Prof. Dr. A. Weber
Dr. Hassan Errami

Universität Bonn
Institut für Informatik II
16.05.2019

Sommersemester 2019

Übungen zur Vorlesung
Objektorientierte Softwareentwicklung (BA-INF-024)
Aufgabenblatt 6 (30 Punkte)

Zu bearbeiten bis: 31.05.2019

In der Woche vom 20.05 - 23.05 findet die erste Online-Aufgabe statt.

Aufgabe 1 *(Listen - 3+5*2+2=15 Punkte)*

a) Implementieren Sie eine verkettete Liste über double-Werten in Java. Erstellen Sie dazu eine Klasse *DoubleNode*, die einen double-Wert sowie den jeweiligen Nachfolger vom Typ *DoubleNode* speichert. Schreiben Sie passende getter und setter-Methoden sowie zwei Konstruktoren, denen Sie entweder nichts oder direkt den double-Wert übergeben können. Schreiben Sie außerdem eine Klasse *DoubleList*, die eine *DoubleNode* (head) speichert, die den Kopf der jeweiligen Liste repräsentiert.

b) Implementieren Sie eine Methode *add(double d)*, die den Wert d hinten in die Liste einfügt. Achten Sie darauf, keine NullPointerException auszulösen, wenn die Liste noch leer ist.

c) Implementieren Sie eine Methode *insertFirst(double d)*, die den Wert d vorne in die Liste einfügt.

d) Implementieren Sie eine Methode *get(int i)*, die den Wert an der i-ten Stelle in der Liste zurückgibt. Achten Sie wieder darauf, dass es zu keiner Exception kommt, falls i ungültig gewählt wurde.

e) Implementieren Sie eine Methode *toString()*, die alle Werte in der Liste durch „;“ getrennt ausgibt.

f) Implementieren Sie eine Methode *remove(int i)*, die den an der i-ten Stelle stehenden Wert entfernt. Achten Sie wieder auf ungültige Werte für i.

g) Schreiben Sie eine Klasse *testClass*. Erstellen Sie eine *DoubleList*, fügen Sie 3 beliebige Werte mittels *add* und 2 beliebige Elemente mittels *insertFirst* ein und löschen Sie erst den in der Mitte stehenden, dann den vordersten und dann den letzten Wert. Rufen Sie auch mittels *get* das vorderste Element ab und geben Sie den Wert aus. Prüfen Sie durch Ausgabe der Ergebnisse Ihrer *toString*-Methode ob Ihre übrigen Methoden korrekt funktionieren.

Aufgabe 2 *(Vererbung bei Exceptions - 5 Punkte)*

Gegeben sei folgender Code:

```
import java.io.*;
public class Hamburger extends Fastfood {
    // Hier code einfügen
}

import java.io.IOException;
class Fastfood {
    void eat() throws IOException { }
}

import java.io.IOException;
public class Maincl {
    public static void main(String[] args) throws IOException {
    Hamburger c=new Hamburger();
    c.eat();
    }
```

```
}
```

Erläutern Sie, für jede der folgenden Methodendeklarationen, warum eine Kompilierung möglich oder nicht möglich ist, wenn man die Codeteile an der entsprechend markierten Stelle einfügt.

```
a) void eat() { }

b) void eat() throws Exception { }

c) void eat(int y) throws Exception { }

d) void eat() throws FileNotFoundException { }

e) void eat() throws RuntimeException { }
```

Aufgabe 3 *(Geprüfte vs. ungeprüfte Exception - 10 Punkte)*

In dieser Aufgabe sollen Sie zwei eigene Ausnahmebehandlungen implementieren. Schreiben Sie hierzu eine eigene NumberTooBigException, die von RuntimeException erbt und beim Aufruf eine entsprechende Fehlermeldung ausgibt.
Schreiben Sie nun ein Programm, was eine Zahl vom Benutzer einliest und überprüft, ob die Zahl größer fünf ist. Sollte die Zahl größer als fünf sein, wird die NumberTooBigException geworfen.
Implementieren Sie eine weitere NumberTooSmallException, die diesmal von Exception erbt. Sie soll ausgelöst werden, wenn der Nutzer eine Zahl kleiner fünf eingibt. Was beobachten Sie? Was ist der Unterschied zwischen einer geprüften und ungeprüften Exception?