

**Shoe Store**

* **Preparation**

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

* **Problem description**

*Write a program that keeps track of the shoes in a shoe store.*

**Shoe**

You are given a class **Shoe,** create the following properties:

* **Brand – string**
* **Type – string**
* **Size – double**
* **Material – string**

The class **constructor** should receive **brand, type, size and material**.

Override **ToString()** method: "Size {Size}, {Material} {Brand} {Type} shoe."

**ShoeStore**

Next, you are given a class named **ShoeStore,** which has a **collection**of type **Shoe**. The name of the collection should be **Shoes, which could not be modified directly**. All the entities of the shoe collection have the **same** properties. The **ShoeStore** also should have the following properties:

* **Name – string**
* **StorageCapacity – int**
* **Shoes – List<Shoe>**

The **constructor** of the **ShoeStore** class should receive **name and storageCapacity**. It should initialize also the **Shoes** with a new instance of the collection.

Implement the following features:

* Getter **Count** - returns the **total count** of the shoes in the storage room.
* **string AddShoe(Shoe shoe)** – **adds** a **Shoe** to the **Shoes** collection and returns: **"**Successfully added {shoeType} {shoeMaterial} pair of shoes to the store.**"**
* If the **StorageCapacity** doesn’t allow adding more shoes in the **Store**

(Shoes.Count == StorageCapacity), returns: **"**No more space in the storage room.**"**

* **int RemoveShoes(string material)** – removes all shoes by a **given material,** as a result, return the **count** of the shoes which were removed**.**
* **List<Shoe> GetShoesByType(string type)** – searches and returns a **list of shoes** by **given type. Search should be case insensitive.**
* **Shoe GetShoeBySize(double size)** – return the first shoe, with the given size**.**
* **string StockList(double size, string type)** –returns a string with information about the shoes which match the given size and typein the following format:
* If **there are** any **pairs mathcing** the given parameters, print the following report on the Console:

**"Stock list for size {size} - {type} shoes:  
{Shoe1}  
{Shoe2}  
{…}**"

* If **none of the pairs match** the given parameters, print the following message on the console:

**"No matches found!**"

**Note: Do not use** "\n\r" **for a new line.**

* **Constraints**
* You will always have a pair of shoes added before receiving methods that manipulate the **shoes in the ShoeStore**.
* **Examples**

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

|  |
| --- |
| **Sample code usage** |
| var store = new ShoeStore("SportiveNation", 10);  var shoeOne = new Shoe("Nike", "running", 42.5, "textile");  var shoeTwo = new Shoe("Salomon", "hiking", 42, "textile");  var shoeThree = new Shoe("Reebok", "running", 38, "textile");  var shoeFour = new Shoe("LaCoste", "casual", 40.5, "leather");  var shoeFive = new Shoe("Adidas", "casual", 39, "textile");  var shoeSix = new Shoe("Nike", "hiking", 42.5, "textile");  var shoeSeven = new Shoe("Adidas", "casual", 42, "leather");  var shoeEight = new Shoe("AirJordan", "running", 42, "leather");  var shoeNine = new Shoe("CalninKlein", "casual", 41.5, "leather");  var shoeTen = new Shoe("Puma", "hiking", 42, "textile");  var shoeEleven = new Shoe("Skechers", "casual", 42.5, "leather");  Console.WriteLine(store.AddShoe(shoeOne));  // Successfully added running textile pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeTwo));  // Successfully added hiking textile pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeThree));  // Successfully added running textile pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeFour));  // Successfully added casual leather pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeFive));  // Successfully added casual textile pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeSix));  // Successfully added hiking textile pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeSeven));  // Successfully added casual leather pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeEight));  // Successfully added running leather pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeNine));  // Successfully added casual leather pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeTen));  // Successfully added hiking textile pair of shoes to the store.  Console.WriteLine(store.AddShoe(shoeEleven));  // No more space in the storage room.  var runningShoes = store.GetShoesByType("Running");  var hikingShoes = store.GetShoesByType("hIKING");  Console.WriteLine(string.Join($"{Environment.NewLine}", runningShoes));  // Size 42.5, textile Nike running shoe.  // Size 38, textile Reebok running shoe.  // Size 42, leather AirJordan running shoe.  Console.WriteLine(string.Join($"{Environment.NewLine}", hikingShoes));  // Size 42, textile Salomon hiking shoe.  // Size 42.5, textile Nike hiking shoe.  // Size 42, textile Puma hiking shoe.  Console.WriteLine(store.RemoveShoes("leather"));  // 4  var shoeBySize = store.GetShoeBySize(42.5);  Console.WriteLine(shoeBySize);  // Size 42.5, textile Nike running shoe.  Console.WriteLine(store.StockList(42, "hiking"));  //Stock list for size 42 - hiking shoes:  //Size 42, textile Salomon hiking shoe.  //Size 42, textile Puma hiking shoe. |

* **Submission**

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