



Static Attributes and Static Methods



Revision

Previously, we
had created this
MUser Class for
SignIn SignUp
application

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public bool isAdmin()
    {
        if (userRole == "Admin")
        {
            return true;
        }
        return false;
    }
}
```

Revision

We declared the **usersList** in the driver program instead of the **MUser** class.

```
static List<MUser> usersList = new List<MUser>();
```



```
static void Main(string[] args){
    string path = "Data.txt";
    readDataFromFile(path);
    int option = 0;
    while (option != 3){
        Console.Clear();
        option = menu();
        if (option == 1){
            MUser user = SignIn();
            if (user != null){
                if (user.isAdmin()){
                    Console.WriteLine("This is Admin");
                    //Admin Menu
                }
                else{
                    Console.WriteLine("This is User");
                    //User Menu
                }
            }
        }
        else if (option == 2){
            MUser user = TakeInputFromConsole();
            addUserIntoList(user);
            storeUserIntoFile(user, path);
        }
        Console.ReadKey();
    }
}
```

Revision

This is not a good approach as all the data and related functions should be in the same class.

```
static List<MUser> userList = new List<MUser>();
```



```
static void Main(string[] args){
    string path = "Data.txt";
    readDataFromFile(path);
    int option = 0;
    while (option != 3){
        Console.Clear();
        option = menu();
        if (option == 1){
            MUser user = SignIn();
            if (user != null){
                if (user.isAdmin()){
                    Console.WriteLine("This is Admin");
                    //Admin Menu
                }
                else{
                    Console.WriteLine("This is User");
                    //User Menu
                }
            }
        }
        else if (option == 2){
            MUser user = TakeInputFromConsole();
            addUserIntoList(user);
            storeUserIntoFile(user, path);
        }
        Console.ReadKey();
    }
}
```


Revision

What if we declared the `usersList` in the `MUsers` class instead of the driver program.

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }
}
```



Revision


And also write
the
addUserIntoList
function in the
class.

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public void addUserIntoList(MUser user)
    {
        usersList.Add(user);
    }
}
```



Revision


Any Problem with
this Approach?

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public void addUserIntoList(MUser user)
    {
        usersList.Add(user);
    }
}
```



MUsers


When we create a user of this class it will also create a **usersList** into the memory.

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public void addUserIntoList(MUser user)
    {
        usersList.Add(user);
    }
}
```



MUsers

For all u1, u2,
u3 and u4,
separate lists
are created and
these lists are
disjoints.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    u1.addUserIntoList(u1);
    u1.addUserIntoList(u2);
    u2.addUserIntoList(u3);
    u2.addUserIntoList(u4);
}
```

MUsers

For all u1, u2,
u3 and u4,
separate lists
are created and
these lists are
disjoints.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    u1.addUserIntoList(u1);
    u1.addUserIntoList(u2);
    u2.addUserIntoList(u3);
    u2.addUserIntoList(u4);
    Console.WriteLine("U1 List");
    u1.printList();
    Console.WriteLine("U2 List");
    u2.printList();
    Console.ReadKey();
}
```

MUsers

For all u1, u2,
u3 and u4,
separate lists
are created and
these lists are
disjoints.

```
U1 List
Fatima 123
Khalid 222
U2 List
Habib 444
Rashid 555
```

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    u1.addUserIntoList(u1);
    u1.addUserIntoList(u2);
    u2.addUserIntoList(u3);
    u2.addUserIntoList(u4);
    Console.WriteLine("U1 List");
    u1.printList();
    Console.WriteLine("U2 List");
    u2.printList();
    Console.ReadKey();
}
```

|| MUsers: What we actually want?

Here the problem is we want list attribute should be same for all objects and its new copy should not be created in to memory for every new object.

|| MUsers: What we actually want?

When we want to share attribute among all objects, then we use **static** keyword before the attributes and the functions using that attribute.

MUsers: What we actually want?

When we want to share attribute among all objects, then we use **static** keyword before the attributes and the functions using that attribute.

usersList should be static.

Also **addUserIntoList** function should also be static. Because it is manipulating the **static** list.

MUsers



When we want to share attribute among all objects, then we use **static** keyword before the attributes and the functions using that attribute.

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    static List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public static void addUserIntoList(MUser user)
    {
        usersList.Add(user);
    }
}
```



MUsers


Now, how to use
this public static
function.

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    static List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword, string userRole)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = userRole;
    }

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public static void addUserIntoList(MUser user)
    {
        usersList.Add(user);
    }
}
```



MUsers

Now, this looks strange that we are adding the users using different objects although the list is same for all objects.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    u1.addUserIntoList(u1);
    u1.addUserIntoList(u2);
    u2.addUserIntoList(u3);
    u2.addUserIntoList(u4);
}
```

MUsers

Therefore, C# does not let us do this in this manner.

We will get compile time error.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    u1.addUserIntoList(u1);
    u1.addUserIntoList(u2);
    u2.addUserIntoList(u3);
    u2.addUserIntoList(u4);
}
```

MUsers

Instead of calling the public static function with the object, we call it with the class name.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    MUser.addUserIntoList(u1);
    MUser.addUserIntoList(u2);
    MUser.addUserIntoList(u3);
    MUser.addUserIntoList(u4);
}
```

MUsers

List is common
for all objects

```
U1 List
Fatima 123
Khalid 222
Habib 444
Rashid 555
U2 List
Fatima 123
Khalid 222
Habib 444
Rashid 555
```

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    MUser.addUserIntoList(u1);
    MUser.addUserIntoList(u2);
    MUser.addUserIntoList(u3);
    MUser.addUserIntoList(u4);
    Console.WriteLine("U1 List");
    u1.printList();
    Console.WriteLine("U2 List");
    u2.printList();
    Console.ReadKey();
}
```

MUsers


Similarly, we can add **verifyUser** Logic inside the class by making the function **public static**.

```
class MUser
{
    string userName;
    string userPassword;
    string userRole;
    static List<MUser> usersList = new List<MUser>();

    public MUser(string userName, string userPassword)
    {
        this.userName = userName;
        this.userPassword = userPassword;
        this.userRole = "NA";
    }

    public static void addUserIntoList(MUser user)
    {
        usersList.Add(user);
    }

    public static bool isValid(MUser user)
    {
        foreach (MUser storedUser in usersList)
        {
            if (storedUser.userName == user.userName &&
                storedUser.userPassword == user.userPassword)
            {
                return true;
            }
        }
        return false;
    }
}
```



MUsers

Similarly, we can add **verifyUser** Logic inside the class by making the function **public static**.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    MUser.addUserIntoList(u1);
    MUser.addUserIntoList(u2);
    MUser.addUserIntoList(u3);
    MUser.addUserIntoList(u4);

    bool flag = MUser.isValid(u2);

    Console.ReadKey();
}
```

MUsers

Similarly, we can add **verifyUser** Logic inside the class by making the function **public static**.

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    MUser.addUserIntoList(u1);
    MUser.addUserIntoList(u2);
    MUser.addUserIntoList(u3);
    MUser.addUserIntoList(u4);

    bool flag = MUser.isValid(u2);

    Console.WriteLine(flag);

    Console.ReadKey();
}
```

MUsers

Similarly, we can add **verifyUser** Logic inside the class by making the function **public static**.

True

```
static void Main(string[] args)
{
    MUser u1 = new MUser("Fatima", "123", "Admin");
    MUser u2 = new MUser("Khalid", "222", "User");
    MUser u3 = new MUser("Habib", "444", "User");
    MUser u4 = new MUser("Rashid", "555", "User");
    MUser.addUserIntoList(u1);
    MUser.addUserIntoList(u2);
    MUser.addUserIntoList(u3);
    MUser.addUserIntoList(u4);

    bool flag = MUser.isValid(u2);

    Console.WriteLine(flag);

    Console.ReadKey();
}
```


CRC Card: MUser

We have developed this MUser Class.

MUser
<pre>static userList: List userName: String userPassword: String userRole: String</pre>
<pre>MUser(userName: String, userPassword: String, userRole: String) static addUserIntoList(user: MUser): void static IsValid(user: MUser): bool</pre>

Food for Thought

- Why **Main method** is Static?
- Any other Example you have seen during the **coding**?



Conclusion

- Static attributes are common to all objects of the same class.
- Static attributes can only be access by static methods.
- Static attributes and static methods of the class are accessed with the class name in other classes or Driver Class.



Learning Objective

Explain the role of **static** keyword and its use in **Data Layer**.



Self Assessment:

What will be the Output?

```
class Student
```

```
{
```

```
    static public string schoolName = "Govt School";
```

```
    public string studentName;
```

```
}
```

```
static void Main(string[] args) {
```

```
    Student s1 = new Student();
```

```
    s1.studentName = "Rock";
```

```
    // calls instance variable
```

```
    Console.WriteLine("Name: " + s1.studentName);
```

```
    // calls static variable
```

```
    Console.WriteLine("School: " + Student.schoolName);
```

```
    Student s2 = new Student();
```

```
    s2.studentName = "Gal";
```

```
    // calls instance variable
```

```
    Console.WriteLine("Name: " + s2.studentName);
```

```
    // calls static variable
```

```
    Console.WriteLine("School: " + Student.schoolName);
```

```
    Console.ReadLine();
```

```
}
```

Self Assessment:

1. Now make a full Fledge SignIn SignUp Application that will have the **static UsersList** in the Class MUser and all the related functions that apply on the list in the class as well.

NOTE:

Do not forget to make the functions static that are using the static UsersList.

