

Musterbeispiel (Übung für den 3. Praxistest)

Aufgabe:

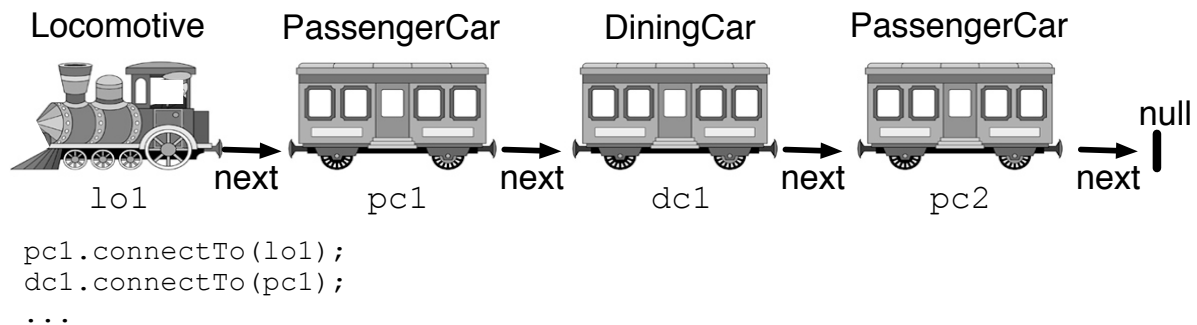
Schreiben Sie die Dateien

- Car.java
- PassengerCar.java
- Train.java
- Locomotive.java
- DiningCar.java

Die Basisklasse *Car* (30%):

Die Klasse *Car* repräsentiert ein Schienenfahrzeug (z.B. Lokomotive, Wagon) mit einem Gewicht in Tonnen (*double*) und Länge in Metern (*double*). Jedes Schienenfahrzeug besitzt weiters eine eindeutige Seriennummer, die vom Konstruktor festgelegt wird und nicht mehr veränderbar ist. Die Seriennummern werden fortlaufend vergeben. Jenes Schienenfahrzeug, das als erstes erzeugt wird, bekommt die Seriennummer 1. Jedes *Car* soll eine *toString*-Methode besitzen, die eine lesbare Repräsentation in der Form *Seriennummer, Gewicht, Länge* liefert.

Weiters verfügt *Car* über eine Methode *connectTo (Car)*, die das Schienenfahrzeug an ein anderes Schienenfahrzeug, das als Parameter angegeben wird, hinten anhängt, sodass ein Zug entsteht. Jedes *Car*-Objekt verfügt zu diesem Zweck über eine *Car*-Referenz *next*, die auf den angehängten Wagon verweist. Die Methode *connectTo* muss also dafür sorgen, dass *next* des Vorgängerswagons entsprechend gesetzt wird.



Abgeleitete Klassen (30%):

Die Klasse *Locomotive* repräsentiert ein spezielles Schienenfahrzeug mit einem Motor. Eine Lokomotive hat also zusätzlich zu den Eigenschaften von *Car* eine Leistung in kW (*int*). Schreiben Sie einen entsprechenden Konstruktor. Die Klasse soll eine *toString*-Methode besitzen, die eine lesbare Repräsentation in der Form *Seriennummer, Gewicht, Länge, Leistung* liefert.

Die Klasse *PassengerCar* repräsentiert ein spezielles Schienenfahrzeug: einen Personenwagen (ohne Motor) der zusätzlich zu den Eigenschaften von *Car* eine bestimmte Anzahl an Sitzplätzen (*int*) hat. Schreiben Sie einen entsprechenden Konstruktor. Die Klasse soll eine *toString*-Methode besitzen, die eine lesbare Repräsentation in der Form *Seriennummer, Gewicht, Länge, Plätze* liefert.

Die Klasse *DiningCar* repräsentiert einen speziellen Personenwagen, nämlich einen Speisewagen wobei hier zusätzlich zu den Eigenschaften von *PassengerCar* angegeben werden muss, ob der Speisewagen geöffnet ist (*boolean*). Schreiben Sie einen entsprechenden Konstruktor. Die Klasse soll eine *toString*-Methode besitzen, die eine lesbare Repräsentation in der Form *Seriennummer, Gewicht, Länge, Plätze, geöffnet?* liefert.

Die Datenelemente der oben beschriebenen Klassen sollen nach der Erzeugung von fremden (nicht verwandten) Klassen nicht mehr veränderbar sein. Stellen Sie (logische) Unveränderbarkeit durch Wahl geeigneter Modifier sicher.

Die Klasse *Train*(40%):

Die Klasse *Train* repräsentiert einen Zug, der nach der Erzeugung zunächst nur aus einer Lokomotive (Datenelement) besteht. Die Lokomotive wird dem Konstruktor übergeben. Die Methode *add(Car)* hängt das übergebene Schienenfahrzeug an das letzte Schienenfahrzeug des Zuges an (Methode *connectTo* der Klasse *Car*). Beim ersten Aufruf von *add* wird das übergebene Schienenfahrzeug also an die Lokomotive gehängt. An dieses Schienenfahrzeug wird dann beim nächsten Aufruf von *add* das nächste Schienenfahrzeug gehängt, u.s.w.. (Verwenden Sie hier *keine* Collectionklassen oder Arrays).

Die Methode *hasOpenDiningCar()* liefert einen *boolean*-Wert zurück, der angibt, ob in dem Zug an einer beliebigen Position ein Speisewagen (*DiningCar*) vorkommt, der geöffnet hat. In diesem Fall wird *true* geliefert, sonst *false*.

Die Methode *toString* liefert eine lesbare Repräsentation des Zuges, d.h., aller Wägen. Für den Fall, dass der Zug über einen geöffneten Speisewagen verfügt, soll zusätzlich "Has open dining car" angegeben werden.

Hinweis:

Vorbedingungen der gefragten Methoden müssen nicht überprüft werden, es sei denn es wird in der Angabe explizit verlangt. Die vorgefertigte Datei *TrainTest.java* enthält die Definition der ausführbaren Klasse, die Sie zum Testen Ihrer Klassen benutzen können. Beachten Sie dass die Testfälle in der *TrainTest* Klasse nicht alle möglichen Testfälle abdecken. Ihr Inhalt wird nicht beurteilt.