Programmrealisierung mit Java

1. Gegenstand der Einstufungstests

Es soll ein kleines Adress-System mit Hilfe der Standard-Java-Platform (Java SE 8+) entwickelt werden unter Berücksichtigung von Konventionen, Code-Qualität, Fehlertoleranz und mit Hinblick auf zukünftige Wartung (Code-Klarheit, Redundanzfreiheit etc.)

Die genauere Aufgabenbeschreibung befindet sich ab Seite 2, bitte lesen Sie sich auch die detaillierteren Angaben zur Spezifikation gründlich durch. In der wesentlichen Implementierungsgestaltung sind Sie frei, genauere Vorgaben existieren aber für einige Elemente, die es zu beachten gilt!

Die Aufgaben bauen aufeinander auf, sie können aber teilweise in verschiedenen Reihenfolgen realisiert werden. Sollte die Ausgabe von Dateien aus Zeitgründen noch nicht Gegenstand des Seminars gewesen sein, dürfen Sie die Ausgabe auf dem Bildschirm simulieren. Sollte die Eingabe von Dateien noch nicht Gegenstand gewesen sein, dürfen Sie die Dateidaten mit Strings simulieren. Auf diese Lösungen dürfen Sie auch zurückgreifen, wenn Sie Probleme mit dem Dateihandling haben und sonst die Aufgaben nicht bearbeiten könnten.

2. Hilfsmittel

Während des Tests dürfen Sie im Prinzip alle Ressourcen nutzen, insbesondere könnte die Java Dokumentation hilfreich sein. Unerwünscht ist der persönliche oder digitale Kontakt zu anderen Personen, ausgenommen natürlich dem Dozenten.

Technische Vorgaben: mindestens Java 8. Umsetzung in einer IDE (z.B. Eclipse).

3. Ergebnis

Das Endresultat packen Sie bitte in eine ausführbare JAR-Datei, die auch den Sourcecode und die von Ihnen verwendete Testdateien (siehe Aufgabenstellung) enthält. Benennen Sie die JAR-Datei mit Ihrem Namen.

Wichtig: Überprüfen Sie vor Abgabe, ob sich die Sourcen und die Eingabedatei ebenfalls im erzeugten JAR befinden! JAR-Dateien können mit jedem ZIP-Tool (z.B. 7-ZIP oder WinZIP) analysiert werden oder natürlich auch mit dem Kommandozeilentool jar.

4. Bearbeitungszeit

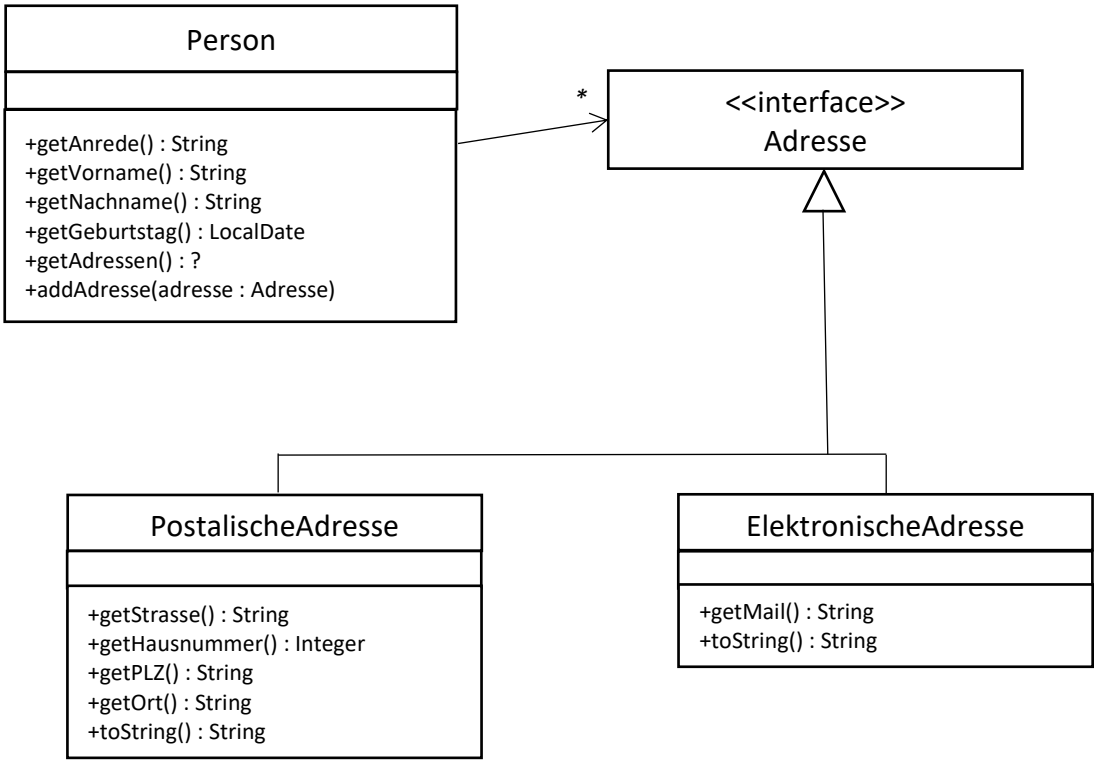
2 Stunden.

Aufgabenstellung: Personen und Adressen

Es wird eine grundlegende Datenstruktur aufgebaut, die es ermöglicht, Personen mit mehreren Adressen beliebigen Typs abzubilden, auszugeben und in Dateien ein- und auszugeben.
Hierzu werden Sie in den Aufgaben angeleitet, verschiedene Pakete zu machen. Diese sollen den jeweiligen Stand der Aufgaben voneinander abtrennen und stellen keine sinnvolle Modularisierung im Sinne einer „guten" Java-Softwarearchitektur dar.

UML-Diagramm

Das folgende UML-Diagramm gilt als Rohentwurf bzw. grundlegende Vorgabe. Es ist nicht vollständig, da hinsichtlich Attribute oder Konstruktoren gar keine Vorgaben gegeben sind.



Für die Datumsdarstellung benutzen Sie bitte `java.time.LocalDate` und keine veralteten Alternativen wie `java.util.Date` oder `java.util.Calendar`. Ggf. recherchieren Sie in der Java-Dokumentation, wie mit dieser Klasse umzugehen ist.

Für den Umgang mit Dateien benutzen Sie bitte das `java.nio.files`-API und nicht die veraltete Klasse `java.io.File`.

Aufgabe 1 (6 Punkte)

Erstellen Sie ein Paket `de.allianz.einstufung.test.a`.

Schreiben Sie die Klasse Person, wie im UML-Diagramm angegeben und vervollständigen Sie diese mit Attributen und Konstruktoren. Lassen Sie für diese Aufgabe erst einmal alle Zusammenhänge mit dem Typ `Adresse` weg.

Aufgabe 2 (4 Punkte)

Schreiben Sie in dem eben erstellten Paket einen Test in einer `main`-Methode, der 2 Instanzen von Person anlegt und diese ausgibt.

Aufgabe 3 (16 Punkte)

Erstellen Sie ein Paket `de.allianz.einstufung.test.b`.

Kopieren Sie die Klassen aus Paket a in Paket b.

Ergänzen Sie das Interface `Adresse` und schreiben Sie die Klasse `ElektronischeAdresse` inkl. sinnvoller Attribute und Konstruktoren, sowie einer passenden `toString()`-Methode. Implementieren Sie `getAdressen()` und `addAdresse()` in Person, so dass `addAdresse()` jeweils eine neue Adresse aufnimmt und `getAdressen()` alle bekannten Adressen dieser Person in einer sinnvollen Datenstruktur zurückgibt.

Ergänzen Sie den Test aus Aufgabe 2, jetzt in Paket b, um eine Mail bei der ersten Person und zwei Mailadressen bei der zweiten Person. Geben Sie diese auf dem Bildschirm aus.

Aufgabe 4 (13 Punkte)

Schreiben Sie im Paket b die Klasse `PostalischeAdresse` inkl. sinnvoller Attribute und Konstruktoren, sowie einer passenden `toString()`-Methode für die Ausgabe.

Kopieren Sie die Testklasse aus Aufgabe 3 und ergänzen in dieser Kopie jeweils eine postalische Adresse für beide Testpersonen und geben Sie diese aus.

Aufgabe 5 (14 Punkte)

Erstellen Sie ein Paket `de.allianz.einstufung.test.c` und kopieren Sie den Inhalt von Paket b dort hinein.

In diesem Abschnitt wollen wir mehrere Personen erzeugen und diese in einer Datei abspeichern.

Erstellen Sie dazu eine Klasse (sinnvoll benannt), der Sie in einer Klassenmethode (sinnvoll benannt) mehrere Personen übergeben können und die eine Datei `personen.txt` erzeugt, mit folgendem Format:

```
<anrede>,<vorname>,<nachname>,<geburtsdatum als tt.mm.jjjj>
```

Für jede Person wird eine Zeile erzeugt.

Schreiben Sie ein Startprogramm, das mehrere Personen erzeugt und die Funktionalität aufruft.

Aufgabe 6 (10 Punkte)

Ab jetzt beginnen die Aufgaben für etwas fortgeschrittene Programmierer. Sind Sie noch dabei?

Kopieren Sie das Paket `de.allianz.einstufung.test.c` in ein neues Paket d.

Jetzt wird es interessant. Wir wollen auch Adressen speichern, diese aber in einer zweiten Datei `adressen.txt`.

Ergänzen Sie die Klassenmethode für das Schreiben der Datei um alles, was nötig ist, die Adressen in einer zweiten Datei (gleichzeitig) auszugeben. Das Format könnte so aussehen:

`<strasse>,<hausnummer>,<plz>,<ort>`

oder

`<email@domain>`

je nachdem, welche Art Adresse vorliegt.

Testen Sie auch dieses Verhalten mit einer Ergänzung zum unter c geschriebenen Test.

Aufgabe 7 (17 Punkte)

Hinweis: es ist nicht nötig, die Aufgaben 5-7 oder 8 gelöst zu haben, es genügt auch ein von Hand erstelltes Datenmodell.

Kopieren Sie Ihre bisherige Lösung in ein neues Paket e.

In dieser Aufgabe wollen wir analytisch mit unseren vorhandenen Daten umgehen und statistische/fachliche Auswertungen anbieten.

Erstellen Sie eine neue Klasse (sinnvoll benannt), in der Sie die fachliche Auswertungen machen.

1. In einer Klassenmethode soll ermittelt werden, welche Personen einen Bindestrich im Nachnamen haben, also einen Doppelnamen tragen. Gesucht sind also mehrere Objekte des Typs `Person`.
2. In einer weiteren Klassenmethode soll ermittelt werden, wie viele Personen mindestens n hinterlegte Adressen beliebigen Typs haben, $n \geq 1$. Gesucht ist hier eine Zahl.

Sofern Sie im Kurs bereits Lambda-Ausdrücke und das Stream-API besprochen haben, lösen Sie die Aufgabe bitte damit. Ansonsten auf „klassischem“ Weg.

In der Testklasse im Paket e erzeugen Sie das Datenmodell und werten die Daten mit den neu angelegten Analysemethoden aus und stellen die Ergebnisse auf dem Bildschirm dar.