Independent Data Layer and Separate UI

Revision: MUser

Previously, We have developed the MUser Class.

MUser
static usersList: List userName: String userPassword: String userRole: String
MUser(userName: String, userPassword: String, userRole: String) static addUserIntoList(user: MUser): void static IsValid(user: MUser): bool

Multiple MUsers

Do you see any Problem with this?

MUser
static usersList: List userName: String userPassword: String userRole: String
MUser(userName: String, userPassword: String, userRole: String) static addUserIntoList(user: MUser): void static IsValid(user: MUser): bool

Multiple MUsers

The class is serving 2 purposes.

1. Representation of MUser information in SignIn System.
2. Providing CRUD operation for all user objects.

MUser
static usersList: List userName: String userPassword: String userRole: String
MUser(userName: String, userPassword: String, userRole: String) static addUserIntoList(user: MUser): void static IsValid(user: MUser): bool

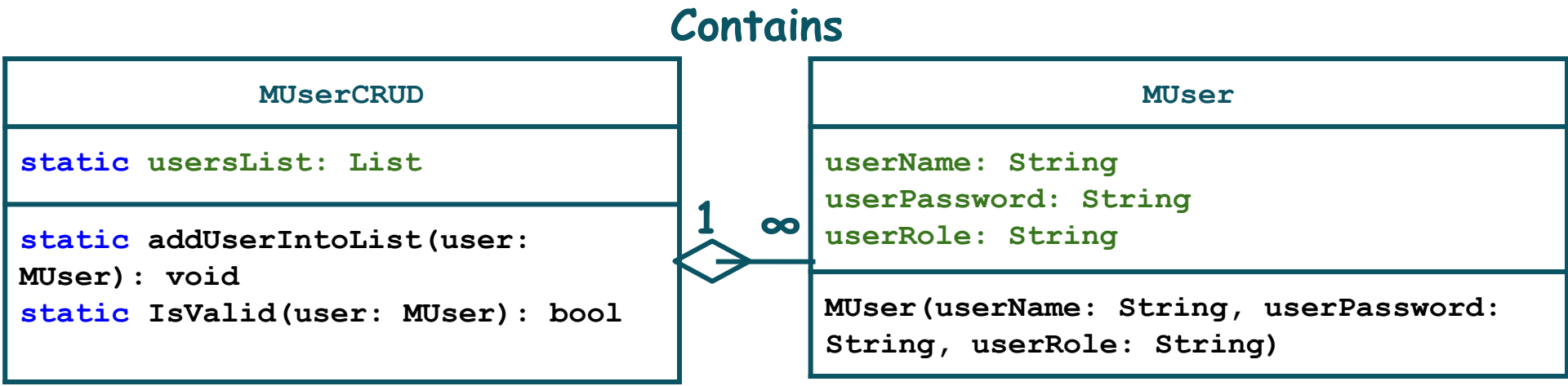
Multiple MUsers

We need to split this class into 2 classes such that MUser only represent the users (Business Logic) and second class should take care of the CRUD operations (Data Layer).

MUser
static usersList: List userName: String userPassword: String userRole: String
MUser(userName: String, userPassword: String, userRole: String) static addUserIntoList(user: MUser): void static IsValid(user: MUser): bool

Multiple MUsers

One Possible model can be



MUsers: Activity

Implement CLI based application that show two menus to user one is for 1) SignIn 2) SignUp. The user interface shall be in main class (program class) and it shall use the MUser for Business Logic and MUserCRUD model for Data Logic.

MUser

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

    public MUser(string userName, string userPassword,
string userRole){
        //Code
    }

    public MUser(string userName, string userPassword){
        //Code
    }
    public string getUserName(){
        return userName;
    }
    public string getUserPassword(){
        return userPassword;
    }

    public string getUserRole(){
        return userRole;
    }
    public bool isAdmin()
    {
        //Code
    }
}
```

MUserCRUD

```
class MUserCRUD
{
    public static List<MUser> usersList = new List<MUser>();

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

    public static MUser SignIn(MUser user)
    {
        // Code
    }

    public static string parseData(string record, int field)
    {
        // Code
    }

    public static void readDataFromFile(string path)
    {
        // Code
    }

    public static void storeUserIntoFile(MUser user, string path)
    {
        // Code
    }
}
```

Driver Program

```
static void Main(string[] args){
    string path = "Data.txt";
    MUserCRUD.readDataFromFile(path);
    int option = 0;
    while (option != 3){
        Console.Clear();
        option = menu();
        if (option == 1){
            MUser user = takeInputwithOutRole();
            user = MUserCRUD.SignIn(user);
            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();
            MUserCRUD.addUserIntoList(user);
            MUserCRUD.storeUserIntoFile(user, path);
        }
        Console.ReadKey();
    }
}
```

Driver Program

```
static int menu()
{
    // Code
    return option;
}

static void printList()
{
    // Code
}

static MUser TakeInputFromConsole()
{
    // Code

    MUser user = new MUser(userName, userPassword,
userRole);
    return user;

}

static MUser takeInputwithOutRole()
{
    // Code
    MUser user = new MUser(userName, userPassword);
    return user;
}
```

MUsers: Activity Updated

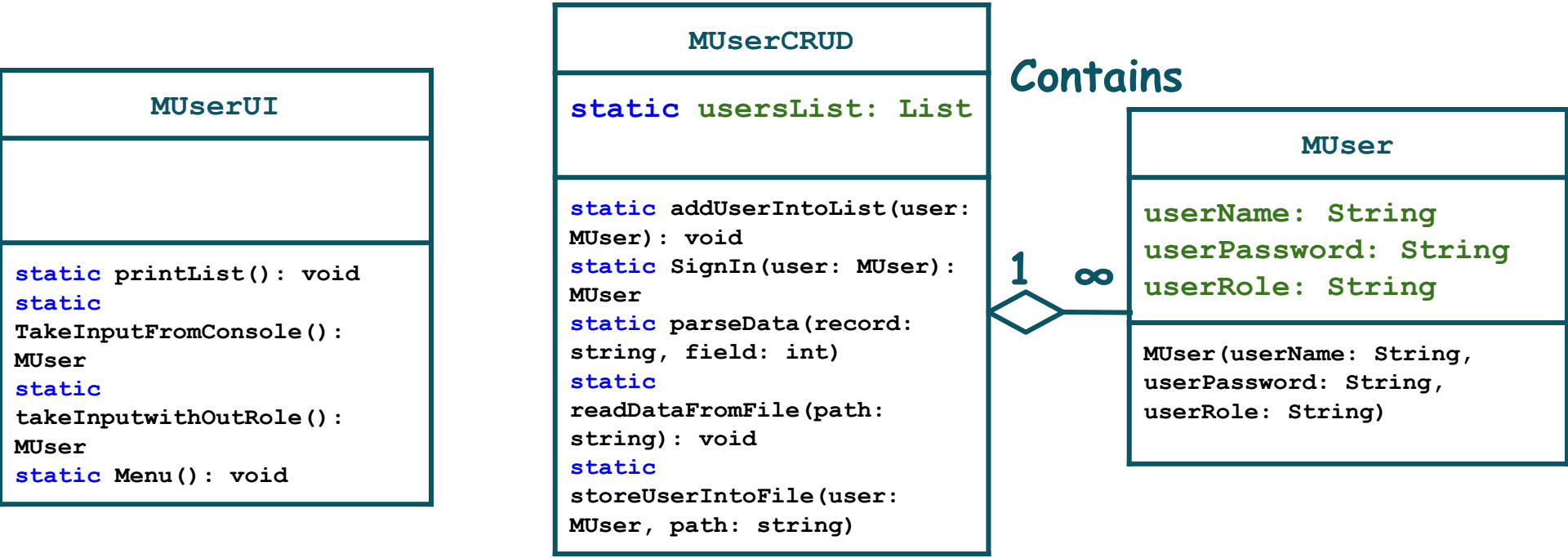
Implement CLI based application that show two menus to user one is for 1) SignIn 2) SignUp.
Now, we will make Three Classes (3 Tier Model)

1. MUser (BL)
2. MUserCRUD (DL)
3. MUserUI (UI)

Now, main program will only use the functions of these classes to implement the Application.

Multiple MUsers

Class Diagram with 3 Tier Model is



MUser

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

    public MUser(string userName, string userPassword,
string userRole){
        //Code
    }

    public MUser(string userName, string userPassword){
        //Code
    }
    public string getUserName(){
        return userName;
    }
    public string getUserPassword(){
        return userPassword;
    }

    public string getUserRole(){
        return userRole;
    }
    public bool isAdmin()
    {
        //Code
    }
}
```

MUserCRUD

```
class MUserCRUD
{
    public static List<MUser> usersList = new List<MUser>();

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

    public static MUser SignIn(MUser user)
    {
        // Code
    }

    public static string parseData(string record, int field)
    {
        // Code
    }

    public static void readDataFromFile(string path)
    {
        // Code
    }

    public static void storeUserIntoFile(MUser user, string path)
    {
        // Code
    }
}
```

MUserUI

```
public static int menu()
{
    //Code
    return option;
}

public static void printList()
{
    //Code
}

public static MUser TakeInputFromConsole()
{
    //Code
    return user;

}

public static MUser takeInputwithOutRole()
{
    //Code
    return user;
}
```

Driver Program

```
static void Main(string[] args){
    string path = "Data.txt";
    MUserCRUD.readDataFromFile(path);
    int option = 0;
    while (option != 3){
        Console.Clear();
        option = MUserUI.menu();
        if (option == 1){
            MUser user = MUserUI.takeInputwithOutRole();
            user = MUserCRUD.SignIn(user);
            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 = MUserUI.TakeInputFromConsole();
            MUserCRUD.addUserIntoList(user);
            MUserCRUD.storeUserIntoFile(user, path);
        }
        Console.ReadKey();
    }
}
```

Learning Objective

Modify the Static Data Layer to Separate Data Layer and Write Separate UI for taking Input and displaying Output

Self Assessment:

1. Implement all the Scenarios with this 3 Tier Model.
2. Also draw the Sequence Diagram for your Scenarios features.