Aufgabenstellung für das *erste* Übungsbeispiel der LVA „Objektorientiertes Programmieren“

Abgabeschluss für Source Code: 21.04.2016!

WICHTIG: Von Studierenden, deren Beispiel nicht spätestens am 21.04.2016 um 23:55 Uhr in TUWEL zur Verfügung steht, kann das Beispiel nicht abgenommen werden. Diese Studierenden haben in der Folge nicht mehr die Möglichkeit, die Übung der LVA im laufenden Semester erfolgreich abzuschließen. Damit haben Sie auch keine Möglichkeit, zur LVA-Prüfung anzutreten und diese im SS 2016 abzuschließen!

Abgabegespräch

Diese Gespräche finden am ***26.04.2016*** *oder* ***27.04.2016*** statt. Den Tag und die genaue Uhrzeit wählen Sie selbst beim Upload Ihres Beispiels in TUWEL. Der Link zu TUWEL ist

https://tuwel.tuwien.ac.at/course/view.php?id=7567.

Allgemeines

Im Laufe des Übungsteiles der LVA 384.061 „Objektorientiertes Programmieren“ muss jeder Teilnehmer drei Beispiele ausarbeiten und abgeben. **JEDES dieser drei Beispiel muss positiv bewertet werden, um die Übung zu bestehen** und damit die Möglichkeit zu erhalten, zur Abschlussprüfung anzutreten. Genauere Informationen zum Bewertungsschema finden Sie in den „Folien zur Übungseinweisung“ im TUWEL-Kurs, weitere Informationen zu Vorlesung und Übung finden Sie im Web unter:

https://tiss.tuwien.ac.at/course/educationDetails.xhtml?courseNr=384061&semester=2016S&windowId=df5
bzw.
https://tuwel.tuwien.ac.at/course/view.php?id=7567

Jeder Teilnehmer muss alle Beispiele **eigenständig** ausarbeiten. Bei der Abgabe jedes Beispiels gibt es ein kurzes Abgabegespräch mit einem Tutor oder Assistenten. Diese Abgabegespräche bestehen aus

- einem Test des Programms und Fragen dazu,
- der Durchführung kleiner Änderungen am Programm und
- theoretischen Fragen zu den jeweils in der Übung behandelten Konzepten.

Um das Beispiel abzugeben müssen Sie ihre Quellcode-Dateien in einem .zip-Archiv zusammenfassen und dieses in TUWEL hochladen. Bitte verwenden Sie **ausschließlich .zip-Dateien** (z.B., bsp1.**zip**) und **keine** anderen Archivformate!

Verwenden Sie keine *packages* um ihren Code zu strukturieren. Ihr hochgeladener Code des Beispiels **muss auf der Kommandozeile kompilieren** (d.h., ohne IDE). **Kommentieren** Sie Ihre Klassen und Methoden. Vergessen Sie insbesondere nicht, ihren Namen und ihre Matrikelnummer am Beginn jeder Klasse anzuführen. Halten Sie sich an die Java Code Conventions, siehe http://www.oracle.com/technetwork/java/codeconvtoc-136057.html.

Die Software für die LVA ist die Java Standard Edition (JDK Version 1.8). Diese finden sie unter
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.

Inhaltliche Fragen zu den Beispielen können Sie in das OOP-Forum von TUWEL stellen. Dadurch können sowohl die Tutoren als auch Studierende Ihre Fragen beantworten.

Organisatorische Fragen zur Lehrveranstaltung können Sie direkt an die Assistenten unter oop@ict.tuwien.ac.at richten.

Als Unterstützung für die Programmierung der Beispiele empfehlen wir das Java Tutorial:

http://docs.oracle.com/javase/tutorial/

Aufgabe: Datenverwaltung für „Stardust Airlines“

Ziel

Verstehen der folgenden Konzepte: Klassen, Objekte, Instanziierung, Vererbung, abstrakte Klassen, Enumerationen, Casting und Generics.

Einleitung

Das Beispiel besteht aus einer ***Minimalanforderung***, mit der Sie eine „~“ bei der Abgabe erreichen können. Für die Extrapunkte durch ein „+“ müssen Sie auch die ***Bonusanforderung*** lösen. Beachten Sie aber, wenn Sie die Bonusaufgabe abgeben, werden Sie auch in Theorie und Praxis intensiver dazu befragt!

Minimalanforderung

Zur Erfüllung der Minimalanforderungen lösen Sie die ***Aufgabenstellung − Minimalanforderung***.

Bonusanforderung

In einem optionalen Zusatzteil wird das Konzept `Generics` behandelt. Dieser Teil ist nicht verpflichtend, wird aber benötigt, um mögliche Zusatzpunkte auf den praktischen Teil des Beispiels zu erhalten. Lösen Sie dafür die ***Aufgabenstellung − Minimalanforderung*** und danach die ***Aufgabenstellung – Bonusanforderung*.**

Hinweis: Legen Sie eine Kopie des Eclipse Projekts an und implementieren Sie die Erweiterung in diesem. Bei der Abgabe sind beide Projekte abzugeben.

Aufgabenstellung − Minimalanforderung

Implementieren Sie eine Klassenhierarchie, die aus (abstrakten und nicht abstrakten) Klassen besteht, welche das Datenmanagement von „Stardust Airlines“ unterstützt. Implementieren Sie dafür die Klassen `Person` und `Employee` (beide abstrakt), und die nicht abstrakten Klassen `Pilot`, `Steward` und `Passenger`. Kreieren Sie außerdem die Klassen `Flight`, `Location` und `Controller` (alle nicht abstrakt) und die Enumerationen `ID_Token`, `PassengerStatus`, `PilotStatus` und `AirplaneType`. Speichern Sie jede Klasse oder Enumeration in einer separaten Java-Datei! Definieren Sie außerdem einen Konstruktor für jede Klasse, der die Initialisierung der Instanzvariablen dieser Klasse vornimmt (nützen Sie dafür die Vererbung in der definierten Hierarchie).

Jede `Person` hat die Attribute `name` und `birthday` (jeweils vom Typ `String`). Die Variablen aller Klassen müssen gekapselt und dürfen nur durch Methoden zugänglich sein. Zugriff auf solche Daten soll durch `get`*`Attributename`* und `set`*`Attributename`* Methoden (z.B. `getName()` und `setName(String name)` in der Klasse `Person` möglich sein. Implementieren Sie diese Methoden zumindest für alle Variablen, die Sie brauchen um ihre spezifische Aufgabenstellung zu bewältigen!

`Passenger` erweitert die Klasse `Person` und erbt dadurch deren Attribute. Diese Klasse fügt die beiden Attribute `id` (Typ `ID_Token`) und `status` (Typ `PassengerStatus`) hinzu. Die Enumeration `ID_Token` definiert die folgenden drei Werte: `PASSPORT`, `IDENTITY_CARD` und `DRIVING_LICENCE`. Die Enumeration `PassengerStatus` definiert die drei Werte: `GOLD_MEMBER`, `PLATIN_MEMBER` und `DIAMOND_MEMBER`.

`Employee` erweitert ebenfalls die Klasse `Person` und definiert zusätzlich die Attribute `salary` (Typ `Float`), `id` (Typ `String`) und `contractstart` (Typ `String`). `Employee` wird durch die Klassen `Steward` (keine weiteren Attribute) und `Pilot` erweitert. `Pilot` hat die zusätzlichen Attribute `planetypes` (Typ `List<Airplanetype>`) und `status` (Typ `PilotStatus`) definiert. Die Enumeration `Airplanetypes` definiert die folgenden sieben Werte: `AIRBUS_321`, `AIRBUS_340`, `AIRBUS_380`, `BOEING_737`, `BOEING_747`, `FOKKER` und `CESSNA`. Die Enumeration `PilotStatus` definiert die drei Werte: `CAPTAIN`, `CO-PILOT` und `TRAINEE`.

`Pilot` definiert außerdem die Methode `addPlanetype(Airplanetype type)`, welche einen spezifischen `Airplanetype` zu der Liste `planetypes` hinzufügt. Implementieren Sie die Methode `getPlanetypes()` um auf diese Liste zuzugreifen. Ein `Vector` ist die Java-Implementierung einer linearen Liste (siehe http://java.sun.com/javase/7/docs/api/).

Die neue Datenmanagement-Software von Stardust Airlines soll neben der allgemeinen Personenverwaltung auch gezielt die Verwaltung von Flügen, den Personen an Bord sowie von Start- und Landeflughäfen unterstützen.

Implementieren Sie zu diesem Zweck die Klasse `Flight` und `Location`. Jeder `Flight` ist durch eine `flightnumber` (Typ `String`), ein `airplane` (Typ `Airplanetype`), seine `crew` (Piloten und Stewards) und seine `passengers` (beide vom Typ `Vector`) sowie seine `origin` und `destination` (beide vom Typ `Location`) definiert. Jede `Location` besitzt die Attribute `country`, `airportcode` und `city` (alle vom Typ `String`).

`Flight` definiert außerdem die Methode `addEmployee(Employee employee_to_be_added)`, die einen `Employee` zum Vector `crew` hinzufügt und eine Methode `addPassenger(Passenger passenger_to_be_added)`, die einen `Passenger` zum Vector `passengers` hinzufügt. Implementieren Sie auch die korrespondierenden *get* Methoden um auf diese als Vector dargestellte Listen zugreifen zu können.

Implementieren Sie eine `Controller` Klasse, um die Objekte und Referenzen für Ihr Beispiel zu erstellen und zu verwalten. Diese Klasse verwaltet die Objektreferenzen und Instanzen Ihrer Klassenhierarchie durch `Vectoren`. Die `Controller` Klasse enthält außerdem die Methode `generateTestdata()`. Diese Methode erstellt eine vernünftige Anzahl von Testobjekten für Ihre Applikation. Des Weiteren enthält diese Klasse Ihre `main()`Methode und, falls von Ihnen benötigt, weitere Hilfsmethoden. Schlussendlich implementieren Sie in der `Controller` Klasse auch ihre `myTask(`*`Parameter …`*`)` Methode. Diese Methode enthält die Implementierung ihrer spezifischen Aufgabe je nach Matrikelnummer.

Folgende Aufgaben sollen – je nach Quersumme Ihrer Matrikelnummer – implementiert werden:
(Quersumme: 0812345 → 0 + 8 + 1 + 2+ 3 + 4 + 5 = 23 → 2 + 3 = 5)

1. Ermitteln Sie die Anzahl der Flüge von und nach Los Angeles. Geben Sie die Anzahl dieser Flüge und all deren Attribute auf der Konsole aus.
2. Ermitteln Sie die Anzahl der Piloten, die einen Airbus 380 fliegen dürfen. Geben Sie die Anzahl dieser Piloten und all deren Attribute auf der Konsole aus.
3. Ermitteln Sie, wie viele Personen sich auf einem bestimmten Flug befinden. Wie viele dieser Personen sind Piloten, Stewards und Passagiere? Geben Sie die Anzahl der Piloten, Stewards und Passagiere und all deren Attribute auf der Konsole aus.
4. Gibt es zumindest einen Piloten pro Flug, der das Flugzeug des Fluges fliegen darf? Geben Sie die Anzahl an Flügen und all deren Attribute, bei denen dies *nicht* der Fall ist, auf der Konsole aus.
5. Ermitteln Sie, wie viele Passagiere sich auf einem bestimmten Flug durch Reisepässe (passports), Personalausweise (identity cards) und wie viele sich durch Führerscheine (driving licenses) ausweisen. Geben Sie die Anzahl von Passagieren und all deren Attribute für jede Kategorie auf der Konsole aus.
6. Ermitteln Sie, wie viele Stewards auf jedem Flug sind. Geben Sie die Anzahl an Stewards und all deren Attribute für den jeweiligen Flug (auch hier alle Attribute) auf der Konsole aus.
7. Gibt es auf jedem Flug einen Co-Pilot? Geben Sie die Anzahl der Flüge und all deren Attribute, bei denen dies *nicht* der Fall ist, auf der Konsole aus.
8. Ermitteln Sie, wie viele Gold, Platin und Diamond Members sich auf einem bestimmten Flug befinden. Geben Sie deren Anzahl und alle Attribute für die jeweilige Kategorie auf der Konsole aus.
9. Ermitteln Sie, wie viele Personen in der Datenverwaltungssoftware von Stardust Airlines erfasst sind. Wie viele sind Passagiere, Stewards oder Piloten? Geben Sie jeweils die Anzahl und alle Attribute für diese drei Kategorien auf der Konsole aus.

Aufgabenstellung − Bonusanforderung

Diese Aufgabe ist optional, aber notwendig, um mögliche Zusatzpunkte auf den praktischen Teil der Abgabe zu erhalten. Verwenden Sie dazu einen Ansatz, der *Generics* (https://en.wikipedia.org/wiki/Generics_in_Java) und *Interfaces* von Java sinnvoll anwendet.

Die Stardust Airline wird in Zukunft nicht nur Passagierflüge anbieten, sondern möchte auch in den Frachttransportsektor expandieren. Weiters soll es auch möglich sein, bei Flügen unterschiedliche Mitarbeiterklassen für die Flugabwicklung festzulegen.

Erweitern Sie entsprechend die ***Aufgabenstellung – Minimalanforderung***:

1. Erweitern Sie die Implementierung der Basisaufgabe so, dass sowohl Passagier- als auch Transportflüge angelegt und zu verwaltet werden können.
2. Bei allen Flügen soll die Möglichkeit bestehen, die Crew auf einen Mitarbeitertyp einzugrenzen (sinnvoller Einsatz von Generics).

Erweitern Sie entsprechend Ihre `generateTestdata`() Methode mit sinnvollen Testdatensätzen.