Übung 05: Collections

Abgabetermin: 6. 4. 2017, 8:15

Name: _____ **Matrikelnummer**: _____

Informatik: ☐ G1 (Marr) ☐ G2 (Prähofer) ☐ G3 (Prähofer) ☐ G4 (Löberbauer)

WIN: ☐ G1 (Khalil) ☐ G2 (Hummel) ☐ G3 (Khalil)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 5	24	Java-Programm, Ausgabe des Testlaufs	Java-Programm	☐	

Übung 05: Straßenbahnfahrplan

In dieser Übung ist ein Straßenbahnfahrplan basierend auf generischen Collection-Klassen zu implementieren. Ziel ist es, einfache Abfragen zum Fahrplan durchführen zu können.

Ein Straßenfahrplan ist definiert durch eine Reihe von Linien (Klasse `Line`), die eine Reihe von Haltestellen (Klasse `Station`) anfährt. Zwischen den Haltestellen besteht eine bestimmte Fahrzeit. Eine Straßenbahn (Klasse `Tram`) fährt eine Linie, hat eine Abfahrtszeit und hält an den Haltestellen der Linie (Klasse `Stop`). Damit ist ein Stop definiert durch die Haltestelle, die Uhrzeit, und die Linie. Die Zeiten (verwenden Sie Klasse `java.time.LocalTime`) für die Stops einer Tram sind durch die Abfahrtszeit und die in der Linie definierten Fahrzeiten bestimmt. Ein Straßenbahnfahrplan (Klasse `TramSchedule`) wird durch eine Menge von Haltestellen (`Station`), Menge von Linien (`Line`) und eine Reihe von Straßenbahnen (`Tram`) bestimmt.

Anmerkung:

Die grundlegenden Klassenstrukturen sind im Download `UE05_Download.zip` bereitgestellt und müssen nur ausimplementiert werden. Insbesondere enthält der Download auch die Klasse `Linz` mit wesentlichen Datendefinitionen für Linien und Abfahrtszeiten, aus denen der Fahrplan einfach aufgebaut werden kann.

a) Aufbau des Fahrplans

Folgend sind die zu implementierenden Klassen genauer beschrieben.

Klasse `Station`

Stationen soll durch eine Enumerationsklasse `Station` implementiert werden, die alle gültigen Stationen des Fahrplans als Werte definiert. Folgende Stationen soll es geben (Auswahl der Haltestellen in Linz): *Universitaet*, *Dornach*, *Gruendberg*, *Ontlstrasse*, *Rudolfstrasse*, *Mozartkreuzung*, *Hauptbahnhof*, *Bulgariplatz*, *Scharlinz*, *Kleinmuenchen*, *Simonystrasse*, *Dauphinestrasse*, *Auwiesen*, *Ebelsberg*, *Ennsfeld*, *SolarCity*.

Zusätzlich speichert eine Station eine sortierte Menge der Trams, die an dieser Station halten. Dabei soll die Menge der Trams nach der Abfahrtzeit sortiert sein.

Klasse `Line`

Eine Line ist definiert durch eine Nummer, eine Zielstation (z.B. Linie *2*, *Universtaet*), eine Reihe von Haltestellen und die Fahrzeiten zu diesen Haltestellen. Die Haltestellen mit den zugehörigen Fahrzeiten sollen in einer sortieren Liste gespeichert werden.

Hinweis: Verwenden Sie eine Hilfsklasse für Haltestelle und Fahrzeit.

Klasse `Tram`

Trams fahren eine bestimmte Linie und starten an der ersten Haltestelle der Linie zu einer bestimmten Uhrzeit. Mit den Fahrzeiten sind damit auch Ankunftszeiten an den entsprechenden Stationen definiert. Die Ankünfte der Trams sollen durch eine sortierte Menge von Stops (Klasse `Stop`, siehe folgend) definiert sein.

Klasse `Stop`

Ein Stop ist durch die Tram, durch die Station und durch die Zeit bestimmt.

Hinweis: Die Stops sollen sowohl bei den Trams als auch bei den Stationen gespeichert werden.

Klasse `TramSchedule`

Der Fahrplan soll zum einfachen Zugriff durch eine Menge von Linien und Trams definiert.

b) Abfragenmethoden

Implementieren Sie nun bei den einzelnen Klassen folgende Abfragen:

bei Klasse `Tram`

- Zugriff auf alle Stopps der Tram
- Zugriff auf alle Stopps ab einer bestimmten Haltestelle

bei Klasse `Station`

- Zugriff auf alle Trams
- Zugriff auf alle Trams einer Linie
- Zugriff auf alle Trams einer Linie ab einer bestimmten Zeit

c) Test

Testen Sie das System, indem Sie einen einfachen Fahrplan für Linz erstellen. Der Fahrplan braucht nur Linie 1 und 2 für die Verbindung von Universität nach Auwiesen (Linie 1), sowie Universität nach SolarCity (Linie 2) enthalten (siehe Anhang).

Anschließend testen sie das System mit mehreren Abfragen von

- Trams an einer bestimmten Station, einer bestimmten Linie nach einer gegebenen Zeit
- Abfragen von Stopps der gefundenen Trams ab der Station

und geben Sie die Ergebnisse entsprechend aus.

Hinweise:

Verwenden Sie die Interfaces und Klassen des Collection-Packages (`java.util`), z.B. `List`, `Map`, `SortedMap`, `Set` und `SortedSet` und Klassen `ArrayList`, `HashSet`, `TreeMap` und `TreeSet`. Überlegen Sie genau den Einsatz der unterschiedlichen Collections.

Überlegen Sie genau die Realisierung der Sortierreihenfolgen für die unterschiedlichen Objektsammlungen. Dies kann entweder durch die Implementierung des `Comparable`-Interfaces durch die Klassen oder durch die Definition eigener `Comparator`-Klassen erfolgen.

Wenn Sie eine Collection an Ihre Testklasse übergeben, achten Sie darauf, dass diese nicht verändert werden soll.

Für die Zeiten können Sie die Klasse `java.time.LocalTime` verwenden, die bereits `Comparable<LocalTime>` implementiert und eine Reihe nützlicher Methoden (z.B. für die Addition mit Minuten) zur Verfügung stellt.

Anhang: Fahrpläne (eingeschränkt auf ausgewählte Stationen mit Fahrzeiten)

Die Klasse `Linz` stellt diese Daten zur Nutzung bereit.

Linie 1 Auwiesen

Universitaet	0
Dornach	2
Gruendberg	6
Ontlstraße	9
Rudolfstraße	14
Mozartkreuzung	19
Hauptbahnhof	24
Bulgariplatz	28
Scharlinz	33
Kleinmuenchen	36
Simonystraße	37
Dauphinestraße	39
Auwiesen	41

Linie 1 Universitaet

Universitaet	41
Dornach	39
Gruendberg	35
Ontlstraße	32
Rudolfstraße	27
Mozartkreuzung	22
Hauptbahnhof	17
Bulgariplatz	13
Scharlinz	8
Kleinmuenchen	5
Simonystraße	4
Dauphinestraße	2
Auwiesen	0

Linie 2 SolarCity

Universitaet	0
Dornach	2
Gruendberg	6
Ontlstraße	9
Rudolfstraße	14
Mozartkreuzung	19
Hauptbahnhof	24
Bulgariplatz	28
Scharlinz	33
Kleinmuenchen	36
Ebelsberg	40
Ennsfeld	45
SolarCity	51

Linie 2 Universtaet

Universitaet	51
Dornach	49
Gruendberg	45
Ontlstraße	42
Rudolfstraße	37
Mozartkreuzung	32
Hauptbahnhof	27
Bulgariplatz	23
Scharlinz	18
Kleinmuenchen	15
Ebelsberg	11
Ennsfeld	6
SolarCity	0