

## Assignment 04: Interfaces, Generics, Collections

Hand in by 2. 4. 2009, 8:15

Name: \_\_\_\_\_

Matrikelnummer: \_\_\_\_\_

Informatik:  G1 (Prähofer)     G2 (Prähofer)     G3 (Würthinger)     G4 (Prähofer)

WIN:             G1 (Khalil)             G2 (Khalil)             G3 (Schwinger)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Assignment 4	24	Text description, Java program, Output of test run	Java program	<input type="checkbox"/>	

### Assignment 04: Train Schedule

In this assignment you should use classes of the generic Collection API to implement a class system for a train schedule. The train schedule (class `Timetable`) has a set of stations (class `Station`), trains (class `Train`) and a sequence of stops<sup>1</sup> (class `Stop`), which belong to a train and define a stop of the train at a station at a particular platform and at a given time.

*Trains (class Train)*

Trains have a unique name and store a list of stops which are sorted based on time. Class `Train` has the following methods:

- Add a stop at a station, at a platform of this station and at a given time (the stop should also be added at the station)
- Get the first and the last stop of this train and access to source and destination stations
- Get the list of stops in ascending order
- Get the set of stops after a given station (name of station should be the parameter)

*Stations (class Station)*

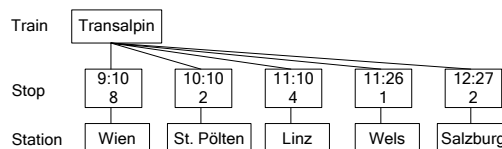
Stations have a unique name and store a list of stops sorted based on time and by equal time based on platform. The class `Station` provides the following methods:

- Get all stops at this station
- Get all trains which stop at this station

*Stops (class Stop)*

Stops represent the connection between stations and trains. They store the train they belong to, the station, the platform and the time (i.e., one can access `Stop` objects from `Train` und `Station` objects).

The following figure shows the interrelations between objects `Train`, `Station`, and `Stop` by an example.



*Train schedule (class Timetable)*

A train schedule has a set of stations. The set of stations is sorted based on station name.

A train schedule also stores a set of trains. The set of trains is sorted based on departure time from the source station and, when equal, based on station name.

<sup>1</sup> To keep the system simple we do not distinguish between arrival and departure but only use stops at a particular time.

Test your system by creating a simple train schedule with several trains and stations and allow printing the following information:

- All stations in alphabetical order together with all train departures (time of departure and final destination of the train)

```
Bahnhof: Linz
  10:12 Graz
  10:46 Salzburg
  ...
Bahnhof: Wels
  11:03 Salzburg
  ...
```

- All trains stopping at one particular station (defined by its name), which stop at this station together with all the following stops of the train.

```
Train connections from Linz
Steirerland:
  10:12 Linz (2)
  10:45 Kirchdorf (2)
  11:18 Rottenmann (2)
  12:09 Graz (1)
Donauwalzer:
  10:46 Linz (3)
  11:03 Wels (1)
  12:33 Salzburg (3)
```

- All trains with all stops

```
Donauwalzer
  09:10 Wien (1)
  10:46 Linz (3)
  11:03 Wels (1)
  12:33 Salzburg (3)

Steirerland
  10:12 Linz (2)
  10:45 Kirchdorf (2)
  11:18 Rottenmann (2)
  12:09 Graz (1)
```

### Hints:

Use interfaces `List`, `Set` and `SortedSet` and classes `ArrayList`, `HashSet` and `TreeSet` from the collection package (`java.util`).

Implement the different sorting criterias by implementing interfaces `Comparable` and `Comparator`.

For time you should use class `datetime.Time`. `Time` already has method `compareTo` which you should use.