Rheinisch Westfälische Technische Hochschule Aachen
Lehrstuhl für Software Engineering

Best Practices of Modern and Efficient Software Engineering

Anonyme und innere Klassen in Java und ihre Verwendungsmuster

Proseminar

von

Birk, Peter Jakobi, Felix

1. Prüfer: Prof. Dr. B. Rumpe

2. Prüfer: Dipl.-Inform. Deni Raco

Betreuer: Marlene Lutz

Diese Arbeit wurde vorgelegt am Lehrstuhl für Software Engineering

Aachen, den 13. Dezember 2017

Eidesstattliche Versicherung

Name, Vorname

Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum

Unterschrift

Kurzfassung

Eine kurze Zusammenfassung der Arbeit.

Gliederung:

1. Einfuehrung
   (a) Idee/Herkunft
   (b) Innere Klassen
   (c) Anonyme Klassen
   (d) Entwicklung/Geschichte
2. Hauptteil
   (a) (Statische) Innere Klassen
      i. Implementierung
      ii. Vorteile
      iii. Nachteile
      iv. Verwendungsmuster
   (b) Anonyme Klassen
      i. Implementierung
      ii. Vorteile
      iii. Nachteile
      iv. Verwendungsmuster
3. Zusammenfassung/Fazit
4. Literaturverzeichnis

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Idee / Herkunft

Kapitel 2

Hauptteil

2.1 (Statische) Innere Klassen

2.1.1 Implementierung

```
1  class outerA {
2    class innerA{
3      void doSomething(){
4      }
5    }
6    void vC (){
7      class localvC{
8        void doSomethingDifferent(){
9        //...
10       }
11     }
12   }
13 }
14 class outerB{
15
16   static class innerB{
17
18   }
19 }
```

Listing 2.1: Beispielimplemntierung von inneren Klassen in Java

Eine Innere Klasse wird in Java durch das Erstellen einer Klasse innerhalb einer Klasse implementiert (Siehe Beispiel 2.1). Der Name muss unterschiedlich sein. Dabei kann diese mit allen Modifiers versehen werden, die auch einer äußeren Klasse zur Verfügung stehen - sie darf aber auch private, protected oder static sein, da sie letzten Endes ein Attribut der äußeren Klasse ist. Auch Vererbung ist möglich. Die IK hat im Normalfall Zugang zu allen Methoden und Variablen der äußeren Klasse - sogar zu denen, die als private deklariert wurden. Es ist weiterhin möglich, IKn zu schachteln.

Es können in einer IK Attribute implementiert werden. Dabei werden gleichnamige Attribute der äußeren Klasse wie bei Vererbung von der IK überdeckt. Jedoch erlaubt Java keine statischen Attribute, da die IK ein instanziertes Attribut der äußeren Klasse ist.

Eine große Unterscheidung muss gemacht werden, wenn eine IK statisch ist. In diesem Fall greift die IK aus einem statischen Kontext auf die Attribute der äußeren Klasse zu - und kann somit nur statische Methoden und Variablen benutzen. Im Endeffekt sind dann auch die Attribute der statischen inneren Klasse statisch.

Instanziierung

```
1
2 outerA A = new outerA();)
3 outerA.innerA a = A.new innerA();
4 // Instanziierung eines Objekts der Inneren Klasse innerA
5
6 outerB.innerB b = new outerB.innerB();
7 // Instanziierung eines Objekts der Statischen Inneren Klasse innerB
```

Listing 2.2: Beispielinstanziierung von inneren Klassen in Java

Eine normale Innere Klasse ist an ihre äußere Klasse gebunden und muss daher mit einem Objekt der äußeren Klasse instanziiert (2.2 *InstNInn*). Bevor der Konstruktor der Inneren Klasse aufgerufen wird, muss also zuerst ein Konstruktor der äußeren Klasse aufgerufen werden. Im Gegensatz dazu benötigt eine statische Innere Klasse keine Instanz der äußeren Klasse. In diesem Fall darf einfach ein Konstruktor der statischen Inneren Klasse im Namensraum der äußeren Klasse genutzt werden (2.2 *InstInn*).

Vererbung

Obwohl klassische Mehrfachvererbung in Java nicht möglich ist, ist es IKn erlaubt, von anderen Klassen zu erben. Dies ist potenziell fehleranfällig. Der Compiler erkennt zwar simple Fehler, aber diese Fehlererkennung hält sich in engen Grenzen (Listing x). Jedoch bringen IKn auch die Möglichkeit einer Mehrfachvererbung ohne Interfaces, da die IK Zugriff auf ihre äußere Klasse hat sowie zusätzlich von einer externen Klasse erben kann. Falls es gleichnamige Attribute gibt, wird das Attribut der äußeren Klasse vom Attribut der Oberklasse verdeckt(**??**lst:InstInn]). Wenn man dieses Prinzip weiter verfolgt, kann man potenziell mit weiteren verschachtelten IKn von beliebig vielen weiteren Klassen erben.

Kapitel 3

Code Listings

This chapter contains the beautiful listing 3.1. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

```
component AdverseDrugReactionApp {
  autoconnect port;
  port
    in Image barcode,
    out String information;

  component MobileEHealthApp
    eHealthApp [bc2Service -> bcSer.Image];
  component BarcodeScannerService
    bcSer [String -> eHealthApp.bcAsString];
  component EHealtServer;
  component AdverseReactionDataBase;

  connect EHealtServer.result -> eHealthApp.answer;
}
```

Listing 3.1: Code listing with user defined syntax highlighting (MontiArc).

Kapitel 4

Zusammenfassung und Ausblick

Abbildung 4.1: Das SE Logo

Anhang A

z. B. Programmdokumentation