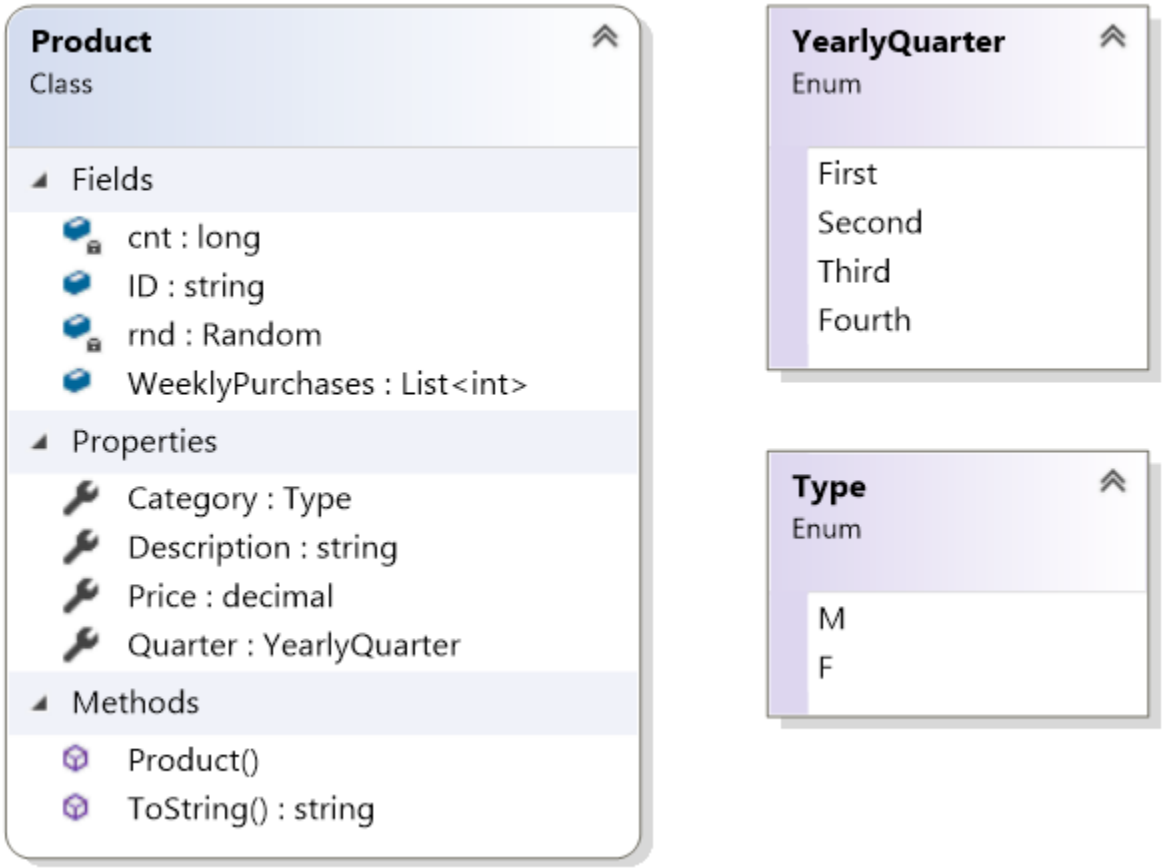
Sofia University
Department of Mathematics and Informatics

Course : OO Programming with C#.NET
Date: 11/12 2020
Student Name:

Домашно No. 4



Problem 1.

Create `class Product` and `enum` types `Type` and `YearQuarter` according to the above UML class diagram. The `ToString`() method returns a `string` composed by the `ID` and the `WeeklyPurchases.` The **constants** of `enum YearlyQuarter` are initialized sequentially to `1,2,3` and `4.` The **constants** of `enum Type` are initialized sequentially to `chars` `'M'` and `'F'` respectively.

Class `Product` has a **general purpose constructor** for initializing properties `Description`, `Category`, `WeeklyPurchases` and `Price`. Property `Quarter` of each `Product` object are initialized at random with the `static` object referenced by `rnd`. Each `Product` instance has an unique `ID` that is composed by prefix provided by the value of the `Category` property and **6- digit** number, **where insignificant digits are replaced by zeros** (for example, `F000101` or `M0123040`).

Initialize the **`static`** datamember **`products`** to a **`List`** of products, where the values of the above properties are provided in the following table

Sample data

Description	Category	WeeklyPurchases	Price
`Electric sander`	`M`	`99, 82, 81, 79`	**`157.98`**
`Power saw`	`M`	`99, 86, 90, 94`	**`99.99`**
`Sledge hammer`	`F`	`93, 92, 80, 87`	**`21.50`**
`Hammer`	`M`	`97, 89, 85, 82`	**`11.99`**
`Lawn mower`	`F`	`35, 72, 91, 70`	**`139.50`**
`Screwdriver`	`F`	`88, 94, 65, 91`	**`56.99`**
`Jig saw`	**`M`**	**`75, 84, 91, 39`**	**`11.00`**
`Wrench`	**`F`**	**`97, 92, 81, 60`**	**`17.50`**
`Sledge hammer`	**`M`**	**`75, 84, 91, 39`**	**`21.50`**
`Hammer`	`F`	`94, 92, 91, 91`	**`11.99`**
`Lawn mower`	`M`	`96, 85, 91, 60`	**`179.50`**
`Screwdriver`	`M`	`96, 85, 51, 30`	**`66.99`**

Write the following LINQ statements:

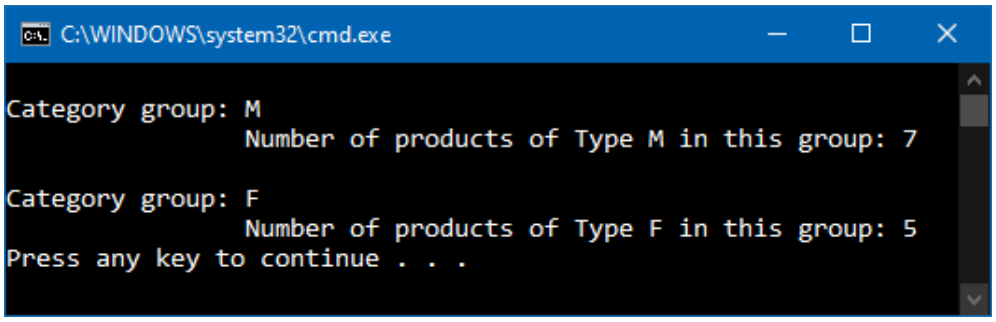
(For references see GroupBy and Query a collection of objects)

a) Write a method

```
public static void GroupByCategoryCountDescending()
```

to declare a LINQ statement that groups **`products`** by **`Category`** and shows the total number (count) of all the **`products`** in each group, where the **`Category`** groups appear sorted in **`descending`** order of the total number value. Evaluate the LINQ statement inside the method and run the method to test the execution.

Expected sample output

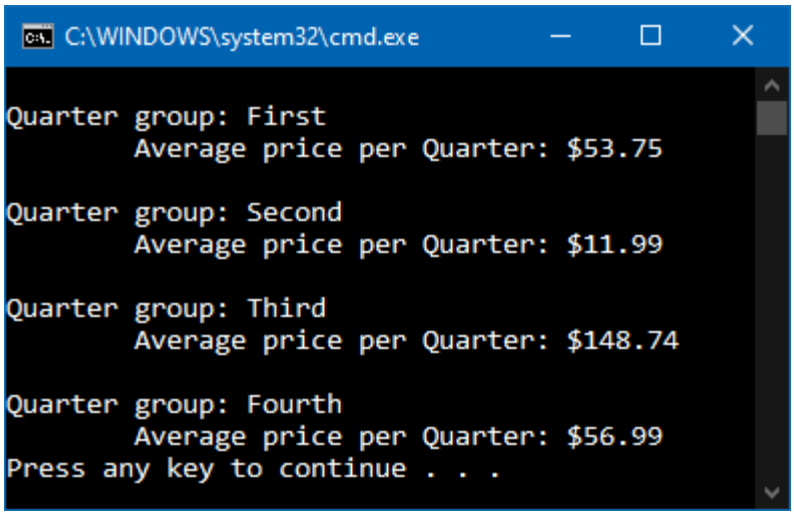


b) Write a method

```
public static void GroupByQtrAndProductPriceAvg()
```

to declare a LINQ statement that groups **`products`** by **`Quarter`** and shows the **`average`** price of all the **`products`** in each group, where the **`Quarters`** are sorted in **`ascending`** order. Evaluate the LINQ statement inside the method and run the method to test the execution.

Expected sample output

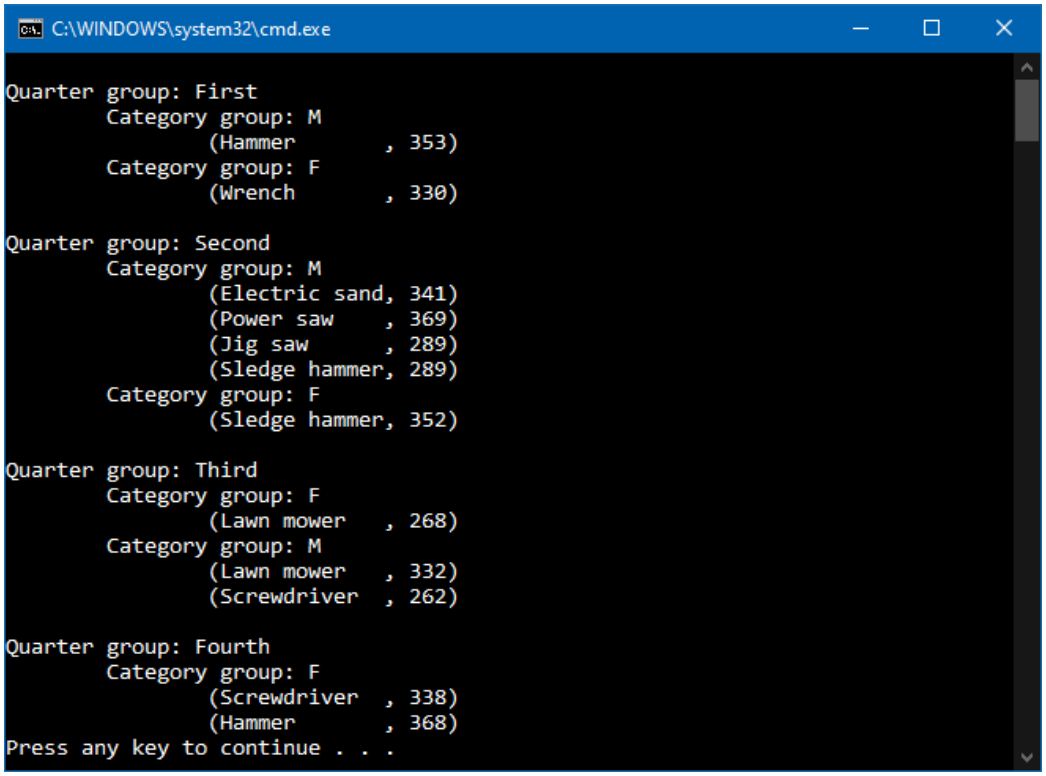


c) Write a method

```
public static void GroupByQtrCategoryWeeklySum()
```

to declare a LINQ statement that groups **`products`** by **`Quarter`** and next by **`Category`** in each **`Quarter`**. The **`Quarter`** groups must be sorted in **`ascending`** order of the **`Quarter`** and the **`Category`** groups show the **`tuple`** of **`Description`** and the total sum of **`WeeklyPurchases`** for each **`Product`** in the respective **`Quarter`**/**`Category`** group... Evaluate the LINQ statement inside the method and run the method to test the execution.

Expected sample output

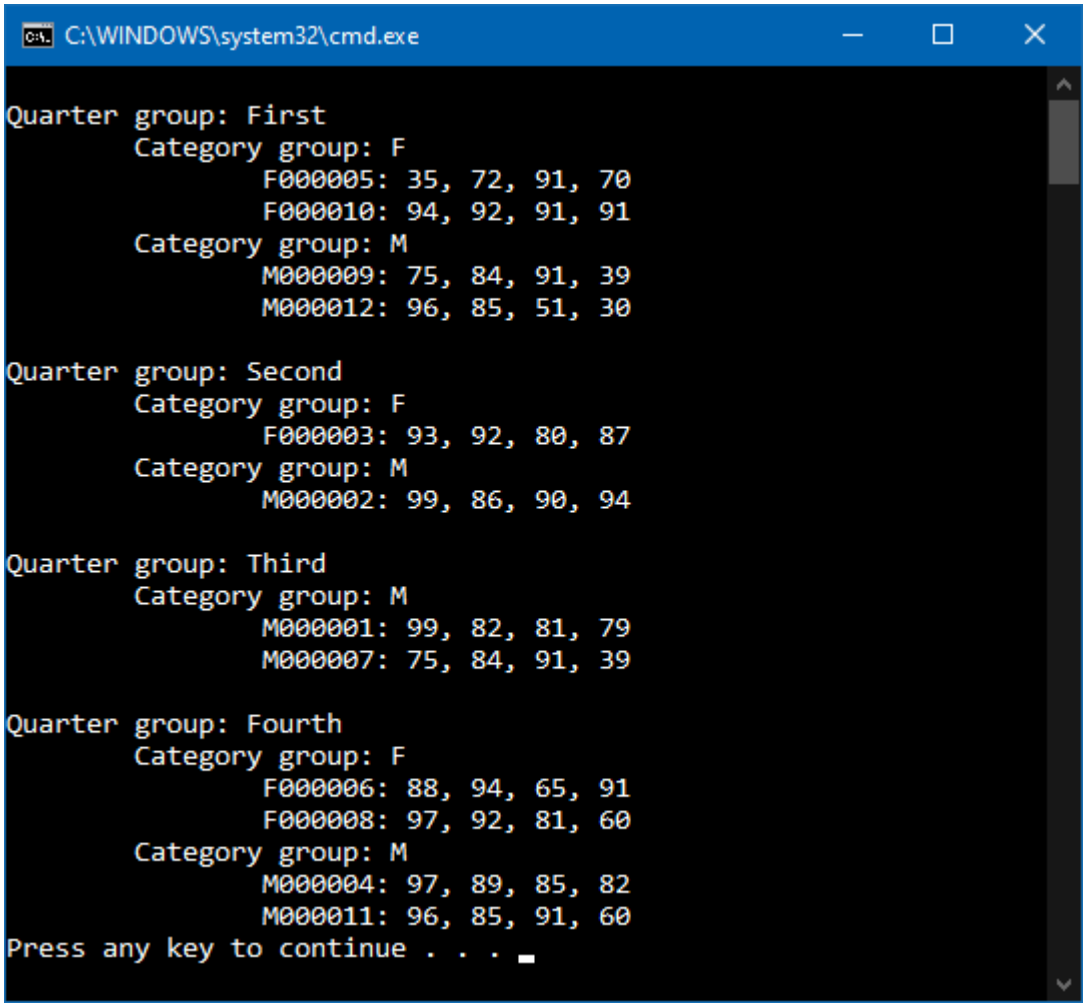


d) Write a method

```
public static void GroupByQtrCategoryAndProducts()
```

to declare a LINQ statement that groups **`products`** by **`Quarter`** and next by **`Category`** in each **`Quarter`**. The **`Quarter`** groups must be sorted in ascending order of the **`Quarter`** and the **`Category`** groups in ascending order of the **`Category`**. Show all the **`products`** in each **`Category`** group sorted in **`ascending`** order of the **`Category`** using the **`ToString`**() method of **`class Product`** . Evaluate the LINQ statement inside the method and run the method to test the execution.

Expected sample output



e) Write a method

```
public static void GroupByQtrMinMaxPrice()
```

to declare a LINQ statement that groups **`products`** by **`Quarter`**. The Quarter groups must be sorted in **`ascending`** order of the **`Quarter`** and each **`Quarter`** group shows the **`Min`** and **`Max Price`** per **`Quarter`**. Evaluate the LINQ statement inside the method and run the method to test the execution.

Expected sample output