Dr. Hassan Errami

Universität Bonn
Institut für Informatik II
15.04.2021

Sommersemester 2021

Praktikum: Objektorientierte Softwareentwicklung (BA-INF-025)

Aufgabenblatt 1 (15 Punkte)

Zu bearbeiten bis: 24.04.2021

Schicken Sie bitte alle Ihre Lösungen in einem ZIP-Archiv per E-Mail an Ihren Tutor. Die E-Mail-Adresse von Ihrem Tutor finden Sie am Ende dieses Blattes. Den Link zu dem Online-Tutoriumraum Ihres Tutors finden Sie auf der Praktikumswebseite:
`https://cg.cs.uni-bonn.de/de/lehre/2021-ss/oose/`

Hinweise: Dieses Übungsblatt soll individuell von Ihnen bearbeitet werden. Sie sollten, um die Punkte dieser Aufgaben zu erhalten, Ihre Lösungen Ihrem Tutor präsentieren können.

Aufgabe 1 *(Comperatoren und Comparable - 2+2+2 Punkte)*

In dieser Aufgabe geht es um die Schnittstellen Comparable und Comparator. Falls Sie mit den beiden Schnittstellen noch nicht vertraut sind, können Sie die offizielle Dokumentation[1] der Schnittstellen oder ein Tutorial[2] zur Hand nehmen.

a) Schreiben Sie eine Klasse, die eine Person repräsentiert. Eine Person hat:

- einen Vornamen
- einen Nachnamen
- ein Alter

Die Klasse soll außerdem über eine Methode print() verfügen, welche die Daten der Person ausgibt. Die Klasse soll die Schnittstelle java.lang.Comparable implementieren. Dabei soll als erstes der Nachname miteinander verglichen werden. Ist dieser gleich, dann der Vorname und zum Schluss das Alter.

b) Schreiben Sie eine Klasse PersonenSortierer, welche die Schnittstelle java.util.Comparator implementiert. Die Reihenfolge der verglichenen Attribute soll dabei anders sein. Vergleichen Sie zuerst den Vornamen, dann den Nachnamen und wenn auch dieser gleich ist das Alter.

c) Legen Sie mehrere Objekte der Klasse Person an und halten Sie diese in einer Datenstruktur, welche die Schnittstelle java.util.List implementiert. Sortieren Sie die Liste zweimal. Nutzen Sie dabei die beiden Sortiermethoden aus java.util.Collections. Sie sollen also einmal mit Hilfe der Comparator-Schnittstelle und einmal mit der Comparable-Schnittstelle sortieren. Geben Sie nach der Sortierung jeweils alle Elemente aus. Suchen Sie außerdem das maximale und minimale Element aus der List mithilfe passender Methoden aus der Collections-Klasse raus, jeweils mit Hilfe der Comparator-Schnittstelle und mit Hilfe der Comparable-Schnittstelle.

Aufgabe 2 *(HashMap - 4 Punkte)*

Informieren Sie sich über die Schnittstelle java.util.Map[3].

a) Nutzen Sie die Klasse Person aus der vorherigen Aufgabe. Legen Sie eine java.util.HashMap an. Fügen Sie der HashMap mehrere Personen hinzu. Nutzen Sie als Schlüssel den Vollständigen Namen der Person. Iterieren Sie über die HashMap und geben Sie alle in der HashMap befindlichen Objekte aus.

b) Begründen Sie: Was geht schneller, das Laden eines Personen Objekts aus einer HashMap mit gegebenem vollständigem Namen oder das Durchsuchen einer Liste?

c) Nennen Sie weitere Möglichkeiten um über eine Map zu iterieren als die, die Sie verwendet haben.

[1] `https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html` und `https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html`

[2] `http://javatricks.de/tricks/objekte-vergleichen-mit-comparable-und-comparator`

[3] `https://docs.oracle.com/javase/8/docs/api/java/util/Map.html` oder `https://www.callicoder.com/java-hashmap/`

Aufgabe 3 *(Threads in Java - 4+1 Punkte)*

Falls Sie sich nicht mehr sicher sind, wie Threads funktionieren, können Sie sich im Internet informieren[4].

a) Erstellen Sie eine Klasse PrimzahlRechner, die von der Klasse Thread erbt. Diese soll Primzahlen in einem bestimmten Bereich suchen und die gefundenen Primzahlen dann in einer Liste speichern. Der Suchbereich wird der Klasse über den Konstruktor übergeben. Nutzen Sie vier Instanzen der Klasse PrimzahlRechner um Primzahlen parallel zu berechnen. Geben Sie anschließend die Primzahlen in aufsteigender Reihenfolge aus. Berechnen Sie so die Primzahlen zwischen 0 und 4000.

b) Welche Möglichkeit gibt es noch um Threads zu implementieren ohne von der Klasse Thread zu erben?

Extra-Aufgabe 4* *(Synchronisieren von Threads - 3 Bonuspunkte)*

Mit Threads können zeitaufwändige Berechnungen beschleunigt werden. Hierbei wird die Berechnung auf mehrere Threads verteilt, die anschließend parallel ausgeführt werden. Ein Problem ergibt sich allerdings, wenn mehrere Threads auf geteilte Ressourcen zugreifen. Einerseits können sich die Threads verklemmen, andererseits kann es auch sein, dass das Programm funktioniert, aber ein unerwartetes Ergebnis produziert. Dies wollen wir im folgenden näher untersuchen.

Schreiben Sie eine Klasse Zähler. Die Klasse soll eine Instanzvariable **counter** enthalten. Die Variable soll mittels der Methode inkrementiereCounter immer um den Wert eins erhöht werden können bis die Zahl 1000 erreicht ist. Desweiteren soll bei jedem Aufruf der aktuelle Zählerstand ausgegeben werden (nicht in der gleichen Zeile, in der die Variable ausgegeben wird). Damit wir identifizieren können, welcher Thread die Methode aufgerufen hat, geben Sie

```
Thread.currentThread().getName()
```

mit aus.

Schreiben Sie eine Klasse Inkrementierer, welche die Schnittstelle java.lang.Runnable implementiert. Die Klasse soll bei Erzeugung eine Instanz der Klasse Zähler übergeben werden können. Der Inkrementierer soll die Methode inkrementiereCounter der Klasse Zaehler aufrufen, bis die Zahl 1000 erreicht ist.

Legen Sie vier Threads an, die alle die gleiche Instanz der Klasse Zaehler übergeben bekommen und starten Sie diese. Lassen Sie das Programm mehrmals laufen. Was beobachten Sie? Wie kommt die Ausgabe zustande?

a) Informieren Sie sich über das Synchronisieren von Threads in Java[5].

b) Ändern Sie das Programm so ab, dass die Ausgabe der Zahlen in aufsteigender Reihenfolge erfolgt. Dabei soll das Hochzählen des Zählers aber immer noch durch mehrere Threads erfolgen.

Tutor/in	Email
Adrian Bajraktari	adrian.bajraktari@uni-bonn.de
Kristi Balla	kristigjermani@gmail.com
Bela Bothin	belabothin@gmail.com
Jakob Hafke	jakob.hafke@gmx.de
Christopher Sander	chris.sander97@googlemail.com
Sarah Sturm	sarah@sarah-sturm.de

[4] https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html oder https://www.tutorialspoint.com/java/java_multithreading.htm

[5] http://openbook.rheinwerk-verlag.de/javainsel9/javainsel_14_005.htm#mjdd7f4718142f506c26dd21b125c3eeb9