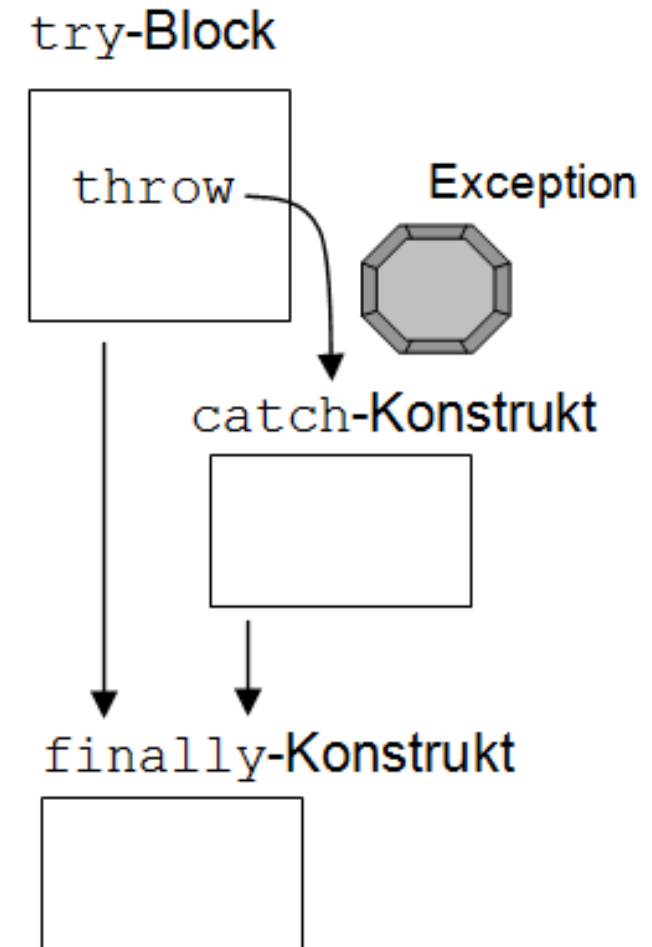
// weitere catch-Konstrukte

finally{
    ...           // finally-Konstrukt ist optional.
                  // Wird in jedem Fall durchlaufen,
                  // egal ob ein Fehler aufgetreten
                  // ist oder nicht.
}
```

try-  
Konstruktcatch-  
Konstruktefinally-  
Konstrukt

## ➤ try - Anweisung

- Tritt ein Programmfehler in einem `try` Block auf, wird eine Instanz der entsprechenden Exception Klasse mit `throw` geworfen und der gerade ausgeführte `try` Block verlassen.
- Die Generierung eines Exception Objektes und die Übergabe mit `throw` an die virtuelle Maschine wird als das Auslösen (Werfen) einer Exception bezeichnet. Das Exception Objekt enthält Informationen über den aufgetretenen Fehler.

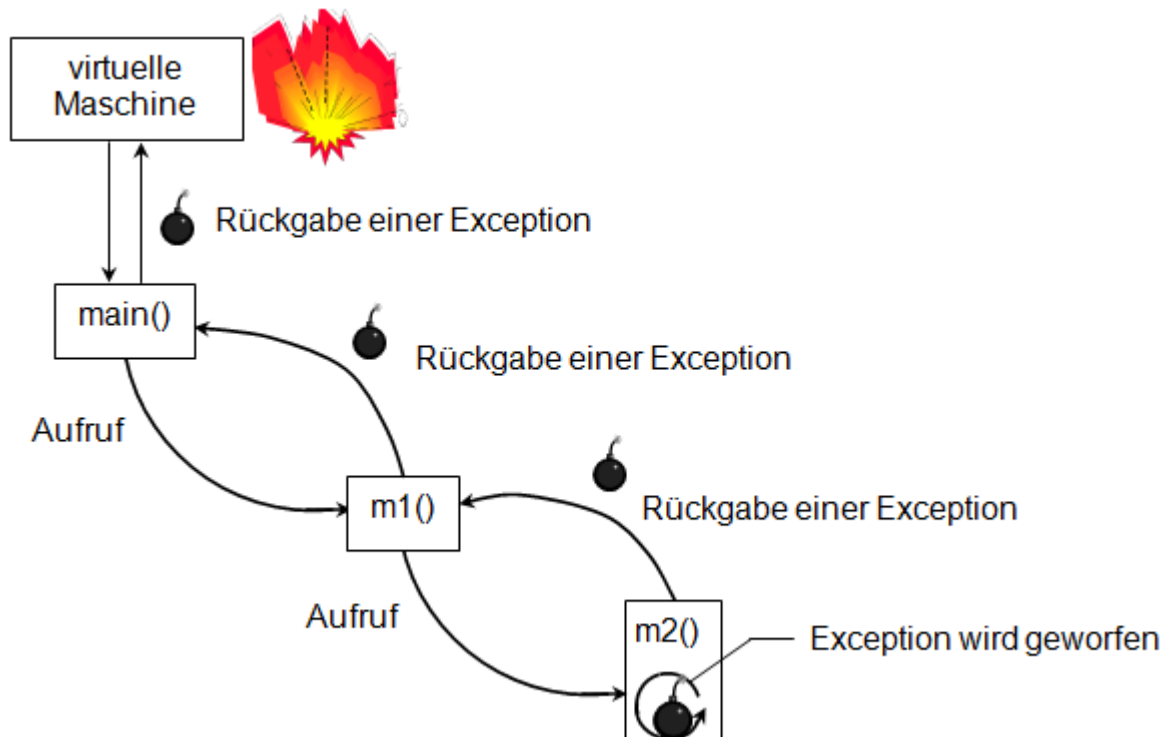




### ➤ Aufgabe 10.01 Division durch Null

Entwickeln Sie eine Klasse `Teilen`. Initialisieren Sie in der `main()`-Methode die zwei Variablen `zaehler` und `nenner` vom Typ `int`. Der Variablen `zaehler` wird eine beliebige ganze Zahl und der Variablen `nenner` die Zahl 0 zugewiesen. Danach soll das Ergebnis der Berechnung `zaehler/nenner` in der Konsole ausgegeben werden. Fangen Sie die bei der Berechnung entstehende Ausnahme `ArithmeticException` in einem `try-catch`-Block ab. Geben Sie einen Hinweis über den aufgetretenen Fehler in der `catch`-Klausel aus und analysieren Sie die Ausnahme, indem Sie sich Informationen über die Exception mit Hilfe der Methode `printStackTrace()` in der Konsole ausgeben lassen.

- Propagieren einer nicht abgefangenen Exception
- eine nicht abgefangene Exception wird an den Aufrufer - bis hin zur virtuellen Maschine - propagiert.



### ➤ throws Klausel

- Eine Methode muss Exceptions nicht selber abfangen, dieses kann auch dem Aufrufer zugewiesen werden. Dieser muss sich dann um das Exception Handling kümmern
- Durch die `throws` Klausel informiert eine Methode den Aufrufer (und den Compiler) über eine mögliche abnormale Rückkehr aus der Methode

```
public Date pruefeDatum(String datum) throws ParseException{  
    ...  
}
```

### ➤ Aufgabe 10.02 Datum prüfen

Vervollständigen Sie die Klasse `Aufg_10_02.java` aus dem Online-Campus. Die Methode `pruefeDatum(String datum)` ist bereits fertig implementiert. Ergänzen Sie die `main()`-Methode um Anweisungen innerhalb von einem `try/catch`-Konstrukt, sodass diese Prüfmethode mit einem Datum als Zeichenkette aufgerufen wird. Geben Sie auf der Systemausgabe aus, ob es sich um ein gültiges Datum handelt.

- Rufen Sie das Programm einmal mit einem gültigen Datum (z.B. 24.12.2013) auf und einmal mit einem ungültigen (z.B. 31.02.2014) auf.
- Was meldet Eclipse, wenn Sie das `throws`-Konstrukt aus der `pruefeDatum`-Methode entfernen?
- Was meldet Eclipse, wenn Sie versuchen, die Methode `pruefeDatum` ohne `try/catch`-Konstrukt zu implementieren?

1. Grundbegriffe der Programmierung

2. Einfache Beispielprogramme

3. Datentypen und Variablen

4. Ausdrücke und Operatoren

5. Kontrollstrukturen

6. Blöcke und Methoden

7. Klassen und Objekte

8. Vererbung und Polymorphie

9. Pakete

10. Ausnahmebehandlung

11. Schnittstellen (Interfaces)

12. Geschachtelte Klassen

13. Ein-/Ausgabe und Streams

14. Applets / Oberflächenprogrammierung

Inhalte

✓ Konzept des Exception Handling

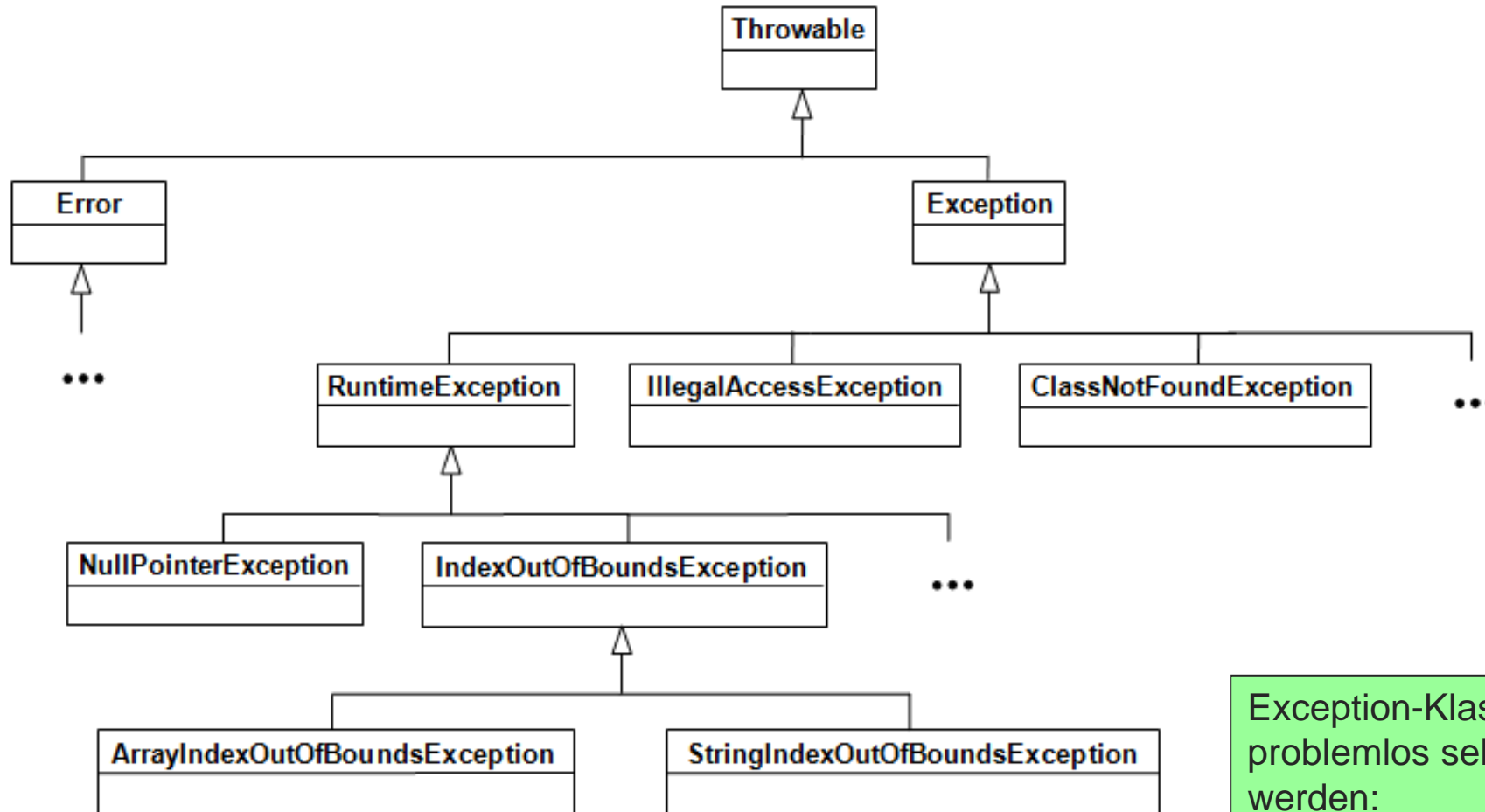
✓ Implementierung

✓ Ausnahmen

## Ausnahmen vereinbaren und auswerfen

- Exceptions haben eine gemeinsame Basisklasse (Throwable) aus dem Paket java.lang.
- Eine Ausnahme-Klasse unterscheidet sich nicht von einer „normalen“ Klasse, außer dass sie von Throwable abgeleitet ist. Die besondere Bedeutung erhält sie durch die Verwendung in throw Anweisungen und in catch Konstrukten.
- Aufgerufen wird ein catch Konstrukt nicht vom Programm, sondern von der Java Virtuellen Maschine. Ein Exception Objekt wird an die virtuelle Maschine übergeben. Diese übernimmt die Kontrolle und sucht das passende catch Konstrukt und übergibt ihm die Exception

```
class MyException extends Exception
{
    public MyException()
    {
        // Aufruf des Konstruktors der Klasse Exception.
        // Ihm wird ein String mit dem Fehlertext uebergeben und
        // in einem von Throwable geerbten Datenfeld gespeichert.
        super ("Fehler ist aufgetreten!");
    }
}
```



Exception-Klassen können problemlos selbst erstellt werden:

```
class MyExceptionDate
extends Exception{
...
}
```

### ➤ Checked Exception

- Exception muss vom Programmierer behandelt werden
- dies wird auch vom Compiler überprüft (checked)

### ➤ Unchecked Exception

- Exception muss nicht, kann aber vom Programmierer behandelt werden
- wird vom Compiler nicht überprüft

Unchecked Exceptions	Checked Exceptions
<code>RuntimeException</code> <code>Error</code>	alle anderen



## ➤ RuntimeException

- RuntimeException treten zur Laufzeit in der VM auf

Exception	Erklärung
ArithmeticException	Ein Integerwert wurde durch Null dividiert
ArrayIndexOutOfBoundsException	Auf ein Feld mit ungültigem Index wurde zugegriffen
ClassCastException	Cast wegen fehlender Typverträglichkeit nicht möglich
NullPointerException	Versuchter Zugriff auf ein Datenfeld oder eine Methode über die null-Referenz

Exceptions vom Typ  
RuntimeException

Exception	Erklärung
AbstractMethodError	Versuch, eine abstrakte Methode aufzurufen
InstantiationError	Versuchtes Anlegen einer Instanz einer abstrakten Klasse oder einer Schnittstelle
OutOfMemoryError	Es konnte kein Speicher allokiert werden
StackOverflowError	Der Stack ist übergelaufen

Exceptions vom Typ  
Error

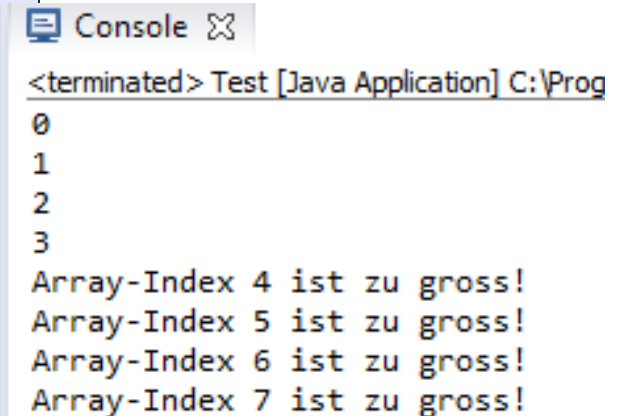
Exception	Erklärung
ClassNotFoundException	Eine Klasse wurde weder im aktuellen Verzeichnis noch in dem Verzeichnis, welches in der Umgebungsvariable CLASSPATH angegeben ist, gefunden
CloneNotSupportedException	Ein Objekt sollte kopiert werden, welches das Cloning aber nicht unterstützt
IllegalAccessException	Ein Objekt hat eine Methode aufgerufen, auf die es keinen Zugriff hat

Exceptions vom Typ  
Exception

## Ausnahmen behandeln I

- Der try-Block bedeutet: Es wird versucht, den Code in den geschweiften Klammern auszuführen. Wenn Exceptions geworfen werden, hat sich der Programmierer um die Behandlung zu kümmern.
- Eine Exception kann in einem catch Konstrukt, das dem try Block folgt, behandelt werden, oder an die aufrufende Methode zur Behandlung weitergereicht werden.
- Hat der Programmierer jedoch keinen Exception Handler für eine weitergereichte Checked Exception geschrieben, dann meldet sich der Compiler mit einer Fehlermeldung. Damit erzwingt der Compiler, dass eine Checked Exception vom Programmierer behandelt wird.
- Wird eine weitergereichte Unchecked Exception vom Programmierer nicht abgefangen, meldet sich das Laufzeitsystem mit einer Fehlermeldung und bricht das Programm ab.

```
public class Test{
    public static void main (String[] args)    {
        int[] intarr = new int [4];
        for (int lv = 0; lv < 8; lv++)
        {
            try                {
                intarr [lv] = lv;
                System.out.println (intarr [lv]);
            }
            catch (ArrayIndexOutOfBoundsException e){
                System.out.println ("Array-Index " + lv + " ist zu gross!");
            }
        }
    }
}
```



Console

<terminated> Test [Java Application] C:\Prog

0  
1  
2  
3  
Array-Index 4 ist zu gross!  
Array-Index 5 ist zu gross!  
Array-Index 6 ist zu gross!  
Array-Index 7 ist zu gross!

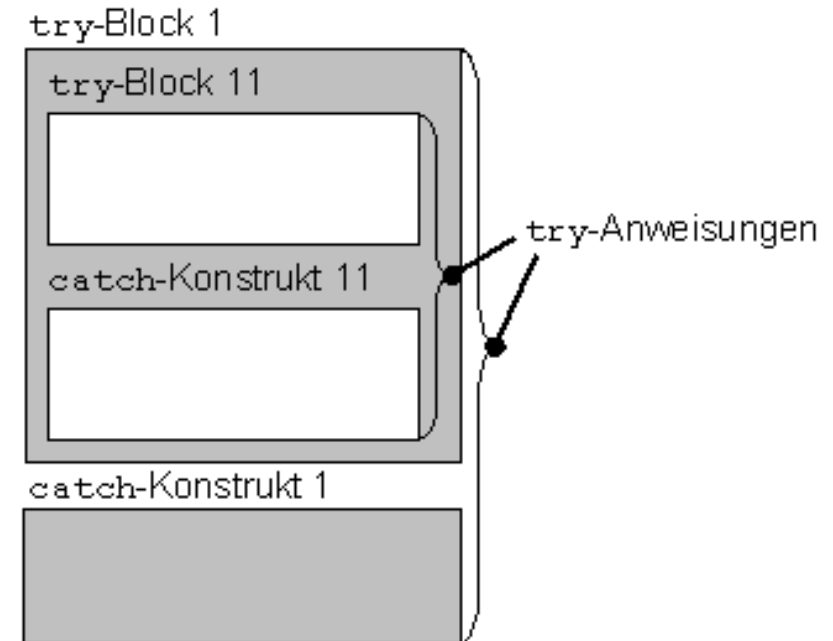
## Ausnahmen behandeln II

---

- Ein Handler für Exceptions einer Klasse A passt infolge des Polymorphie Konzeptes der Objektorientierung auch auf Exceptions aller von A abgeleiteten Klassen.
- Zuerst müssen die Handler für die am meisten spezialisierten Klassen der Exception-Hierarchie aufgelistet werden und dann in der Reihenfolge der zunehmenden Generalisierung die entsprechenden allgemeinen Handler.
- Fügt man am Ende der Folge der Handler noch einen Handler für die Basisklasse ein, ist man auch in Zukunft sicher, dass alle Exceptions behandelt werden, auch wenn jemand neue Exceptions ableitet.
- Stellt sich innerhalb des Handlers (z.B. anhand der in der Exception übergebenen Informationen oder weil Korrekturmaßnahmen fehlgeschlagen) heraus, dass dieser Handler die Exception nicht behandeln kann, so kann dieselbe Exception erneut im catch Block mit throw ausgeworfen werden. Der Handler kann aber ggf. auch andere Exceptions auswerfen.
- Alle innerhalb von Handlern ausgeworfenen Exceptions werden nach außen an die nächste umschließende try Anweisung weitergereicht. Die try Anweisungen können also geschachtelt werden.

```
class MyException2 extends Exception
{
    public MyException2()
    {
        super ("Fehler ist aufgetreten!");
    }
}
```

```
public class Versuch
{
    public static void main (String[] args){
        try {
            try{
                throw new Exception();
            }
            catch (MyException2 e){
                System.out.println ("MyException2 gefangen");
            }
        }
        catch (Exception e2){
            System.out.println ("Exception gefangen");
        }
    }
}
```



## Vorteile des Exception Konzeptes

---

- Eine saubere Trennung des Codes in „normalen“ Code und in Fehlerbehandlungscode.
- Der Compiler prüft, ob Checked Exceptions vom Programmierer abgefangen werden.  
Damit werden Nachlässigkeiten beim Programmieren bereits zur Kompilierzeit und nicht erst zur Laufzeit entdeckt.
- Das Propagieren einer Exception erlaubt, diese auch in einem umfassenden Block oder einer aufrufenden Methode zu behandeln.
- Da Exception Klassen in einem Klassenbaum angeordnet sind, können (je nach Bedarf) spezialisierte Handler oder generalisierte Handler geschrieben werden..

### ➤ Aufgabe 10.03 Bankkonto

Erstellen Sie eine Klasse `Bankkonto`. Eine Kontoführung soll durch Einzahlungen und Auszahlungen simuliert werden. Die Klasse `Bankkonto` besitzt die Methoden:

- `public void einzahlen (double betrag)`
- `public void auszahlen (double betrag)`
- `public double getKontostand()`

Die Methoden `einzahlen()` und `auszahlen()` werfen eine Exception vom Typ `TransaktionsException` beim Auftreten eines Transaktionsfehlers. Leiten Sie hierzu die Klasse `TransaktionsException` von der Klasse `Exception` ab. Ein Transaktionsfehler wird durch einen negativen Ein- und Auszahlungsbetrag oder ein nicht ausreichend großes Guthaben für einen Auszahlungsbetrag verursacht. Die Methode `getKontostand()` liefert den aktuellen Kontostand, der durch ein privates Datenfeld vom Typ `double` realisiert wird. Die Klasse `Bankkonto` wird mit der Klasse `TestBankkonto` getestet (zu finden im Online-Campus als `TestBankkonto.java`).

### ➤ Aufgabe 10.04 Login

Es soll ein Login-Szenario entwickelt werden. Die Klasse Login besitzt folgende Instanzvariablen und Methoden:

- `private boolean angemeldet;`
- `public void anmelden (String benutzer, String passwort)`
- `public void abmelden()`
- `public void bearbeiten()`

Die Methode `anmelden()` setzt bei erfolgreicher Anmeldung die Instanzvariable `angemeldet` auf `true` und wirft bei fehlschlagender Authentisierung ein Objekt der Klasse `ZugriffUnguelteigException`, die von der Klasse `Exception` abgeleitet wird. Ebenfalls soll, wenn ein nicht angemeldeter Benutzer auf die Methode `bearbeiten()` zugreifen möchte, eine Ausnahme vom Typ `KeineBerechtigungException` geworfen werden. Die Methode `abmelden()` setzt die Instanzvariable `angemeldet` auf `false`. Die Methode `bearbeiten()` gibt eine Meldung auf der Konsole aus, um einen Arbeitsvorgang zu simulieren. Entwickeln Sie die Klassen `Login`, `ZugriffUnguelteigException`, `KeineBerechtigungException`. Die entwickelten Klassen sollen mit der Klasse `Testlogin` aus dem Online Campus getestet werden.