FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Übung 6 zur OOS SS16 Bearbeitung bis 12.05.2016

Aufgabe 27: (Zeichnen eines Fensters mit vorgestellten Elementen)

Erzeugen Sie eine Datei `BeispielFrame.java`, die folgendes Fenster darstellt:

a) Besetzen Sie das obere Textfeld mit Ihrem Namen vor.

b) Besetzen Sie das untere Textfeld mit "Passwort" vor und überprüfen Sie, was bei der Darstellung erscheint.

c) Setzen Sie den Zustand der CheckBox auf true und überprüfen Sie, was bei der Darstellung erscheint.

d) Klicken Sie in der Darstellung auf die CheckBox und überprüfen Sie, was sich dabei tut.

e) Klicken Sie in der Darstellung auf den Button und überprüfen Sie, was sich dabei tut.

Aufgabe 28: (Ereignisverarbeitung)

Verwenden Sie die Datei `BeispielFrame.java` aus Aufgabe 27, die obiges Fenster darstellt.

a) Schreiben Sie eine lokale Listener-Klasse, die das Interface `ActionListener` so implementiert, dass bei Drücken des "Ausführen"-Buttons die Inhalte des Textfeldes neben "Name" und des Passwortfeldes neben "Passwort" verglichen werden und das Resultat dieses Vergleiches auf die Standardausgabe geschrieben wird.

b) Erzeugen Sie ein Objekt der Klasse aus a) und weisen Sie es dem JButton des Fensters zu. Überprüfen Sie, ob Sie den Listener korrekt implementiert haben.

c) Schreiben Sie eine anonyme Listener-Klasse für die CheckBox, die das Interface `ActionListener` so implementiert, dass beim Setzen der

FH AACHEN UNIVERSITY OF APPLIED SCIENCES

CheckBox`true` ausgegeben wird und beim Zurücksetzen der CheckBox `false` ausgegeben wird.

d) Schreiben Sie eine anonyme Listener-Klasse für die CheckBox, die das Interface `ActionListener` so implementiert, dass Sie Schreibrecht auf das Textfeld neben "Name" nur gibt, wenn die CheckBox gesetzt ist.

Aufgabe 29: (anonyme Klassen)

a) Ändern Sie das Interface `BenutzerVerwaltung` aus dem Praktikum so ab, dass es zu einer abstrakten Klasse `Bv` wird, indem die Methode `boolean benutzerOk(Benutzer benutzer)` so implementiert wird, dass sie immer `false` zurückgibt.

b) Ändern Sie die Klasse `Test`, die analog zur Vorlesung wie folgt definiert ist:

```
class Test {
  public boolean meth (Benutzer ben, BenutzerVerwaltung bv) {
    return(bv.benutzerOk(ben));
  }
}
```

so dass der zweite Parameter von der abstrakten Klasse aus a) ist.

c) Ändern Sie die Implementierung der main-Methode auf der Folie "Ereignisverarb. 7" so ab, dass die anonyme Klasse im 2. Parameter der Methode `meth` die abstrakte Klasse aus a) implementiert und die Implementierung völlig analog zu der auf Folie "Ereignisverarb. 7" ist.

d) Welche Ausgabe liefert die in c) abgeänderte main-Methode. Begründen Sie Ihre Antwort.

Aufgabe 30: (Adapterklasse)

Implementieren Sie eine Adapterklasse zu dem Interface `BenutzerVerwaltung`.

Aufgabe 44: (generische Datentypen)

a) Implementieren Sie eine generische Klasse `Box`, die ein generisches Datum kapselt. Gehen Sie hierzu wie folgt vor:

i) Definieren Sie ein privates Attribut des generischen Datums.
ii) Implementieren Sie einen Konstruktor, der das generische Datum vorbesetzt.
iii) Stellen Sie getter und setter für das generische Datum zur Verfügung.

b) Erzeugen Sie in einem `main`-Programm einer anderen Klasse ein Objekt der generischen Klasse aus a), wobei Sie als generisches Datum ein Objekt der Klasse `Benutzer` verwenden und geben Sie das generische Datum mit `println` aus.

Übung 6 zur OOS SS16 **Bearbeitung bis 12.05.2016**

c) Implementieren Sie eine generische Klasse `EingeschränkteBox`, die analog zur Klasse `Box` implementiert ist, jedoch als generisches Datum lediglich Klassen zulässt, die das Interface `BenutzerVerwaltung` implementieren.

d) Erzeugen Sie in dem `main`-Programm aus b) zwei Objekte der Klasse aus c), wobei Sie jeweils ein generisches Datum des folgenden Typs verwenden:
`BenutzerVerwaltung` und `BenutzerVerwaltungAdmin`.

e) Kann man das 2. Objekt aus d) einer Variablen des 1. Typs aus d) zuweisen?

Aufgabe 55: (weitere Baummethode)

Erweitern Sie die Baumimplementierung gemäß des Composite-Patterns, die sich im Paket `composite` im Download-Übungsverzeichnis befindet, um die beiden Methoden `knotenAnzInLevel`, die die Anzahl der Knoten des Baumes in einem beliebigen Level ermittelt und `knotenInLevel`, die die Einträge der Knoten des Baumes in einem beliebigen Level von links nach rechts ausgibt. Dabei hat die Wurzel das Level 0.

Aufgabe 56: (Lineare Liste nochmal nutzen)

Implementieren Sie in der Klasse aus Aufgabe 54, die Ihre Implementierung der linearen Liste verwendet, noch die folgenden statischen Methoden:

`präfix:` überprüft, ob eine Liste mit einer anderen Liste beginnt

`postfix:` überprüft, ob eine Liste mit einer anderen Liste endet

`infix:` überprüft, ob eine Liste eine andere Liste enthält

`contains:` überprüft, ob eine Liste ein vorgegebenes Objekt enthält

Design-Aufgabe 2: (Java-Event-Handling) erst in Übung

a) Entwerfen Sie ein Design in UML, so dass das Java-Event-Handling implementiert wird.
Gehen Sie hierzu möglichst allgemein vor, so dass man Ihr Design auch noch zur Lösung ähnlicher Probleme verwenden kann.
Lösen Sie sich deshalb von den konkreten Frameklassen und den Eventlistenern und geben Sie diesen allgemeinere Namen.
Gehen Sie davon aus, dass der Zustand eines Frames durch eine int-Zahl dargestellt wird und dass jede Änderung dieses Zustandes den EventListenern mitgeteilt wird und dort eine Reaktion ausgelöst wird.

b) Geben Sie eine Implementierung Ihres Designs aus a) an.