# More Exercise: Objects and Classes

Problems for exercises and homework for the ["Programming Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3837/programming-fundamentals-with-java-september-2022).

You can check your solutions in [Judge.](https://judge.softuni.org/Contests/1328)

## Company Roster

Define a class **Employee** with the following information: **name, salary, position, department, email,** and **age**. The **name, salary**, **position,** and **department** are **mandatory,** while the rest are **optional**.

Your task is to write a program that takes **N** lines of employees from the console and calculates the department with the highest average salary, and prints for each employee in that department his **name, salary, email, and age** – **sorted by salary in descending order**. If an employee **doesn't have** an **email** – in place of that field, you should print "**n/a**" instead, if he doesn't have an **age** – print "**-1**" instead. The **salary** should be printed to **two decimal places** after the separator.

**Hint**: you can define a **Department** class with a list of employees.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Peter 120.00 Dev Development peter@abv.bg 28  Todor 333.33 Manager Marketing 33  Itan 840.20 ProjectLeader Development itan@itan.com  George 0.20 Freeloader Nowhere 18 | Highest Average Salary: Development  Itan 840.20 itan@itan.com -1  Peter 120.00 peter@abv.bg 28 |
| 6  Stan 496.37 Temp Coding stan@yahoo.com  Yana 610.13 Manager Sales  Todor 609.99 Manager Sales todor@abv.bg 44  Venecia 0.02 Director BeerDrinking beer@beer.br 23  Andrey 700.00 Director Coding  Popeye 13.3333 Sailor SpinachGroup popeye@pop.ey | Highest Average Salary: Sales  Yana 610.13 n/a -1  Todor 609.99 todor@abv.bg 44 |

## Raw Data

You are the owner of a courier company and want to make a system for tracking your cars and their cargo. Define a class **Car** that holds information about the **model, engine, cargo,** and a **collection of exactly 4 tires**. The engine, cargo, and tire **should be separate classes**, and create a constructor that receives all information about the Car and creates and initializes its inner components (engine, cargo, and tires).

On the first line of input you will receive a number **N** - the number of cars you have, on each of the next **N** lines you will receive information about a car in the format "**{Model} {EngineSpeed} {EnginePower} {CargoWeight} {CargoType} {Tire1Pressure} {Tire1Age} {Tire2Pressure} {Tire2Age} {Tire3Pressure} {Tire3Age} {Tire4Pressure} {Tire4Age}**" where the speed, power, weight and tire age are **integers**, tire pressure is a **double.**

After the **N** lines, you will receive a single line with one of 2 commands "**fragile**" or "**flamable**", if the command is "**fragile**" print all cars whose **Cargo Type is** "**fragile**" with a **tire** whose **pressure is** **< 1** if the command is "**flamable**" print all cars whose **Cargo Type is** "**flamable**" and have **Engine Power > 250**. The cars should be printed in order to appear in the input.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  ChevroletAstro 200 180 1000 fragile 1.3 1 1.5 2 1.4 2 1.7 4  Citroen2CV 190 165 1200 fragile 0.9 3 0.85 2 0.95 2 1.1 1  fragile | Citroen2CV |
| 4  ChevroletExpress 215 255 1200 flamable 2.5 1 2.4 2 2.7 1 2.8 1  ChevroletAstro 210 230 1000 flamable 2 1 1.9 2 1.7 3 2.1 1  DaciaDokker 230 275 1400 flamable 2.2 1 2.3 1 2.4 1 2 1  Citroen2CV 190 165 1200 fragile 0.8 3 0.85 2 0.7 5 0.95 2  flamable | ChevroletExpress  DaciaDokker |

## Car Salesman

Define two classes **Car** and **Engine.** A **Car** has a **model, engine, weight,** and **color**. An Engine has a **model**, **power, displacement,** and **efficiency**. A Car's **weight** and **color** and its Engine's **displacements** and **efficiency** are **optional**.

On the first line, you will read a number **N** which will specify how many lines of engines you will receive, on each of the next **N** lines, you will receive information about an **Engine** in the following format "**{Model} {Power} {Displacement} {Efficiency}**". After the lines with engines, on the next line, you will receive a number **M** – specifying the number of Cars that will follow, on each of the next **M** lines, information about a **Car** will follow in the following format "**{Model} {Engine} {Weight} {Color}**", where the engine in the format will be the **model of an existing** **Engine**. When creating the object for a **Car**, you should keep a **reference to the real engine** in it, instead of just the engine's model, note that the optional properties **might be missing** from the formats.

Your task is to print each car (in the order you received them) and its information in the format defined below, if any of the optional fields have not been given, print "**n/a**" in its place instead:

"**{CarModel}:** **{EngineModel}:** **Power: {EnginePower}** **Displacement: {EngineDisplacement}** **Efficiency: {EngineEfficiency}** **Weight: {CarWeight}** **Color: {CarColor}**"

### Optional

Override the classes' **ToString()** methods to have a reusable way of displaying the objects.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  V8-101 220 50  V4-33 140 28 B  3  FordFocus V4-33 1300 Silver  FordMustang V8-101  VolkswagenGolf V4-33 Orange | FordFocus:  V4-33:  Power: 140  Displacement: 28  Efficiency: B  Weight: 1300  Color: Silver  FordMustang:  V8-101:  Power: 220  Displacement: 50  Efficiency: n/a  Weight: n/a  Color: n/a  VolkswagenGolf:  V4-33:  Power: 140  Displacement: 28  Efficiency: B  Weight: n/a  Color: Orange |
| 4  DSL-10 280 B  V7-55 200 35  DSL-13 305 55 A+  V7-54 190 30 D  4  FordMondeo DSL-13 Purple  VolkswagenPolo V7-54 1200 Yellow  VolkswagenPassat DSL-10 1375 Blue  FordFusion DSL-13 | FordMondeo:  DSL-13:  Power: 305  Displacement: 55  Efficiency: A+  Weight: n/a  Color: Purple  VolkswagenPolo:  V7-54:  Power: 190  Displacement: 30  Efficiency: D  Weight: 1200  Color: Yellow  VolkswagenPassat:  DSL-10:  Power: 280  Displacement: n/a  Efficiency: B  Weight: 1375  Color: Blue  FordFusion:  DSL-13:  Power: 305  Displacement: 55  Efficiency: A+  Weight: n/a  Color: n/a |

## Teamwork Projects

It's time for teamwork projects, and you are responsible for making the teams. First, you will receive an integer - the **count** of the teams you will have to **register**. You will be given a **user** and a **team** (separated with "-"). The user is the **creator** of that team.

For every newly created team, you should **print** a message:

"Team {team Name} has been created by {user}!"

Next, you will receive a user with the team (separated with "*->*") which means that the user wants to **join** that **team**. Upon receiving the command: "end of assignment", you should print **every team**, **ordered** by the **count** of its **members** (**descending**) and then by **name** (**ascending**). For each team (disband teams as well), you have to print its members **sorted** by name (**ascending**). However, there are several **rules**:

* If a user tries to **create** a team more than once, a message should be displayed:   
  "Team {teamName} was already created!"
* The creator of a team cannot **create** another team - the message should be thrown:   
  "{user} cannot create another team!"
* If a user tries to **join** a currently non-existing team, a message should be displayed:   
  "Team {teamName} does not exist!"
* A Member of a team cannot **join** another team - the message should be thrown:  
  "Member {user} cannot join team {team Name}!"
* At the **end** (*after teams' report*), teams with **zero** members (with **only a creator**) should **disband, ordered by name in ascending other**.
* Every **valid** team should be printed ordered by **name** (ascending) in this format:

"**{teamName}:  
 - {create}  
 -- {member}...**"

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2  Didy-PowerPuffsCoders  Tony-Tony is the best  Petya->PowerPuffsCoders  Tony->Tony is the best  end of assignment | Team PowerPuffsCoders has been created by Didy!  Team Tony is the best has been created by Tony!  Member Tony cannot join team Tony is the best!  PowerPuffsCoders  - Didy  -- Petya  Teams to disband:  Tony is the best | Tony created a team that he tried later to join. So the message was shown. Since no one is trying to join his team, the team has to **disband**. |
| 3  Tatyana-CloneClub  Helena-CloneClub  Trifon-BRBClub  Peter->bRBClub  Peter->BRBClub  Tatyana->Leda  Petter->BRBClub  Cossima->CloneClub  end of assignment | Team CloneClub has been created by Tatyana!  Team CloneClub was already created!  Team BRBClub has been created by Trifon!  Team bRBClub does not exist!  Team Leda does not exist!  BRBClub  - Trifon  -- Peter  -- Petter  CloneClub  - Tatyana  -- Cossima  Teams to disband: | Note that when you join a team, you should check **first** if it exists, **then** check if the user is already in a team:  Tatyana has created CloneClub, and then she tries to join a non-existing team – so a message for the non-existing team is shown. |