# 03. Vintage locos



*Vintage locomotives are fascinating pieces of engineering history, embodying the evolution of rail transportation over the years. These locomotives, often characterized by their distinctive designs and steam-powered engines, played a crucial role in the development of modern railroads.*

## Preparation

Download the skeleton provided in Judge. **Do not** change the **packages**!

**Pay attention to name the package vintageLocos, all the classes, their fields, and methods the same way they are presented in the following document. It is also important to keep the project structure as described.**

## Problem description

Your task is to create a repository that stores locomotives by creating the classes described below.

### Locomotive

First, write a class **Locomotive** with the following properties:

* **name: String**
* **builder: String**
* **buildDate: LocalDate**
* **gauge: int**
* **weight: int**
* **maxSpeed: int**

The class **constructor** should receive **name, builder, buildDate, gauge, weight** and **maxSpeed**. You need to create the appropriate **getters and setters**. All locomotive names will be **unique.** It is guaranteed that there **will be no duplicates** of names.

Override the **toString()** method in the following format:

* **"This {locomotive name} is made by {locomotive builder} at {locomotive build date in the format "dd.MM.yyyy"}."**

**Hint:** You can use this to **format** the date:

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("**dd.MM.yyyy**");

String **formattedDate** = LocalDate.from(getBuildDate()).format(formatter);

### TrainStation

**Next**, write a class **TrainStation**. The **TrainStation** class should have those **properties**:

* **name: String**
* **capacity: int**
* **railGauge: int**
* **locomotives: List<Locomotive>**

The class **constructor** should receive **name, capacity** and **railGauge.** Also, it should initialise the **locomotives** with a new **collection** instance.Implement the following features:

* **Method addLocomotive(Locomotive locomotive)** – **adds** an **entity** to the **collection** of locomotives **if a space for it** and the rail gauge of the station **matches** the gauge of the locomotive, otherwise print:
* **"This train station is full!"**

If the **rail gauge** of the station **does not match** the gauge of the locomotive print:

* **"The rail gauge of this station does not match the locomotive gauge! Difference: {the difference between the station gauge and the locomotive gauge in absolute value} mm."**
* **Method removeLocomotive(String name)** – removes a **Locomotive** by **given name,** if such **exists**, and **returns boolean** (**true** if it is removed, otherwise – **false**)
* **Method getFastestLocomotive()**– **returns String** - the **fastest locomotive** by **the speed** in the given train station if there is one, in the following format:
* **"{locomotive name} is the fastest locomotive with a maximum speed of {locomotive maxSpeed} km/h."**

If there are no locomotives in the station print:

* **"There are no locomotives."**
* **Method getLocomotive(String name)** – **returns** the **locomotive** with the **given name,** otherwise – returns **null**
* **Method getCount()** – **returns** the **count** of **locomotives** in the given train station
* **Method getOldestLocomotive()** **–** **returns** **String** **–** the name of the oldest **Locomotive** in the given train station. Otherwise print:
* **"There are no locomotives."**
* **Method getStatistics()** – **returns** a **String** in the following **format** (print the locomotives in **order of addition**):
  + **"Locomotives departed from {tain station name}:  
    1. {locomotive name}  
    2. {locomotive name}  
    (…)**

1. **{locomotive name}"**

If there are **no** locomotives in some train stations print **only** this text:

* **"There are no locomotives departing from {rail station name} station."**

## Constraints

* The **name**, **build date** and **max speed** of the locomotive will always be **unique**.
* You will always have a locomotive added before receiving methods manipulating the TrainStation's locomotive.

## Examples

This is an example of how the **TrainStation** class is **intended to be used**.

|  |
| --- |
| **Sample code usage** |
| *//Initialize the repositories (Train station)* TrainStation deptford = new TrainStation("Deptford Train station", 5, 1435); TrainStation euston = new TrainStation("Euston", 2, 1524);  *//Initialize entities (Locomotive)* Locomotive puffingBilly = new Locomotive("Puffing Billy","Jonathan Forster", LocalDate.*of*(1814, 1, 1), 1524, 8250, 8);  Locomotive flyingScotsman = new Locomotive("Flying Scotsman","Doncaster Works", LocalDate.*of*(1923, 2, 14), 1435, 96250, 161);  Locomotive cityOfTruro = new Locomotive("City of Truro","GWR Swindon Works", LocalDate.*of*(1903, 4, 1), 1435, 56200, 160);  deptford.addLocomotive(flyingScotsman);  *//The rail gauge of this station does not match the locomotive gauge! Difference: 89 mm.* deptford.addLocomotive(puffingBilly); deptford.addLocomotive(cityOfTruro); euston.addLocomotive(puffingBilly);  System.***out***.println(euston.removeLocomotive("Puffing Billy")); *//true* System.***out***.println(euston.removeLocomotive("Silver Star")); *//false*  System.***out***.println(deptford.getFastestLocomotive());  *//Flying Scotsman is the fastest locomotive with a maximum speed of 161 km/h.*  System.***out***.println(euston.getFastestLocomotive()); *//There are no locomotives.* System.***out***.println(euston.getOldestLocomotive()); *//There are no locomotives.*  System.***out***.println(deptford.getLocomotive("Flying Scotsman"));  *//This Flying Scotsman is made by Doncaster Works at 14.02.1923.*  System.***out***.println(deptford.getOldestLocomotive()); *//City of Truro* System.***out***.println(deptford.getCount()); *//2* System.***out***.println(deptford.getStatistics());  *//Locomotives departed from Deptford Train station: //1. Flying Scotsman //2. City of Truro*  System.*out*.println(euston.getStatistics());  *//There are no locomotives departing from Euston station.* |

## Submission

Submit **single .zip file**, containing **vintageLocos** package, **with the classes inside** (**TrainStation**, **Locomotive** and the **Main** **class)**, there is no specific content required inside the **Main** class e.g. you can do any kind of local testing of your program there. However, there should be **main(String[] args)** method inside.