

Refactoring

REFACTORING

MOTIVATION

- Erhöhen der Lesbarkeit
- Reduzieren der Komplexität
- Erhöhen der Wartbarkeit
- Erhöhen der Erweiterbarkeit
- Erhöhen der Testbarkeit

WORT-HERKUNFT

Factoring

== De-Komposition; Zerlegen, Aufteilen von komplizierten Problemen in kleine Teile

Re-Factoring

Ändern der Zerlegung

BEDEUTUNG

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

— Martin Fowler

WICHTIGE ELEMENTE

1. Ändern der **internen Struktur**
2. **externes Verhalten** bleibt gleich
3. **diszipliniertes** Vorgehen
4. Viele **kleine** Schritte
 - Mikro-Refactoring
 - es kann weniger kaputt gehen

UNTERSTÜTZUNG

- (automatisierte) Tests
- IDE mit Support für Mikro-Refactorings
- Typsystem der Programmiersprache
 1. strenge Typisierung
 2. dynamische Typisierung

BEISPIELE REFACTORING

UMBENENNEN VON VARIABLEN

Vorher

```
public String getFullName(String s1, String s2) {  
  
    s1 = s1.trim();  
    s2 = s2.trim();  
  
    return s1 + " " + s2;  
}
```

Nachher

```
public String getFullName(String vorname, String nachname) {  
  
    vorname = vorname.trim();  
    nachname = nachname.trim();  
  
    return vorname + " " + nachname;  
}
```

EXTRAHIEREN VON METHODEN

```
@Bean
public CommandLineRunner loadData(MemberProfileRepository repository) {
    return (args) -> {
        /// STEP 1

        // save a couple of profiles
        repository.save(new MemberProfile("robkle", "Kleinschmager"));
        repository.save(new MemberProfile("mickni", "Knight"));
        repository.save(new MemberProfile("geolaf", "Laforge"));

        // STEP 2

        // fetch all profiles
        log.info("MemberProfiles found with findAll():");
        log.info("-----");
        for (MemberProfile profile : repository.findAll()) {
            log.info(profile.toString());
        }
        log.info("");

        // STEP 3

        // fetch an individual customer by ID
        MemberProfile profile = repository.findOne(1L);
        log.info("Profile found with findOne(1L):");
        log.info("-----");
        log.info(profile.toString());
        log.info("");
    };
};
```

EXTRAHIEREN VON METHODEN 2

```
@Bean
public CommandLineRunner loadData(MemberProfileRepository repository) {
    return args -> {

        deleteAllExistingProfiles(repository);
        importProfiles(repository);
        fetchAndPrintAllProfiles(repository);

    };
}
```

VARIABLE IN OBJECT UMWANDELN

Vorher

```
public class Order {  
  
    String customer;  
    List<Item> items;  
}
```

Nachher

```
public class Order {  
  
    Customer customer;  
    List<Item> items;  
}  
  
public class Customer {  
    String name;  
}
```

CODE FORMATIERUNG

Vorher

```
public boolean equals(Object obj) {  
  
    if (  
        this == obj) return true;  
        if (!(obj instanceof MemberProfile)) {  
            return false; }  
        MemberProfile that = (MemberProfile) obj;  
        EqualsBuilder eb = new EqualsBuilder();  
        eb.append(this.getMemberId(), that.getMemberId());  
  
    return eb.isEquals();  
}
```

CODE FORMATTIERUNG

Nachher

```
public boolean equals(Object obj) {  
  
    if ( this == obj) {  
        return true;  
    }  
  
    if (!(obj instanceof MemberProfile)) {  
        return false;  
    }  
  
    MemberProfile that = (MemberProfile) obj;  
    EqualsBuilder eb = new EqualsBuilder();  
    eb.append(this.getMemberId(), that.getMemberId());  
  
    return eb.isEquals();  
}
```

BESCHREIBENDE VARIABLEN

Vorher

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&  
      (browser.toUpperCase().indexOf("IE") > -1) &&  
      wasInitialized() && resize > 0 )  
{  
    // do something  
}
```

Nachher

```
boolean isMacOs      = platform.toUpperCase().indexOf("MAC") > -1;  
boolean isIEBrowser  = browser.toUpperCase().indexOf("IE")  > -1;  
boolean wasResized   = resize > 0;  
  
if (isMacOs && isIEBrowser && wasInitialized() && wasResized)  
{  
    // do something  
}
```

VORHANDENES OBJECT ÜBERGEBEN

Vorher

```
int low = daysTempRange().getLow();  
int high = daysTempRange().getHigh();  
  
withinPlan = plan.withinRange(low, high);
```

Nachher

```
withinPlan = plan.withinRange(daysTempRange());
```


QUELLEN

- <https://www.refactoring.com>