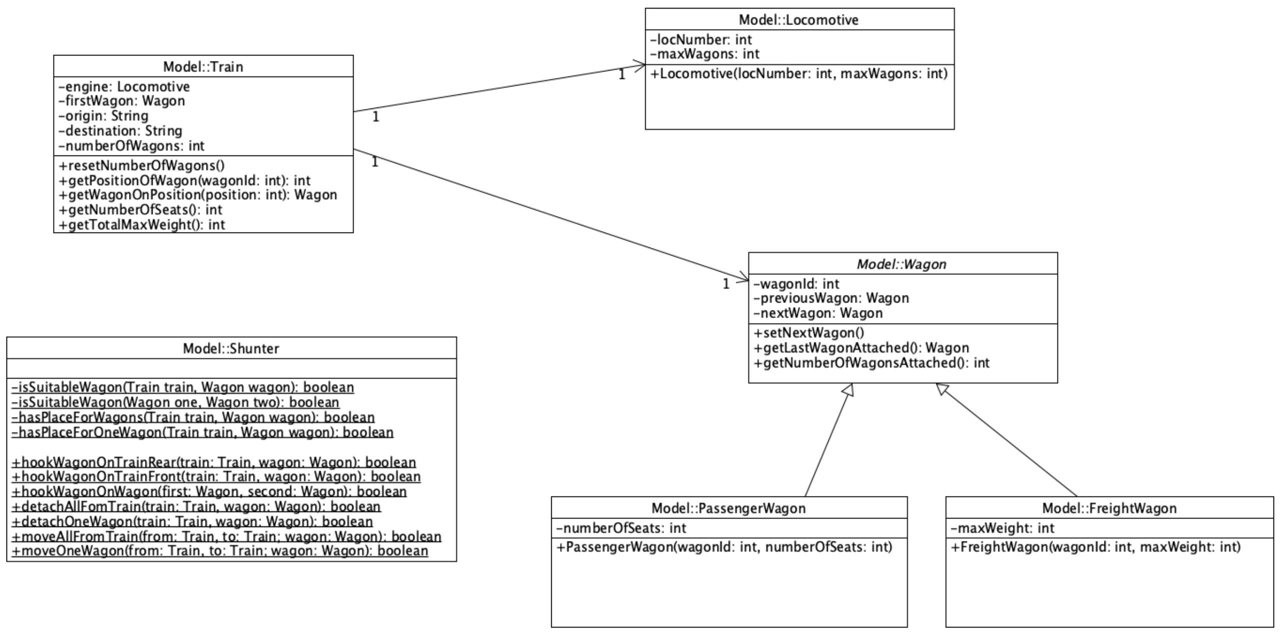
Algorithms and Data structures

Trains

In this assignment you will work on an application that deals with trains. A train always has a locomotive to pull (the engine) the train and wagons can be hooked to the locomotive or to another wagon. Of course wagons can also be detached and furthermore, they can be moved from one train to the other. This is all done by a shunter. A wagon can be hooked to a train by hooking is to the last wagon or by hooking it to the locomotive. But wagons can also be hooked to each other without being a part of a train. The shunter can hook a wagon with all the wagons behind it to a train, or detach a wagon with all wagons behind and even move a row of wagons from one train to another. When hooking or detaching wagons to or from a train the number of wagons of the train should always be adjusted (recalculated).

Trains can either exist of passenger wagons or freight wagons. As soon as a train has a first wagon of any kind this determines the kind of train it is. Passenger wagons may not be hooked to a freight train and vice versa. The engine of a train has a capacity, the maximum number of wagons it can pull. So, the number of wagons of a train can never exceed the capacity of the engine.

The number of seats of a passenger wagon are registered and the maximum weight a freight wagon is registered. For a train three attributes can be calculated, the number of wagons, the total number of seats of all the wagons for a passenger train and the total maximum weight of all the wagons for a freight train. For a passenger train the weight will be set to zero, and for a freight train the number of passengers will be set to zero.



Model::Train
-engine: Locomotive
-firstWagon: Wagon
-origin: String
-destination: String
-numberOfWagons: int
+resetNumberOfWagons()
+getPositionOfWagon(wagonId: int): int
+getWagonOnPosition(position: int): Wagon
+getNumberOfSeats(): int
+getTotalMaxWeight(): int
1
1
1
1
Model::Locomotive
-locNumber: int
-maxWagons: int
+Locomotive(locNumber: int, maxWagons: int)
Model::Wagon
-wagonId: int
-previousWagon: Wagon
-nextWagon: Wagon
+setNextWagon()
+getLastWagonAttached(): Wagon
+getNumberOfWagonsAttached(): int
Model::Shunter
-isSuitableWagon(Train train, Wagon wagon): boolean
-isSuitableWagon(Wagon one, Wagon two): boolean
-hasPlaceForWagons(Train train, Wagon wagon): boolean
-hasPlaceForOneWagon(Train train, Wagon wagon): boolean
+hookWagonOnTrainRear(train: Train, wagon: Wagon): boolean
+hookWagonOnTrainFront(train: Train, wagon: Wagon): boolean
+hookWagonOnWagon(first: Wagon, second: Wagon): boolean
+detachAllFomTrain(train: Train, wagon: Wagon): boolean
+detachOneWagon(train: Train, wagon: Wagon): boolean
+moveAllFromTrain(from: Train, to: Train; wagon: Wagon): boolean
+moveOneWagon(from: Train, to: Train; wagon: Wagon): boolean
Model::PassengerWagon
-numberOfSeats: int
+PassengerWagon(wagonId: int, numberOfSeats: int)
Model::FreightWagon
-maxWeight: int
+FreightWagon(wagonId: int, maxWeight: int)

Part 1.

For this assignment there is a start project available. Four classes are already implemented in the model package, but most of the code of the methods mentioned in the class diagram is missing. The classes PassengerWagon and FreightWagon do not exist yet and should be implemented according to the diagram. In the controller package you will find a launcher class containing the main method.

The project also contains unit tests to test your implementations. The unit tests are by far not complete. Make sure to test all possible cases.

Getters and setters are not shown in the class diagram. Only implement getters and setters when necessary.

a) Start by implementing the two subclasses PassengerWagon and FreightWagon including the methods in the diagram. Make sure the Wagon class cannot be instantiated.
b) Implement the three methods of the Wagon class. See comments in the code for details.
c) Implement the five methods of the Train class. See comments for details.
d) The Shunter class is a helper class that contains static methods. All of the methods return a boolean value to indicate whether the action in the method succeeded. There are four private helper methods that should be used in the other methods. Wagons cannot be hooked to a train when the number of wagons will exceed the maximum number the engine can pull. Wagons cannot be detached from a train whenever the wagon is not on the train. Passenger wagons cannot be hooked to a freight train and vice versa. Implement all the methods in the Shunter class using this information and the comments in the class.
e) Use the available test to check your code.
f) Add sufficient tests to check all methods you have implemented. Make sure to test all possible outcomes of the methods.

Part 2.

A different approach would be to make the Train class implement the Iterable interface and use for-each loops instead of while loops.

a) Change the code of the Train class so that it implements the Iterable interface.
b) Choose two methods that use a while loop and change it to a for each loop using the Iterator of the Train class.