

Studiengang: Wirtschaftsinformatik Bachelor



**Westfälische  
Hochschule**

Sommersemester 2017

4. Fachsemester

# **Architektur betrieblicher Informationssysteme**

## **Semesteraufgabe 7 - Entwurfsmuster (Strategie, Observer)**

Abgabedatum:

5. Juli 2017

Aktualisierungsdatum:

18. Juli 2017

Autoren:

Mehmet Tüfekci (Matr. 201521617) – [Mehmet.Tuefekci@studmail.w-hs.de](mailto:Mehmet.Tuefekci@studmail.w-hs.de)

Mario Kellner (Matr. 201520916) – [Mario.Kellner@studmail.w-hs.de](mailto:Mario.Kellner@studmail.w-hs.de)

Julian Kranen (Matr. 201223532) – [Julian.Kranen@studmail.w-hs.de](mailto:Julian.Kranen@studmail.w-hs.de)

Aufgabe 1 (Strategiemuster)

## Aufgabe 1 (Strategiemuster)

Erweitern Sie das Beispiel zum Strategie-Muster aus der Vorlesung, indem Sie eine (Schrei-) Stockente erzeugen, die ihr Quakverhalten von „Quaken“ auf „Schreien“ umschalten kann. Welche neue(n) Klasse(n) müssen Sie implementieren? Erstellen Sie den entsprechenden Quellcode.

Da wir das Interface Quakverhalten haben und wir nur ein neues Quakverhalten implementieren müssen, können wir das Interface **Quakverhalten** in eine Klasse namens „Schreien“ implementieren:

```
1 public class Schreien implements QuakVerhalten {
2     public void quaken() {
3         System.out.println("Quarwwoooooaaaaarr!");
4     }
5 }
```

Listing 1: Schreien.java

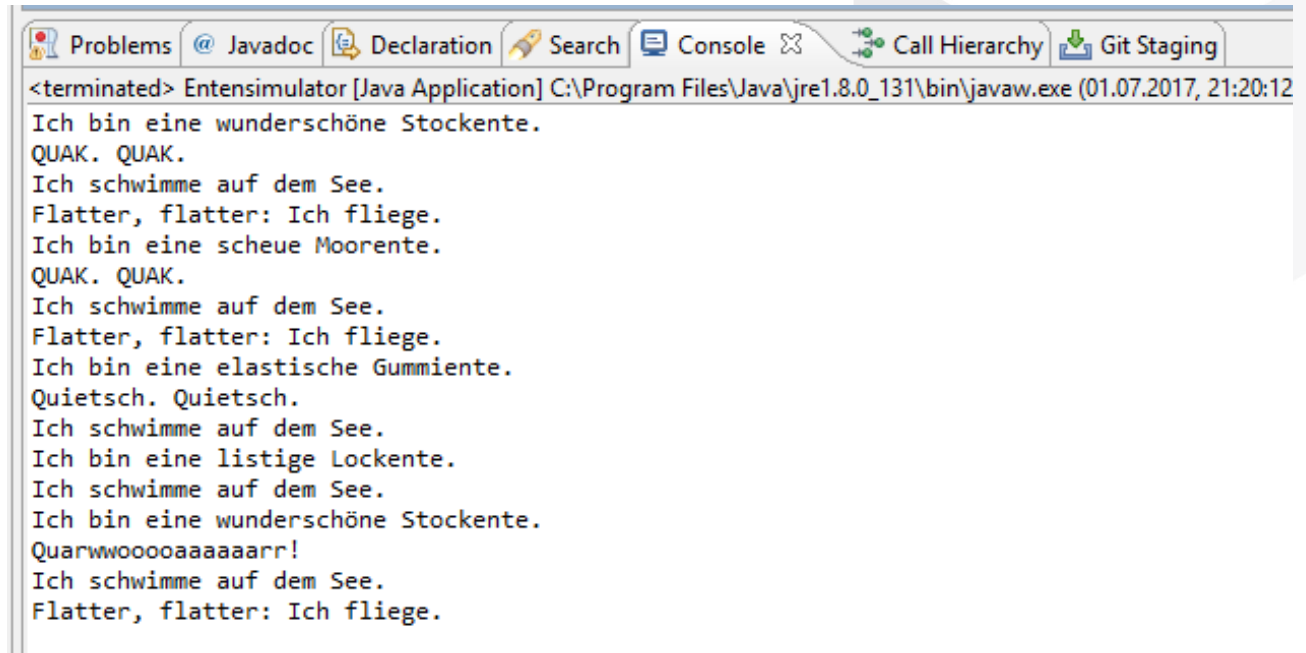
Wir müssen anschließend nur die Funktion setQuakVerhalten mit unserem Objekt aufrufen, damit wir das Quakverhalten zur Laufzeit ändern können:

```
1 [...]
2 public static void main(String[] args) {
3     ArrayList<Ente> entenliste = new ArrayList<Ente>();
4     entenliste.add(new Stockente());
5     entenliste.add(new Moorente());
6     entenliste.add(new Gummiente());
7     entenliste.add(new Lockente());
8
9     for (int x=0; x < entenliste.size(); x++)
10         enteInAktion(entenliste.get(x));
11
12     entenliste.get(0).setQuakVerhalten(new Schreien());
13
14     enteInAktion(entenliste.get(0));
15 }
16 [...]
```

Listing 2: EntenSimulator.java

Aufgabe 1 (Strategiemuster)

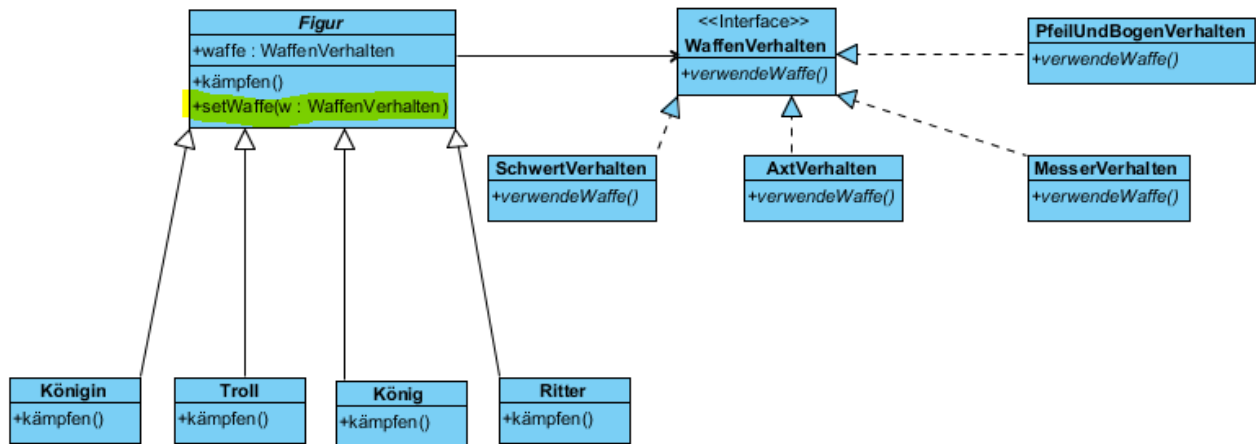
Konsolenausgabe:



```
<terminated> Entensimulator [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (01.07.2017, 21:20:12)
Ich bin eine wunderschöne Stockente.
QUAK. QUAK.
Ich schwimme auf dem See.
Flutter, flutter: Ich fliege.
Ich bin eine scheue Moorente.
QUAK. QUAK.
Ich schwimme auf dem See.
Flutter, flutter: Ich fliege.
Ich bin eine elastische Gummiente.
Quietsch. Quietsch.
Ich schwimme auf dem See.
Ich bin eine listige Lockente.
Ich schwimme auf dem See.
Ich bin eine wunderschöne Stockente.
Quarwwooooaaaaaarr!
Ich schwimme auf dem See.
Flutter, flutter: Ich fliege.
```

Aufgabe 2 (Strategie-Muster Klassendiagramm)

Aufgabe 2 (Strategie-Muster Klassendiagramm)



Die Funktion `setWaffe` wird in der abstrakten Klasse „Figur“ definiert.

Aufgabe 3 (Observer-Muster)

## Aufgabe 3 (Observer-Muster)

In einem Internet-Auktionshaus soll eine Funktion implementiert werden, die einem Interessenten Bescheid gibt, wenn sich der Preis eines Artikels geändert hat. Implementieren Sie diese Funktion (in Grundzügen, ähnlich wie in der Vorlesung) unter Verwendung des Observer-Patterns.

Ausgabe der Konsole:

```
Problems Javadoc Declaration Search Console Call Hierarchy Git Staging
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (01.07.2017, 20:40:53)
[Kranen] Gebot für Ultra geiler Laptop
[Kranen] Gebot von Tuefekci
[Kranen] Gebotener Preis 15.51

[Tuefekci] Gebot für Ultra geiler Laptop
[Tuefekci] Gebot von Tuefekci
[Tuefekci] Gebotener Preis 15.51

[Kranen] Gebot für Ultra geiler Laptop
[Kranen] Gebot von Tuefekci
[Kranen] Gebotener Preis 16.0

[Tuefekci] Gebot für Ultra geiler Laptop
[Tuefekci] Gebot von Tuefekci
[Tuefekci] Gebotener Preis 16.0

[Kranen] Gebot für Ultra geiler Laptop
[Kranen] Gebot von Kranen
[Kranen] Gebotener Preis 18.99

[Tuefekci] Gebot für Ultra geiler Laptop
[Tuefekci] Gebot von Kranen
[Tuefekci] Gebotener Preis 18.99
```

Folgende Klassen wurden definiert:

```
1 package observer;
2
3 import java.time.*;
4
5 public class Main {
6     public static void main(String[] args) {
7         Benutzer user_mar = new Benutzer(1, "Mario", "Kellner");
8         Benutzer user_meh = new Benutzer(2, "Mehmet", "Tuefekci");
9         Benutzer user_jul = new Benutzer(3, "Julian", "Kranen");
10
11         Auktion laptop = new Auktion(1, "Ultra geiler Laptop", "Mega geiler Laptop mit Pentium 4
12             Prozessor", LocalDateTime.of(2017, Month.JULY, 05, 12, 30));
13
14         user_meh.addToBeobachtungsListe(laptop);
15         user_jul.addToBeobachtungsListe(laptop);
16
17         laptop.setzeGebot(user_meh, 15.51f);
18
19         laptop.setzeGebot(user_meh, 16f);
20
21         laptop.setzeGebot(user_jul, 18.99f);
22
23         laptop.setzeGebot(user_mar, 49.99f);
```

Aufgabe 3 (Observer-Muster)

```
23
24     laptop.setzeGebot(user_meh, 99.99f);
25 }
26 }
```

Listing 3: Main.java

```
1 package observer;
2
3 public class Gebot {
4     public float preis;
5     public Benutzer benutzer;
6 }
```

Listing 4: Gebot.java

```
1 package observer;
2
3 import java.util.*;
4
5 public class Benutzer implements Observer {
6
7     public Integer kundenNummer = 0;
8     public String vorname = "";
9     public String nachname = "";
10    public List<Auktion> beobachtungsListe = new ArrayList<>();
11    public List<Auktion> aktiveAuktionen = new ArrayList<>();
12
13    public Benutzer(Integer kundenNummer, String vorname, String nachname) {
14        kundenNummer = kundenNummer;
15        vorname = vorname;
16        nachname = nachname;
17    }
18
19    public void addToAktiveAuktionen(Auktion auk) {
20        aktiveAuktionen.add(auk);
21    }
22
23    public void addTobeobachtungsListe(Auktion auk) {
24        auk.addObserver(this);
25
26        beobachtungsListe.add(auk);
27    }
28
29    @Override
30    public void update(Observable arg0, Object arg1) {
31        List<Gebot> gebote = (List<Gebot>)arg1;
32
33        System.out.println "[" + Nachname + "] Gebot für " + ((Auktion)arg0).ArikelName);
34        System.out.println "[" + Nachname + "] Gebot von " + gebote.get(gebote.size()-1).benutzer.
            Nachname);
```

Aufgabe 3 (Observer-Muster)

```
35     System.out.println("[ " + Nachname + " ] Gebotener Preis " + gebote.get(gebote.size()-1).preis);
36     System.out.println();
37
38 }
39 }
```

Listing 5: Benutzer.java

```
1 package observer;
2
3 import java.time.LocalDateTime;
4 import java.util.*;
5
6 public class Auktion extends Observable {
7     public Integer auktionsNr = 0;
8     public String arikelName = "";
9     public String beschreibung = "";
10
11     public List<Gebot> gebote = new ArrayList<>();
12     public LocalDateTime AuktionsEnde;
13
14     public Auktion(Integer auktionsNr, String arikelName, String beschreibung, LocalDateTime
        auktionsEnde) {
15         auktionsNr = auktionsNr;
16         arikelName = arikelName;
17         beschreibung = beschreibung;
18         auktionsEnde = auktionsEnde;
19
20         setChanged();
21     }
22
23     public void setzeGebot(Benutzer user, float preisInEuro) {
24         if(AuktionsEnde.isAfter(LocalDateTime.now())) {
25             if(gebote.isEmpty() || (preisInEuro > gebote.get(gebote.size()-1).preis)) {
26                 Gebot gebot = new Gebot();
27                 gebot.preis = preisInEuro;
28                 gebot.benutzer = user;
29
30                 gebote.add(gebot);
31                 user.addToAktiveAuktionen(this);
32
33                 setChanged();
34                 notifyObservers(gebote);
35             }
36         }
37     }
38 }
```

Listing 6: Auktion.java