

# Refactoring

---

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

# Was ist Refactoring?

*Refactoring ist, wenn einem auffällt, daß der Funktionsname `foobar` ziemlich bescheuert ist, und man die Funktion in `sinus` umbenennt.*

Quelle: "356: Refactoring" by Andreas Bogk on Lutz Donnerhacke: "Fachbegriffe der Informatik"

# Was ist Refactoring?

*Refactoring ist, wenn einem auffällt, daß der Funktionsname `foobar` ziemlich bescheuert ist, und man die Funktion in `sinus` umbenennt.*

Quelle: "356: Refactoring" by Andreas Bogk on Lutz Donnerhacke: "Fachbegriffe der Informatik"

*Refactoring (noun): a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour.*

Quelle: (Fowler 2011, p. 53)

# Anzeichen, dass Refactoring jetzt eine gute Idee wäre

- Code “stinkt” (zeigt/enthält *Code Smells*)
- Schwer erklärbarer Code
- Verständnisprobleme, Erweiterungen

*“Three strikes and you refactor.”*

Quelle: (Fowler 2011, p. 58): “The Rule of Three”

# Bevor Sie loslegen ...

1. **Unit Tests** schreiben
  - Normale und ungültige Eingaben
  - Rand- und Spezialfälle
2. **Coding Conventions** einhalten
  - Sourcecode formatieren (lassen)
3. Haben Sie die fragliche Codestelle auch wirklich verstanden?!

# Vorgehen beim Refactoring

- Kleine Schritte: immer nur **eine** Änderung zu einer Zeit
- Nach **jedem** Refactoring-Schritt **Testsuite** laufen lassen
  - => Nächster Refactoring-Schritt erst, wenn alle Tests wieder “grün”
- Versionskontrolle nutzen: **Jeden** Schritt **einzel**n committen

# Refactoring-Methode: Rename Method/Class/Field

## Vorher

```
public String getTeN() {}
```

## Nachher

```
public String getTelefonNummer() {}
```

# Refactoring-Methode: Encapsulate Field

## Vorher

```
int cps;

public void printDetails() {
    System.out.println("Credits: " + cps);
}
```

## Nachher

```
private int cps;

int getCps() { return cps; }
void setCps(int cps) { this.cps = cps; }

public void printDetails() {
    System.out.println("credits: " + getCps());
}
```



# Refactoring-Methode: Extract Method/Class

## Vorher

```
public void printInfos() {  
    printHeader();  
    // Details ausgeben  
    System.out.println("name:    " + name);  
    System.out.println("credits: " + cps);  
}
```

## Nachher

```
public void printInfos() {  
    printHeader();  
    printDetails();  
}  
  
private void printDetails() {  
    System.out.println("name:    " + name);  
    System.out.println("credits: " + cps);  
}
```

# Refactoring-Methode: Move Method

## Vorher

```
public class Kurs {  
    int cps;  
    String descr;  
}  
  
public class Studi extends Person {  
    String name;  
    int cps;  
    Kurs kurs;  
  
    public void printKursInfos() {  
        System.out.println("Kurs: " + kurs.descr);  
        System.out.println("Credits: " + kurs.cps);  
    }  
}
```

# Refactoring-Methode: Move Method (cnt.)

## Nachher

```
public class Kurs {  
    int cps;  
    String descr;  
  
    public void printKursInfos() {  
        System.out.println("Kurs: " + descr);  
        System.out.println("Credits: " + cps);  
    }  
}  
  
public class Studi extends Person {  
    String name;  
    int cps;  
    Kurs kurs;  
  
    public void printKursInfos() { kurs.printKursInfos(); }  
}
```

# Refactoring-Methode: Pull Up, Push Down (Field, Method)

## Vorher

```
public class Person { }

public class Studi extends Person {
    String name;
    public void printDetails() { System.out.println("name: " + name); }
}
```

## Nachher

```
public class Person { protected String name; }

public class Studi extends Person {
    public void printDetails() { System.out.println("name: " + name); }
}
```

Behebung von **Bad Smells** durch **Refactoring**

=> Änderung der inneren Struktur ohne Beeinflussung des äußeren Verhaltens

- Verbessert Lesbarkeit, Verständlichkeit, Wartbarkeit
- Immer nur kleine Schritte machen
- Nach jedem Schritt Testsuite laufen lassen
- Katalog von Maßnahmen, beispielsweise *Rename*, *Extract*, *Move*, *Push Up/Pull Down*, ...
- Unterstützung durch IDEs wie Eclipse, Idea, ...



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

## Exceptions

- Citation “*Refactoring ist, wenn . . .*”: “356: Refactoring” by Andreas Bogk on Lutz Donnerhacke: “Fachbegriffe der Informatik”
- Citation “*Refactoring (noun): a change . . .*”: (Fowler 2011, p. 53)
- Citation “*The Rule of Three*”: (Fowler 2011, p. 58)