**C#Advanced**

**Defining Classes,Generics**

Problems with exercise and homework for the ["C# Advanced" course @ Software University.](https://softuni.bg/trainings/3584/csharp-advanced-january-2022)

* You can check your solutions in [Judge](https://judge.softuni.org/Contests/1479/Defining-Classes-Exercise)
* **Car Salesman (Defining Classes)**

Define two classes **Car** and **Engine.**

Start by defining a class **Car** that holds information about:

* **Model:** a **string property.**
* **Engine:** a **property holding the engine object.**
* **Weight**: an **int property, it is optional.**
* **Color:** a **string property, it is optional.**

Next, the **Engine class** has the following properties:

* **Model:** a **string property.**
* **Power:** an **int property.**
* **Displacement:** an **int property, it is optional.**
* **Efficiency:** a **string property, it is optional.**

**Input**

* 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}**"
* Keep in mind that **"displacement" and "efficiency"** are optional**,** they **could be missing** from the command.
* Next, you will receive a number **M,** which will specify how many lines of **car** you will receive.
* On each of the next **M** lines, you will receive information about a **Car** in the following format: "**{model} {engine} {weight} {color}**".
* Keep in mind that "**weight" and "color" are optional,** they could **be missing** from the command.
* The **"engine"** will always 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: The optional properties **might be missing** from the formats.

**Output**

Your task is to **print** all the **cars** in the order they were received and their information in the format defined below. If any of the optional fields are missing, print "**n/a**" in its place:

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

**Bonus\***

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 |

* **Threeuple (Generics)**

Create a Class **Threeuple**. Its name is telling us, that it will hold no longer, just a pair of objects. The task is simple, our **Threeuple** should **hold three objects**. Make it have getters and setters. You can even extend the previous class

**Input**

The input consists of three lines:

* The first one is holding a name, an address and a town. Format of the input:

**{first name} {last name} {address} {town}**

* The second line is holding a **name**, **beer** **liters**, and a **boolean** variable with value - **drunk** or **not**. Format:

**{name} {liters of beer} {drunk or not}**

* The last line will hold a **name**, a **bank** **balance** (**double**) and a **bank name**. Format:

**{name} {account balance} {bank name}**

**Output**

* Print the Threeuples' objects in format:

**"{firstElement} -> {secondElement} -> {thirdElement}"**

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Adam Smith Wallstreet New York  Mark 18 drunk  Karren 0.10 USBank | Adam Smith -> Wallstreet -> New York  Mark -> 18 -> True  Karren -> 0.1 -> USBank |
| Anatoly Andreevich Kutuzova Kaliningrad  Marley 9 not  Grant 2 NGB | Anatoly Andreevich -> Kutuzova -> Kaliningrad  Marley -> 9 -> False  Grant -> 2 -> NGB |

**Note**: You may extend your previous solution.