Taazaa Training

“Assignment - 5”

Topics:-

- **Day5A (Collection with Naming convention)**
- **Dictionary**
  (Insert, Update, Delete and Search)
- **Hash Table**
  (Insert, Update, Delete and Search)
- **Private Constructor**

Submitted by: -

Gurpreet Singh

Day5A (Collection with Naming convention)

Source Code:-

Recipe1.cs under Models folder

```
namespace Day5A.Models
{
    public class Recipe1//Pascal Case
    {
         public int step{get;set;}//camelcase
        public string rName{get;set;}//camelcase
    }
}
```

Recipe2cs.cs

```
using System.Collections;
using System.Collections.Generic;
using Day5A.Models;
namespace Day5A.Recipe
{
    public class Recipe2cs //pascal case
    {
        public ArrayList mangoShake()//camel case
        {
            ArrayList arrayList=new ArrayList();//camel case
            arrayList.Add("Mango");
            arrayList.Add("Ice Cubes");
            arrayList.Add("Milk");
            arrayList.Add("Sugar");
            return arrayList;

        }

        public List<Recipe1> bananaShake()//camel case
        {
            List<Recipe1> Obj=new List<Recipe1>();
            Obj.Add(new Recipe1{
                step=1,//camel case
                rName="Banana"//camel case
            });
            Obj.Add(new Recipe1{
                step=2,//camel case
```

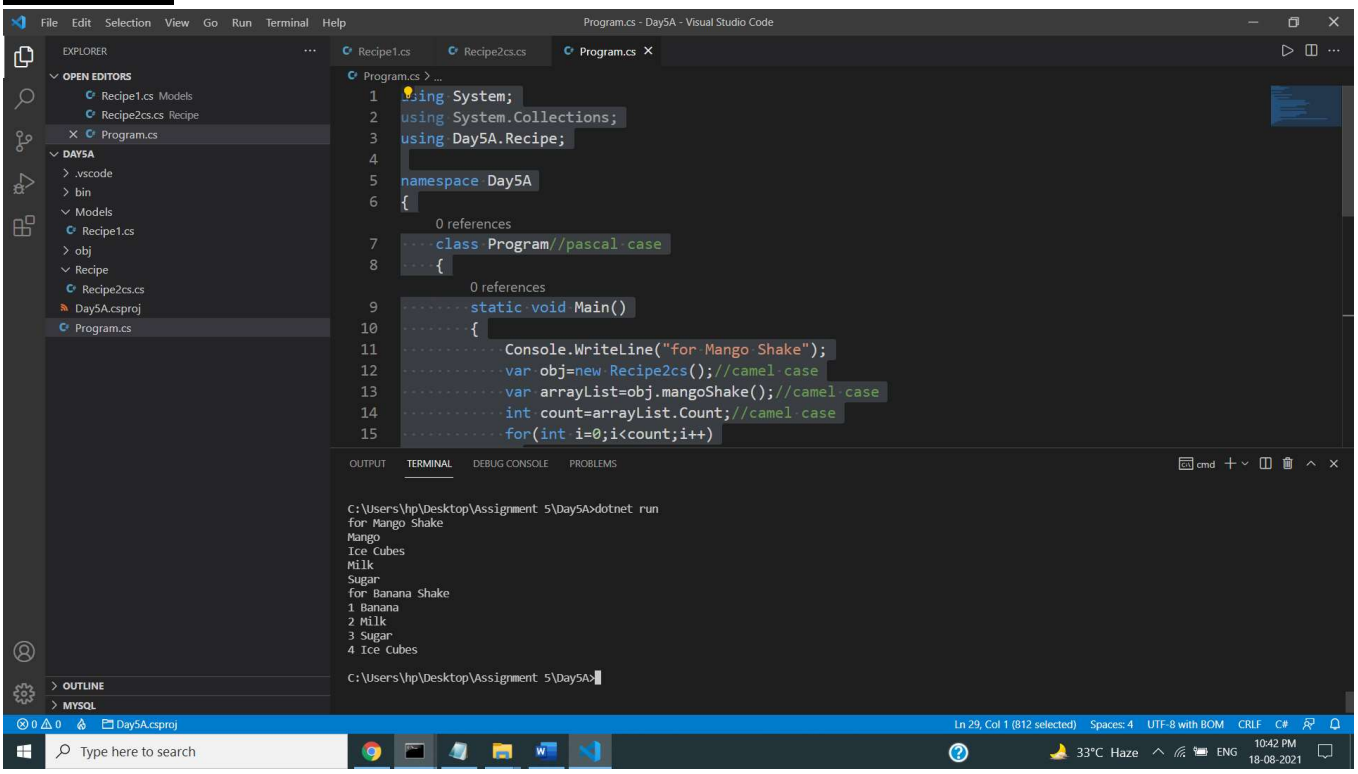
```
                rName="Milk"//camel case
            });
            Obj.Add(new Recipe1{
                step=3,//camel case
                rName="Sugar"//camel case
            });
            Obj.Add(new Recipe1{
                step=4,//camel case
                rName="Ice Cubes"//camel case
            });

            return Obj;
        }

    }
}
```

Progarm.cs

```
using System;
using System.Collections;
using Day5A.Recipe;

namespace Day5A
{
    class Program//pascal case
    {
        static void Main()
        {
            Console.WriteLine("for Mango Shake");
            var obj=new Recipe2cs();//camel case
            var arrayList=obj.mangoShake();//camel case
            int count=arrayList.Count;//camel case
            for(int i=0;i<count;i++)
            {
                Console.WriteLine(arrayList[i]);
            }
            Console.WriteLine("for Banana Shake");
            var list=obj.bananaShake();//camel case
            count=list.Count;//camel case
            for(int i=0;i<count;i++)
            {
```

```
                Console.WriteLine(list[i].step + " "+list[i].rName);//ca
mel case
            }
        }
    }
}
```

Output:-



Dictionary
(Insert, Update, Delete and Search)

Theory:-

The Dictionary<TKey, TValue> is a generic collection that stores key-value pairs in no particular order.

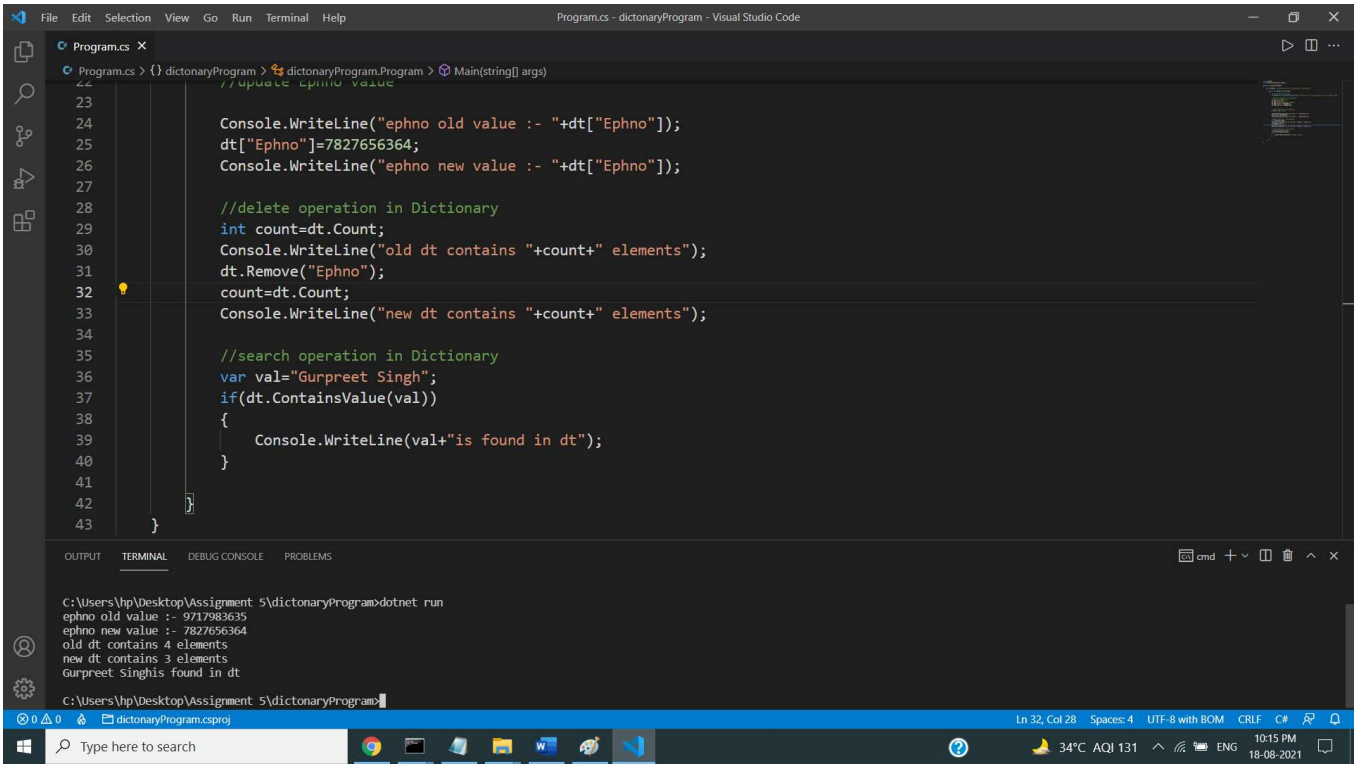
- **It stores key-value pairs.**
- **Comes under System.Collections.Generic namespace.**
- **Implements IDictionary<TKey, TValue> interface.**
- **Keys must be unique and cannot be null.**
- **Values can be null or duplicate.**
- **Values can be accessed by passing associated key in the indexer e.g. myDictionary[key]**
- **Elements are stored as KeyValuePair<TKey, TValue> objects.**

Source Code:-

```
using System;
using System.Collections.Generic;

namespace dictonaryProgram
{
    class Program //implementation of all operations of dictionary
    {
        static void Main(string[] args)
        {
            //declaration of Dictionary
            var dt=new Dictionary<string,object>(); //here key is string
 type and value is of object type

            //Insertion operation in Dictionary
            //inserting value in dt
            dt.Add("Eid",101);
            dt.Add("Ename","Gurpreet Singh");
            dt.Add("Ephno",9717983635);
            dt.Add("Esalary",1000000.00);


            //update operation in Dictionary
            //update Ephno value
```

```
            Console.WriteLine("ephno old value :- "+dt["Ephno"]);
            dt["Ephno"]=7827656364;
            Console.WriteLine("ephno new value :- "+dt["Ephno"]);

            //delete operation in Dictionary
            int count=dt.Count;
            Console.WriteLine("old dt contains "+count+" elements");
            dt.Remove("Ephno");
            count=dt.Count;
            Console.WriteLine("new dt contains "+count+" elements");

            //search operation in Dictionary
            var val="Gurpreet Singh";
            if(dt.ContainsValue(val))
            {
                Console.WriteLine(val+"is found in dt");
            }

        }
    }
}
```

Output:-



Hash Table
(Insert, Update, Delete and Search)

Theory:-

The Hashtable is a non-generic collection that stores key-value pairs, similar to generic Dictionary<TKey, TValue> collection.

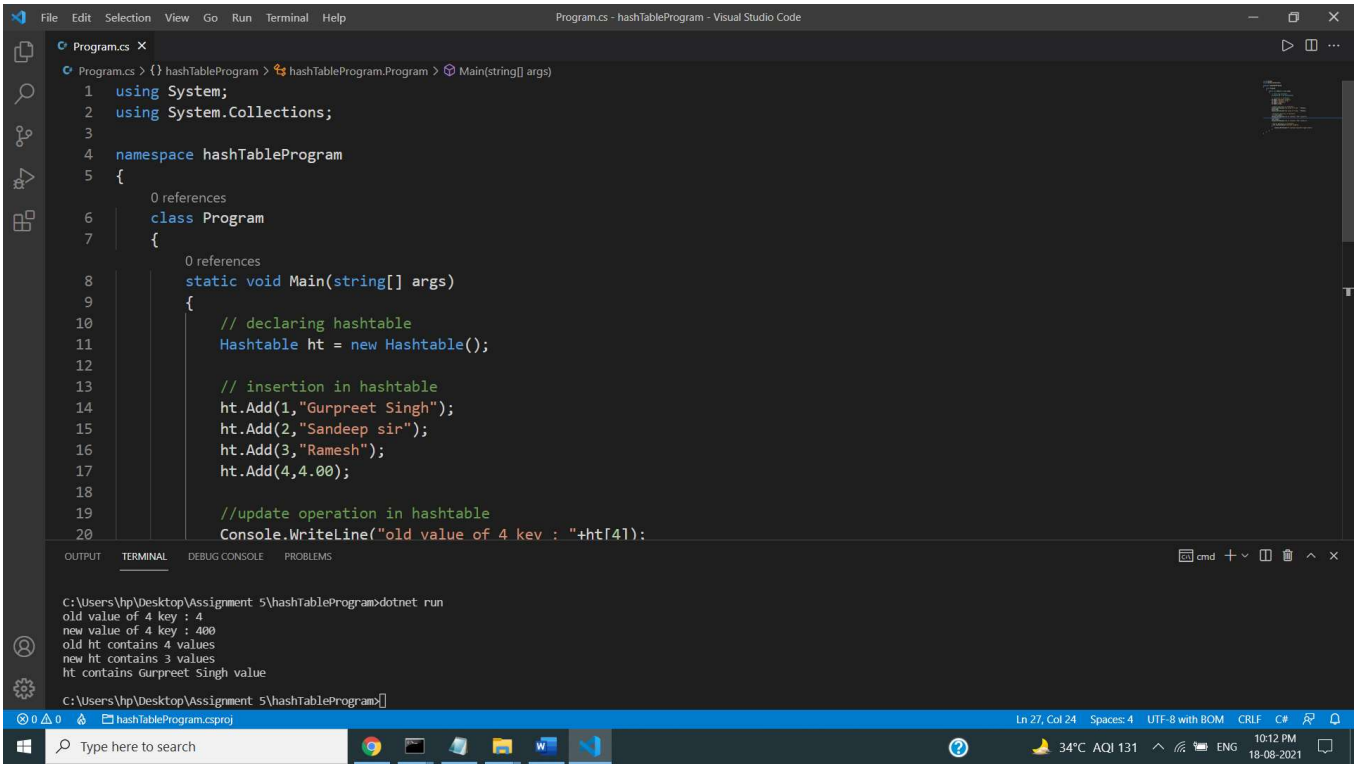
- **Hashtable stores key-value pairs.**
- **Comes under System.Collection namespace.**
- **Implements IDictionary interface.**
- **Keys must be unique and cannot be null.**
- **Values can be null or duplicate.**
- **Values can be accessed by passing associated key in the indexer e.g. myHashtable[key]**
- **Elements are stored as DictionaryEntry objects.**

Source Code:-

```
using System;
using System.Collections;

namespace hashTableProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            // declaring hashtable
            Hashtable ht = new Hashtable();

            // insertion in hashtable
            ht.Add(1,"Gurpreet Singh");
            ht.Add(2,"Sandeep sir");
            ht.Add(3,"Ramesh");
            ht.Add(4,4.00);

            //update operation in hashtable
            Console.WriteLine("old value of 4 key : "+ht[4]);
            ht[4]=400;
            Console.WriteLine("new value of 4 key : "+ht[4]);

            //deletion operation in hashtable
            int n=ht.Count;
```

```
            Console.WriteLine("old ht contains "+n+" values");
            ht.Remove(4);
            n=ht.Count;
            Console.WriteLine("new ht contains "+n+" values");

            //search operation in hashtable
            if(ht.ContainsValue("Gurpreet Singh"))
            {
                Console.WriteLine("ht contains Gurpreet Singh value");
            }
        }
    }
}
```

Output:-



Private Constructor

Theory:-

- **Private constructor is a special instance constructor which is used in a class that contains static member only.**
- **If a class has one or more private constructor and no public constructor then other classes are not allowed to create instance of this class; this means you can neither create the object of the class nor can it be inherited by other classes.**
- **The main purpose of creating private constructor is to restrict the class from being instantiated when it contains every member as static.**

Source Code:-

```
using System;

namespace privateConstructorProgram
{
    class Program
    {
        public static string name;
        public static int num;
        // Creating private Constructor
        // using private keyword
        private Program()
        {
        Console.WriteLine("Welcome to Private Constructor");
        }
        // Default Constructor
        // with parameters
        public Program(string a, int b) {

            name = a;
            num = b;
        }
        public static void Main() {

        // This line raises error because
        // the constructor is inaccessible
        // Program obj1 = new Program();

        // Here, the only default
        // constructor will invoke
        Program obj2 = new Program("Gurpreet", 101);
```

```
        // Here, the data members of Geeks
        // class are directly accessed
        // because they are static members
        // and static members are accessed
        // directly with the class name
        Console.WriteLine(Program.name + ", " + Program.num);
    }

    }
}
```

Output:-