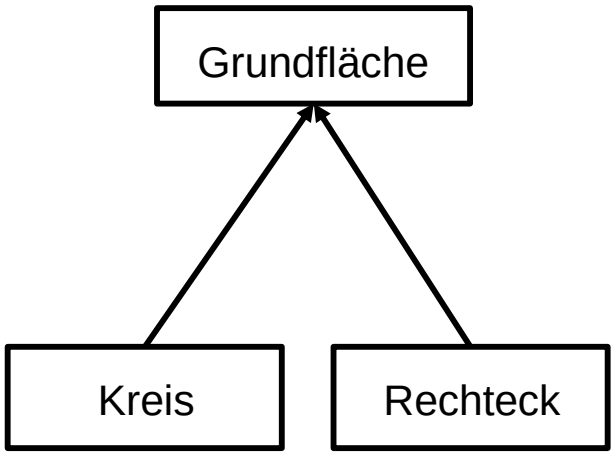
PGdP Tutorium: Elfte Stunde

Benedikt Werner

München, 15. Januar 2017

Abstrakte Klassen

- Klassen, die keine Instanzen haben können
- Sinnvoll für „abstrakte“ Oberklassen




```
public abstract class Grundfläche {
    public abstract double umfang();

    public abstract double flaeche();

    public String toString() {
        return "Grundfläche("
                + umfang() + ", "
                + flaeche() + ")";
    }
}
```

Interfaces

- Interfaces definieren bestimmte Methoden, die Klassen mit diesem Interface zur Verfügung stellen müssen
- Im Prinzip sind Interfaces abstrakte Klassen, eine Klasse kann aber mehrere Interfaces implementieren
- Methoden eines Interfaces sind automatisch `abstract`
- Mit dem Keyword `default` kann eine Standardimplementierung gegeben werden

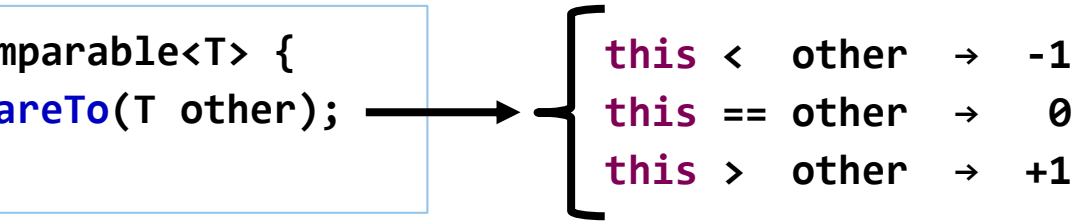
```
public interface Printable {
    void print();

    default void printMiniJava() {
        MiniJava.write(toString());
    }
}
```

```
class MyClass implements Printable {
    void print() {
        System.out.println("MyClass");
    }
}
```

```
class A implements B, C, D
```

Aufgabe 11.2

```
public interface Comparable<T> {
    public int compareTo(T other);
}
```



1. `Grundflaeche` und `Prisma` sollen `Comparable<T>` implementieren
   - Grundflaechen werden nach Fläche verglichen
   - Prismen werden nach Volumen verglichen
   - Wenn `other == null`, dann soll eine `NullPointerException` geworfen werden
   - Test:

```
List<Grundflaeche> gs = new LinkedList();
gs.add(new Kreis(7)); gs.add(new Rechteck(2, 2));
Collections.sort(gs);
```

2. `istQuadrat()` und `zuQuadrat()` in ein Interface `Quadrierbar` verschieben, dass nur die Klassen implementieren, deren Instanzen überhaupt ein Quadrat sein können
3. Interface `Polygon` mit der Methode `int getEckenAnzahl()` erstellen, dass alle Klassen implementieren, die eine endliche Zahl an Ecken besitzen
4. Testen

Keyword `final`

- Variablen die mit `final` markiert wurden können nicht verändert werden
- Methoden die mit `final` markiert wurden können nicht überschrieben werden
- Von Klassen die mit `final` markiert wurden kann nicht geerbt werden

```
final int zahl = 3;
zahl = 5;  // FEHLER!

final int[] array = new int[4];
array[0] = 3;        // OK!
array = new int[2]; // FEHLER!
```

Try-catch

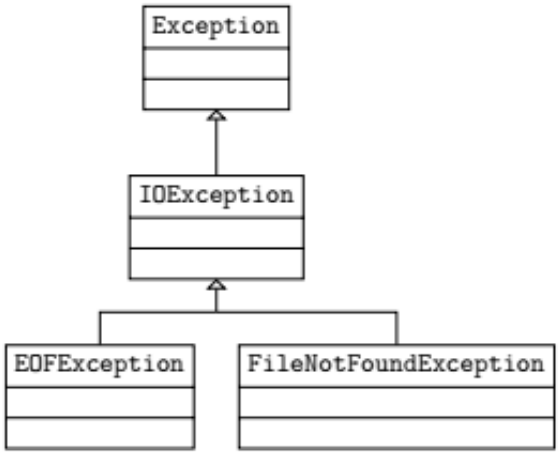
- Dienen zur Fehlerbehandlung
- Wenn im try-Block ein Fehler auftritt und ein zum Fehler passender catch-Block vorhanden ist wird der Fehler abgefangen und der catch-Block ausgeführt

```
try {
    // …
}
catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("ArrayIndexOutOfBoundsException gefangen");
}
catch (EOFException | FileNotFoundException e) {
    // …
}
catch (Exception e) {
    System.out.println("Exception gefangen!");
}
```

Aufgabe 11.3

Was wird auf der Konsole ausgegeben, wenn im try-Block

1. eine EOFException geworfen wird?
2. eine FileNotFoundFException geworfen wird?
3. eine Division durch 0 erfolgt?
4. Keine Exception gewrofen wird?




```java
import java.io.*;
public class ExceptionTest {
   public static void main (String[] args) {
      try {
        // ...
      }
      catch (EOFException e) {
         System.out.println("EOFException");
      }
      catch (IOException e) {
         System.out.println("IOException");
      }
      catch (Exception e) {
         System.out.println("Exception");
      }
      System.out.println("ENDE");
   }
}
```