

Funktionale und objektorientierte Programmierkonzepte



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Präsenz-Sprechstunde B

Simon Hock, Nhan Huynh, Daniel Mangold



Organisatorisches

Vererbung

Überschreiben vs. Überladen von Methoden

Konstrukturen

Schlüsselwort this

Enums

Arbeitsphase

- 24.11.2021 Präsenzsprechstunde im Raum in Raum S103/**223**!
- Erinnerung: Themenvorschläge im Forum
- Zusätzliche Materialien wie die Präsentationen sind im Moodle Forum zu finden.



- Grundlegendes Konzept der Objektorientierung
- Modellierung von Hierarchien in der realen Welt mit Hilfe von Klassen
- Basisklasse: Generalisierung
 - Verallgemeinerung eines Objekts mit seinem Verhalten
- Abgeleitete Klassen: Spezialisierung
 - Eigene Ausprägungen und eigenes Verhalten

Beispiel Typhierarchie

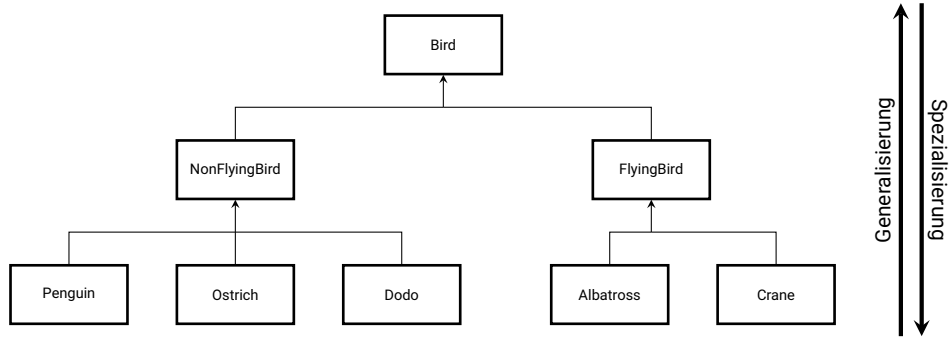


Abbildung: Typhierarchie - Modellierung von Vögeln



- Eigene Implementierung einer geerbten Methode
- Erlaubt das Funktionalitäten in der abgeleiteten Klasse zu verändern
- Gleichzeitig können aber die anderen Methoden der Oberklasse immer noch weiter verwendet werden.
- `@Override` Annotation (optional)
 - ▣ Informiert den Compiler, dass das Element ein in einer Oberklasse deklariertes Element überschreiben soll.

Erweiterung der Klasse Robot

```
1 public class FastRobot extends Robot {  
2  
3     public FastRobot(int x, int y, Direction direction, int numberOfCoins) {  
4         super(x, y, direction, numberOfCoins);  
5     }  
6  
7     @Override  
8     public void move() {  
9         super.move();  
10        super.move();  
11    }  
12 }
```

Welche Implementierung wird ausgeführt?



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Implementierung von Robot?
- Implementierung von FastRobot?

```
1 Robot robot = new FastRobot(0,0,Direction.UP,0);  
2 robot.move();
```




- Methoden mit gleichem Namen können innerhalb einer Klasse definiert werden.
- Der Rückgabebetyp kann beliebig sein, aber die Parameterlisten müssen verschieden sein.

Information: Methoden werden nicht ausschließlich anhand ihres Namens identifiziert, sondern auch über die Typen, Anzahl und Reihenfolge ihrer Argumente (Parameter) - ihre Signatur.

Überladen von Methoden - Beispiel



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1 public class Point {  
2  
3     private final double x;  
4     private final double y;  
5  
6     public Point(double x, double y){  
7         this.x = x;  
8         this.y = y;  
9     }  
10 }
```

```
1 public Point copy() {  
2     return new Point(x, y);  
3 }  
4  
5 public Point[] copy(int n) {  
6     Point[] copies = new Point[n];  
7     for (int i = 0; i < copies.length; i++) {  
8         copies[i] = new Point(x, y);  
9     }  
10    return copies;  
11 }
```

- Haben keinen Rückgabewert und der Name ist gleich dem Namen der Klasse.
- Konstrukturen werden nur einmal aufgerufen und zwar, wenn man ein neues Objekt erstellt.
- Konstrukturen werden nicht vererbt.
- Falls kein Konstruktor definiert wird, legt der Compiler einen leeren Konstruktor ohne Parameter an.
 - ▣ Es gibt mindestens einen Konstruktor.
 - ▣ Konstrukturen können überladen werden.

```
1 Robot robot = new Robot(0,0,Direction.UP,0);
```

```
1 public Robot(int x, int y) {  
2     super(x, y);  
3     setGlobalWorld();  
4  
5     world.checkXCoordinate(x);  
6     world.checkYCoordinate(y);  
7  
8     world.addRobot(this);  
9 }
```



- Referenzvariable, die auf das aktuelle Objekt verweist.
- Das Schlüsselwort `this` wird hauptsächlich in drei Situationen verwendet:
 - ▣ Vermeidung von Mehrdeutigkeit von Variablenreferenzen
 - ▣ Aktuelles Objekt als Argument, das an ein anderes Objekt übergeben wird
 - ▣ Alternativer Konstruktoraufruf

Vermeidung von Mehrdeutigkeit von Variablenreferenzen

```
1 public class Point {  
2  
3     private final double x;  
4     private final double y;  
5  
6     public Point(double x, double y){  
7         this.x = x;  
8         this.y = y;  
9     }  
10 }
```

Aktuelles Objekt als Argument, das an ein anderes Objekt übergeben wird

```
1 public Point(Point point) {  
2     x = point.x;  
3     y = point.y;  
4 }  
5  
6 public Point copy() {  
7     return new Point(this);  
8 }
```



```
1 public Point(double x, double y){  
2     this.x = x;  
3     this.y = y;  
4 }  
5  
6 public Point() {  
7     this(0, 0);  
8 }
```



- Kurzform von Enumeration - Aufzählung
- Abgeleitete Klassen von `java.lang.Enum`
- Bietet die Möglichkeit, vordefinierte Konstanten für Variablen festzulegen
- Nützliche Methode: `int ordinal()`: Gibt die Position des Enums in der Enumdeklaration zurück.

```
1 public enum Direction {  
2     UP,  
3     RIGHT,  
4     DOWN,  
5     LEFT  
6 }
```

Selbstständiges Arbeiten