# 03. Estate Agency

A black and white map with a house on it

Description automatically generated

*Every estate agency needs to manage a database of real estates to operate effectively. The real estate database is crucial for the daily work of agents in an estate agency. It allows for the addition, deletion, and searching of real estates by various features. We need your help to implement and restore several methods that have disappeared from the program directories.*

**Preparation**

Download the skeleton provided in Judge. **Do not** change the **StartUp** class or its **namespace**.

**Pay attention to name the project EstateAgency, 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 real estates** by creating the classes described below.

### RealEstate

You are given a class **RealEstate** with the following properties:

* **Address – string**
* **PostalCode – string**
* **Price – decimal**
* **Size – double** (representing the size in square meters)

The class **constructor** should receive **address, postalCode, price** and **size**.

Override the **ToString()** method in the following format:  
**"Address: {Address}, PostalCode: {PostalCode}, Price: ${Price}, Size: {Size} sq.m."**

### EstateEgency

**Next**, you are given a class **EstateAgency** that has **RealEstates** (a List that stores **RealEstate** objects). The **EstateAgency** class should have the following **properties**:

* **Capacity - int**
* **RealEstates – List<RealEstate>**
* **Count - int** - **returns** the **number** of real estates, in the **EstateAgency**.

The class **constructor** should receive **capacity** and initialize the **RealEstates** with a new instance of the collection.

Implement the following features:

* **Method AddRealEstate(RealEstate realEstate)**  
  – **adds** a **new real estate** to the **collection** of **RealEstates**, **if** the **Capacity** **allows it**. If there is a **RealEstate** with the **same Address, already added**, do not duplicate real estates, just **skip the command.**
* **Method RemoveRealEstate(string address)**  
  – **removes** a real estate **by** the **given address,** if such **exists**, and **returns boolean**(**true** if it is removed, otherwise – **false**)
* **Method GetRealEstates(string postalCode)**– **returns** **a list** of **all real estates with** the **specified postal code.**
* **Method GetCheapest()**– **returns** the **real estate** **with the lowest price**.
* **Method GetLargest()**– **returns** the **Area** of the real estate **with the largest size.**
* **Method EstateReport()**– **returns** a **string** with all estates in the following **format**:
  + **"Real estates available:  
    {RealEstate1}  
    {RealEstate2}**

**(…)**

**{RealEstaten}"**

**Constraints**

* The **addresses** of the real estates will be **always unique**.
* You will **always have real estates added** before receiving methods, manipulating the real estates in the Estate Agency.

**Examples**

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

|  |
| --- |
| **Sample code usage** |
| // Initialize the repository (EstateAgency)  EstateAgency estateAgency = new EstateAgency(5);  // Initialize entities (RealEstate)  RealEstate home = new RealEstate("123 Main St", "12345", 250000.0m, 120.0);  RealEstate office = new RealEstate("456 Business Rd", "67890", 500000.0m, 300.0);  RealEstate warehouse = new RealEstate("789 Industrial Ave", "10111", 750000.0m, 500.0);  RealEstate store = new RealEstate("321 Market St", "12131", 200000.0m, 80.0);  RealEstate apartment = new RealEstate("654 Elm St", "12131", 180000.0m, 70.0);  // Get Count  Console.WriteLine(estateAgency.Count); // 0  // Add RealEstates  estateAgency.AddRealEstate(home);  estateAgency.AddRealEstate(office);  estateAgency.AddRealEstate(warehouse);  estateAgency.AddRealEstate(store);  estateAgency.AddRealEstate(apartment);  // Try to add real estate when the capacity is full  RealEstate villa = new RealEstate("987 Luxury Ln", "16171", 1000000.0m, 600.0);  estateAgency.AddRealEstate(villa); // Should not add as capacity is full  // Get Count  Console.WriteLine(estateAgency.Count); // 5  // Remove RealEstate  Console.WriteLine(estateAgency.RemoveRealEstate("987 Luxury Ln")); // False  Console.WriteLine(estateAgency.RemoveRealEstate("123 Main St")); // True  // GetRealEstate  Console.WriteLine(string.Join(Environment.NewLine, estateAgency.GetRealEstates("12131")));  // Address: 321 Market St, PostalCode: 12131, Price: $200000.0, Size: 80 sq.m.  // Address: 654 Elm St, PostalCode: 12131, Price: $180000.0, Size: 70 sq.m.  // Get Cheapest RealEstate  Console.WriteLine(estateAgency.GetCheapest());  // Address: 654 Elm St, PostalCode: 14151, Price: $180000.0, Size: 70 sq.m.  // Get Largest RealEstate  Console.WriteLine(estateAgency.GetLargest());  // 500  // Real Estates Report  Console.WriteLine(estateAgency.EstateReport());  // Real estates available:  // Address: 456 Business Rd, PostalCode: 67890, Price: $500000.0, Size: 300 sq.m.  // Address: 789 Industrial Ave, PostalCode: 10111, Price: $750000.0, Size: 500 sq.m.  // Address: 321 Market St, PostalCode: 12131, Price: $200000.0, Size: 80 sq.m.  // Address: 654 Elm St, PostalCode: 14151, Price: $180000.0, Size: 70 sq.m. |

**Submission**

Zip all the files in the project folder except **bin** and **obj** folders.