Hausaufgabe 3

Funktionale Interfaces und Scala Basics

Abgabe bis 10. Juni 2024, 23:59 Uhr
(maximal erreichbare Punktzahl: 20 Punkte)

3.1 Java: Funktionale Interfaces (8 Punkte)

Ihnen ist im Ordner `src/main/java` die Klasse `FunctionalInterfaces` im Package `impl` gegeben. Diese Klasse umfasst eine `main`-Methode sowie vier weitere statische Methoden, die von der `main`-Methode aufgerufen werden.

In der `main`-Methode sind einige Zeilen auskommentiert, damit das Projekt kompiliert werden kann. Kommentieren Sie die markierten Zeilen wieder ein und betrachten Sie die Methodenaufrufe. Nach dem Einkommentieren ist die Klasse fehlerbehaftet, da die vier aufgerufenen Methoden nicht die korrekte Signatur für die jeweiligen Aufrufe haben. Ihre Aufgabe ist es, dies zu beheben.

Es wird bei den Methodenaufrufen jeweils ein Lambda-Ausdruck übergeben, jedoch fehlt den Methoden der entsprechende Parameter, um das funktionale Interface zu verarbeiten. Stattdessen erwarten alle Methoden einfach ein `Object`. Bestimmen Sie für jede Methode, welches funktionale Interface es als Parameter erwarten müsste und ersetzen Sie das `Object` durch das entsprechende Interface.

Nachdem Sie die Methodensignaturen korrigiert haben, kompiliert das Programm. Es mangelt allerdings noch an Funktionalität. Implementieren Sie daher die Methoden so, dass das übergebene funktionale Interface angewendet wird, um die erwartete Rückgabe zu generieren.

Sie können Ihre Lösung mithilfe der Testklasse `TestFunctionalInterfaces` überprüfen. Kommentieren Sie dazu zuerst die markierten Zeilen in der Testklasse ein, damit die Tests auf Ihre Lösung zugreifen!

3.2 Scala: Grundlagen (12 Punkte)

Scala ist sehr gut für die Verarbeitung von Daten geeignet. In dieser Aufgabe betrachten wir beispielhaft einen Datensatz von Prüfungsanmeldungen und -ergebnissen und wollen Scala einsetzen, um Erkenntnisse aus den Daten zu gewinnen.

Der Datensatz ist in zwei Dateien unterteilt, welche sich im `resources`-Ordner des Projekts befinden: `registrations.csv` enthält eine Liste aller Prüfungsanmeldungen mit dem Namen der Student*innen, der jeweiligen Matrikelnummer und der Angabe, um den wievielten Versuch es sich handelt. `results.csv` enthält eine Liste der Prüfungsergebnisse mit der jeweiligen Matrikelnummer und Note.

Ihnen ist im Ordner `src/main/scala` die Klasse `ExamRegistrations` im Package `impl` gegeben. In dieser Klasse finden Sie sechs Methoden, die von Ihnen implementiert werden sollen.

Nutzen Sie für Ihre Implementierung die `for`-Notation!

1. `readRegistrationsFromFile` - Liest die übergebene CSV-Datei aus und überträgt die gelesenen Daten in eine Liste von Tupeln, wobei jede Zeile der CSV-Datei und jedes Tupel der Liste jeweils eine Prüfungsanmeldung repräsentieren. Die Tupel haben das Format (<Name> : String, <Matrikelnummer> : Int, <Versuch> : Int).

2. `readResultsFromFile` - Liest die übergebene CSV-Datei aus und überträgt die gelesenen Daten in eine Liste von Tupeln, wobei jede Zeile der CSV-Datei und jedes Tupel der Liste jeweils ein Prüfungsergebnis repräsentieren. Die Tupel haben das Format (<Matrikelnummer> : Int, <Note> : Double).

3. `findBestStudent` - Ermittelt die Student*in mit der besten Note und gibt den Namen der Student*in zurück.

4. `findAdequateStudents` - Ermittelt alle Student*innen, die im zweiten Versuch eine 3.0 oder besser geschrieben haben und gibt eine Liste ihrer Matrikelnummern zurück.

5. `findDuplicateRegistrations` - Ermittelt die Student*innen, die sich mehrfach angemeldet haben und daher mehrmals in der Liste von Prüfungsanmeldungen auftauchen und gibt eine Liste ihrer Matrikelnummern zurück.

6. `findAvgGradeOfRepeatedExams` - Berechnet die Durchschnittsnote aller Student*innen, die die Prüfung im Zweitversuch schreiben.

Sie können Ihre Lösung mithilfe der in der Testklasse `TestExamRegistrations` gegebenen Tests überprüfen.

Hinweis: eventuell kann die `TestExamRegistrations`-Klasse nicht sofort ausgeführt werden. Sollte dies bei Ihnen der Fall sein, können Sie das Problem wie folgt beheben (vorausgesetzt, Sie haben das Scala-Plugin für IntelliJ installiert!):

- Navigieren Sie in IntelliJ zu Einstellungen unter
  `File -> Settings`
- Öffnen Sie die Gradle-Einstellungen unter
  `Build, Execution, Deployment -> Build Tools -> Gradle`
- Setzen Sie im Abschnitt `Build and Run` die Einstellung `Run tests using:` auf
  `IntelliJ IDEA`

Abgabe

Laden Sie Ihre Lösung bis 23:59 Uhr am 10.06.2024 als ZIP-Archiv bei ISIS hoch. Als Dateinamen für das Archiv verwenden Sie bitte `{VornameNachname}.zip`.

Achten Sie unbedingt darauf, dass die Ordnerstruktur Ihrer Abgabe exakt der Struktur der Vorgabe entspricht! Sollten Sie von dieser Struktur abweichen, kann Ihre Abgabe schlimmstenfalls nicht bewertet werden. Ihre Ordnerstruktur sollte wie folgt aussehen:

```
{VornameNachname}.zip
| - ha03-vorgabe
| | - src
| | | - main
| | | | - java
| | | | | - impl
| | | | | | FunctionalInterfaces.java
| | | | - scala
| | | | | - impl
| | | | | | ExamRegistrations.scala
| | | | ...
| | | ...
| | ...
```

Bitte entfernen Sie vor dem Hochladen ggf. die Ordner `.idea/` und `build/` aus Ihrer Abgabe. Sie werden bei der Bewertung ignoriert und blähen Ihre Abgabe nur unnötig auf.