# 03. Clothing Magazine



*You are an appraiser and you have to audit clothing magazines. Let's get started!*

1. **Preparation**

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

**Pay attention to name the project ClothingMagazine, 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.**

1. **Problem Description**

Your task is to create a repository that stores clothes by creating the classes described below.

### Cloth

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

* **Color - string**
* **Size - int**
* **Type - string**

The class **constructor** should receive **color, size and type**.

Override the **ToString()** method in the following format:  
**"Product: {type} with size {size}, color {color}"**

### Magazine

**Next**, you are given a class **Magazine** that has **Clothes** (a List that stores the entity **Cloth**). All entities inside the repository have the **same properties**. The **Magazine** class should have the following **properties**:

* **Type - string**
* **Capacity – int**
* **Clothes – List<Cloth>**

The class **constructor** should receive **type** and **capacity**, also it should initialize the **Clothes** with a new instance of the collection.Implement the following features:

* **Method AddCloth(Cloth cloth)** – **adds** an **entity** to the collection **if** **there** **is** **room** for it
* **Method RemoveCloth(string color)** – removes a cloth by **given color,** if such **exists**, and **returns boolean** (**true** if it is removed, otherwise – **false**)
* **Method GetSmallestCloth()** – **returns the Cloth with the smallest Size**
* **Method GetCloth(string color)** – **returns** the **Cloth** with the **given color**
* **Method GetClothCount()** – **returns** the **number** of **clothes**
* **Method Report()** – **returns** a **string** in the following **format** (print the clothes in **ordered by Size**):
  + **"{type} magazine contains:  
    {Cloth1}  
    {Cloth2}  
    (…)"**

1. **Constraints**

* The **color** and **size** of the clothes will be **always unique**.
* You will always have clothes added before receiving methods manipulating the Magazines’ clothes.

1. **Examples**

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

|  |
| --- |
| **Sample code usage** |
| //Initialize the repository (Magazine)Magazine magazine = **new** Magazine(**"**Zara**"**, 20);  //Initialize entity (Cloth)Cloth cloth1 = **new** Cloth(**"**red**"**, 36, **"**dress**"**);  //Print ClothConsole.WriteLine(cloth1); //Product: dress with size 36, color red//Add Clothmagazine.AddCloth(cloth1);  //Remove ClothConsole.WriteLine(magazine.RemoveCloth(**"**black**"**)); //falseCloth cloth2 = **new** Cloth(**"**brown**"**, 34, **"**t-shirt**"**);  Cloth cloth3 = **new** Cloth("blue", 32, "jeans");  //Add Clothmagazine.AddCloth(cloth2);  magazine.AddCloth(cloth3);  //Get smallest clothCloth smallestCloth = magazine.GetSmallestCloth(); Console.WriteLine(smallestCloth); //Product: jeans with size 32, color blue  //Get ClothCloth getCloth = magazine.GetCloth(**"**brown**"**); //Product: t-shirt with size 34, color brownConsole.WriteLine(getCloth);   Console.WriteLine(magazine.Report()); //Zara magazine contains:  //Product: jeans with size 32, color blue  //Product: t-shirt with size 34, color brown //Product: dress with size 36, color red |

1. **Submission**

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