# Problem 1 – Estates

A real estate agency operates with several types of estates: **apartments**, **offices**, **houses** and **garages**. It holds a database of **sale offers** and **rent offers**. Each estate has **unique name**, **type** (apartment, office, house or garage), **area** (in square meters) and **location** and can have **furniture** or not. **Apartments** and **offices** have additionally number of **rooms** and may have **elevator** in the building or not. **Houses** have additionally number of **floors**. **Garages** have additionally **width** and **height** (in meters). **Sale offers** hold an **estate** for sale and **sale** **price**. **Rent offers** hold an **estate** for rental and **rental** **price**.

### Input Source Code

You are given a Visual Studio C# project (source code) holding a **set of interfaces** for the estates and for the offers and an **engine** that executes the following **commands** (see the sample input and output below):

* **create Apartment name area location isFurnitured numberOfRooms hasElevator** – adds an apartment in the database by given unique name and other parameters.
* **create Office name area location isFurnitured numberOfRooms hasElevator** – adds an office by given unique name and other parameters.
* **create House name area location isFurnitured numberOfFloors** – adds a house.
* **create Garage name area location isFurnitured width height** – adds a garage.
* **create RentOffer estateName rentPrice** – adds a rent offer for existing estate and given price.
* **create SaleOffer estateName salePrice** – adds a sale offer for existing estate and given price.
* **status** – prints all **estates** and **offers** from the database (in the order of their creation) in the format like in the sample output below. The engine knows how to find and print all estates and offers. You need to implement just the printing of each individual estate and offer.

public override string ExecuteCommand(string cmdName, string[] cmdArgs)

{

switch (cmdName)

{

case "find-rents-by-location":

return this.ExecuteFindRentsByLocation(cmdArgs[0]);

//break;

case "find-rents-by-price":

return this.ExecuteRentsByPrice(cmdArgs[0], cmdArgs[1]);

default:

return base.ExecuteCommand(cmdName, cmdArgs);

}

* }
* **find-sales-by-location location** – finds all sale offers for the specified location (case-sensitively), ordered by name and prints them in the format like in the sample output below. Prints "No offers" in case of no matches.
* **end** – indicates the end of the input commands. Stops the engine execution.

### Design the Class Hierarchy

Your **first** **task** is to **design an object-oriented class hierarchy** to model the real estates agency, estates and offers using the **best practices** for object-oriented design (OOD) and object-oriented programming (OOP). **Avoid duplicated code** though abstraction, inheritance, and polymorphism and encapsulate correctly all fields.

You are allowed to change the content of the "**Data**" folder only, in the namespace "**Estates.Data**". You are **not allowed to change engine and interfaces**. Please don't modify the content of "**Interfaces**" and "**Engine**" folders.

### Implement the Existing Commands

Implement your classes so that the engine **executes correctly all above described commands**. Don't modify the engine and the existing interfaces. Create your classes in the "**Data**" directory.

### Implement Additional Commands

* **find-rents-by-location location** – finds and prints all rent offers for the specified location (case-sensitively), ordered by name, in the format like in the sample output below.
* **find-rents-by-price minPrice maxPrice** – prints all rent offers within the specified price range (inclusively), ordered by price and by name (as second criteria), in the format like in the sample below.
* private string ExecuteFindRentsByLocation(string location)
* {
* var rentsByLocation = this.Offers
* .Where(x => x.Estate.Location == location && x.Type == OfferType.Rent)
* .OrderBy(x => x.Estate.Name);
* return FormatQueryResults(rentsByLocation);
* }
* private string ExecuteRentsByPrice(string minPrice, string maxPrice)
* {
* decimal minP = decimal.Parse(minPrice);
* decimal maxP = decimal.Parse(maxPrice);
* var rentsByPrice = this.Offers
* .Where(x => x.Type == OfferType.Rent)
* .Where(x => ((RentOffer) x).PricePerMonth >= minP &&
* ((RentOffer) x).PricePerMonth <= maxP)
* .OrderBy(x => ((RentOffer) x).PricePerMonth)
* .ThenBy(x => ((RentOffer) x).Estate.Name);
* return FormatQueryResults(rentsByPrice);
* }

You are **not allowed to modify the existing engine** but you may use other OOP techniques to add functionality to it.

Print all numbers in the default format for their data type defined in the system interfaces (**int** / **double** / **decimal**). The decimal separator is "**.**".

### Constraints

* Estate **area** should be in range [0…10000].
* Office / apartment **rooms** should be in range [0…20].
* House **floors** should be in range [0…10].
* Garage **widths** and **heights** should be in range [0…500].

### Sample Input

|  |
| --- |
| create Apartment aptLozenec24 150 Sofia true 4 true  create Apartment aptBotev28 54 Sofia true 2 false  status  create Office officeVitosha44 70 Sofia true 1 false  create Office officePlovdiv 44 Plovdiv false 1 true  create House houseBankya 206.40 Bankya true 3  create House houseSofia 120 Sofia true 1  create Garage garageLozenec 18 Sofia false 3 6  create RentOffer aptLozenec24 750.00  create SaleOffer aptLozenec24 195000  create RentOffer aptLozenec24 720.00  create SaleOffer officeVitosha44 96000  create RentOffer officeVitosha44 720.0  create RentOffer officePlovdiv 450.50  create SaleOffer houseBankya 320000  create RentOffer houseBankya 950  create RentOffer garageLozenec 100  create RentOffer garageLozenec 120  create SaleOffer garageLozenec 12000  create SaleOffer garageLozenec 11000  create RentOffer garageLozenec 720  status  find-sales-by-location Sofia  find-rents-by-location Sofia  find-rents-by-price 700 1000  find-rents-by-price 0 99  end |

Note: the engine skips all empty input lines.

### Sample Output

|  |
| --- |
| Apartment created.  Apartment created.  Estates:  Apartment: Name = aptLozenec24, Area = 150, Location = Sofia, Furnitured = Yes, Rooms: 4, Elevator: Yes  Apartment: Name = aptBotev28, Area = 54, Location = Sofia, Furnitured = Yes, Rooms: 2, Elevator: No  No offers  Office created.  Office created.  House created.  House created.  Garage created.  RentOffer created.  SaleOffer created.  RentOffer created.  SaleOffer created.  RentOffer created.  RentOffer created.  SaleOffer created.  RentOffer created.  RentOffer created.  RentOffer created.  SaleOffer created.  SaleOffer created.  RentOffer created.  Estates:  Apartment: Name = aptLozenec24, Area = 150, Location = Sofia, Furnitured = Yes, Rooms: 4, Elevator: Yes  Apartment: Name = aptBotev28, Area = 54, Location = Sofia, Furnitured = Yes, Rooms: 2, Elevator: No  Office: Name = officeVitosha44, Area = 70, Location = Sofia, Furnitured = Yes, Rooms: 1, Elevator: No  Office: Name = officePlovdiv, Area = 44, Location = Plovdiv, Furnitured = No, Rooms: 1, Elevator: Yes  House: Name = houseBankya, Area = 206.4, Location = Bankya, Furnitured = Yes, Floors: 3  House: Name = houseSofia, Area = 120, Location = Sofia, Furnitured = Yes, Floors: 1  Garage: Name = garageLozenec, Area = 18, Location = Sofia, Furnitured = No, Width: 3, Height: 6  Offers:  Rent: Estate = aptLozenec24, Location = Sofia, Price = 750.00  Sale: Estate = aptLozenec24, Location = Sofia, Price = 195000  Rent: Estate = aptLozenec24, Location = Sofia, Price = 720.00  Sale: Estate = officeVitosha44, Location = Sofia, Price = 96000  Rent: Estate = officeVitosha44, Location = Sofia, Price = 720.0  Rent: Estate = officePlovdiv, Location = Plovdiv, Price = 450.50  Sale: Estate = houseBankya, Location = Bankya, Price = 320000  Rent: Estate = houseBankya, Location = Bankya, Price = 950  Rent: Estate = garageLozenec, Location = Sofia, Price = 100  Rent: Estate = garageLozenec, Location = Sofia, Price = 120  Sale: Estate = garageLozenec, Location = Sofia, Price = 12000  Sale: Estate = garageLozenec, Location = Sofia, Price = 11000  Rent: Estate = garageLozenec, Location = Sofia, Price = 720  Query Results:  [Estate: aptLozenec24, Location: Sofia, Price: 195000]  [Estate: garageLozenec, Location: Sofia, Price: 12000]  [Estate: garageLozenec, Location: Sofia, Price: 11000]  [Estate: officeVitosha44, Location: Sofia, Price: 96000]  Query Results:  [Estate: aptLozenec24, Location: Sofia, Price: 750.00]  [Estate: aptLozenec24, Location: Sofia, Price: 720.00]  [Estate: garageLozenec, Location: Sofia, Price: 100]  [Estate: garageLozenec, Location: Sofia, Price: 120]  [Estate: garageLozenec, Location: Sofia, Price: 720]  [Estate: officeVitosha44, Location: Sofia, Price: 720.0]  Query Results:  [Estate: aptLozenec24, Location: Sofia, Price: 720.00]  [Estate: garageLozenec, Location: Sofia, Price: 720]  [Estate: officeVitosha44, Location: Sofia, Price: 720.0]  [Estate: aptLozenec24, Location: Sofia, Price: 750.00]  [Estate: houseBankya, Location: Bankya, Price: 950]  No Results |