Code with Object Oriented Philosophy

اَللّٰهُمَّ اُرْزُقْنِى عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرُزُقُنِى رِزُقًا وَاسِعًا حَلَالًا طَيِّبًا

مُبَارَكًا مِنُ عِنُدِكَ

Review: SignIn SignUp Application

Make a SignIn and SignUp Application using Class that will check if the user is stored in the file then it will allow it to LogIn. If the user SignUp then the record is stored in the file in comma separated format.

Step 1: Make the class

Make a class.

```
class credential
{
    public string name;
    public string password;
}
```

Step 2: Make the Menu

```
static int menu()
        {
            int option;
            Console.WriteLine("1. SignIn");
            Console.WriteLine("2. SignUp");
            Console.WriteLine("Enter Option");
            option = int.Parse(Console.ReadLine());
            return option;
        }
```

Step 3: Read the data from the file

```
static void readData(string path, List<credential> users)
{
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            credential info = new credential();
            info.name = parseData(record, 1);
            info.password = parseData(record, 2);
            users.Add(info);
        }
        fileVariable.Close();
    }
    else
    {
        Console.WriteLine("Not Exists");
    }
}
```

```
static string parseData(string record, int field)
{
    int comma = 1;
    string item = "";
    for(int x = 0; x < record.Length; x++)
    {
        if (record[x] == ',')
        {
            comma++;
        }
        else if (comma == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

Step 4: Make the SignIn Function

```
static void signIn(string n, string p, List<credential> users)
{
        bool flag = false;
        for (int x = 0; x < users.Count; x++)
        {
            if (n == users[x].name && p == users[x].password)
            {
                 Console.WriteLine("Valid User");
                 flag = true;
                 break;
            }
        }
        if (flag == false)
        {
            Console.WriteLine("Invalid User");
        }
        Console.ReadKey();
}
```

Step 5: Make the SignUp Function

```
static void signUp(string path, string n, string p)
        {
            StreamWriter file = new StreamWriter(path, true);
            file.WriteLine(n + "," + p);
            file.Flush();
            file.Close();
        }
```

Step 6: Main

```
static void Main(string[] args){
     List<credential> users = new List<credential>();
     string path = "G:\\OOP 2022\\BootingCSharp\\textfile.txt";
     int option;
     do{
        readData(path, users);
        Console.Clear();
        option = menu();
        Console.Clear();
        if (option == 1){
            Console.WriteLine("Enter Name: ");
            string n = Console.ReadLine();
            Console.WriteLine("Enter Password: ");
            string p = Console.ReadLine();
            signIn(n, p, users);
        }
        else if (option == 2){
            Console.WriteLine("Enter New Name: ");
            string n = Console.ReadLine();
            Console.WriteLine("Enter New Password: ");
            string p = Console.ReadLine();
            signUp(path, n, p);
         }
     }
     while (option < 3);
     Console.Read();}
```

Review: SignIn SignUp Application

Do you see any problem in this Solution?

Issues: SignIn SignUp Application

1. We are reading the file on every iteration. The time complexity is increased at it takes a lot of time to read the file on every iteration.
2. We are not adding the object in the list during the signUp function, we are directly adding the attributes in the file.
3. There is no separate function to take input from the user.
4. Right now, the functions are not of single responsibility.

Solution: SignIn SignUp Application

Code that resolves the previous issue is as follows.

Class

```
class MUser {
  public string name;
  public string password;
  public string role;
  public MUser(string name, string password) {
    this.name = name;
    this.password = password;
  }

  public MUser(string name, string password, string role) {
    this.name = name;
    this.password = password;
    this.role = role;
  }

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

Menu Function

```
static int menu()
        {
            int option;
            Console.WriteLine("1. SignIn");
            Console.WriteLine("2. SignUp");
            Console.WriteLine("Enter Option");
            option = int.Parse(Console.ReadLine());
            return option;
        }
```

ReadData

```
static string parseData(string
record, int field)
{
    int comma = 1;
    string item = "";
    for(int x = 0; x < record.Length;
x++)
    {
        if (record[x] == ',')
        {
            comma++;
        }
        else if (comma == field)
        {
            item = item + record[x];
        }
    }
    return item;
}
```

```
static bool readData(string path, List<MUser> users)
{
    if (File.Exists(path))
    {
        StreamReader fileVariable = new StreamReader(path);
        string record;
        while ((record = fileVariable.ReadLine()) != null)
        {
            string name = parseData(record, 1);
            string password = parseData(record, 2);
            string role = parseData(record, 3);
            MUser user = new MUser(name, password, role);
            storeDataInList(users, user);
        }
        fileVariable.Close();
        return true;
    }
        return false;
}
```

For SignIn: TakeInputWithoutRole

```
static MUser takeInputWithoutRole()
{
    Console.WriteLine("Enter Name: ");
    string name = Console.ReadLine();
    Console.WriteLine("Enter Password: ");
    string password = Console.ReadLine();
    if (name != null && password != null)
    {
        MUser user = new MUser(name, password);
        return user;
    }
    return null;

}
```

For SignUp: TakeInputWithRole

```
static MUser takeInputWithRole()
{
    Console.WriteLine("Enter Name: ");
    string name = Console.ReadLine();
    Console.WriteLine("Enter Password: ");
    string password = Console.ReadLine();
    Console.WriteLine("Enter Role: ");
    string role = Console.ReadLine();
    if (name != null && password != null && role != null)
    {
        MUser user = new MUser(name, password, role);
        return user;
    }
    return null;
}
```

For SignUp: StoreDataInList

```
static void storeDataInList(List<MUser> users, MUser user)
{
    users.Add(user);
}
```

For SignUp: StoreDataInFile

```
static void storeDataInFile(string path, MUser user)
{
    StreamWriter file = new StreamWriter(path, true);
    file.WriteLine(user.name + "," + user.password + "," + user.role);
    file.Flush();
    file.Close();
}
```

For SignIn: Validate User

```
static MUser signIn(MUser user, List<MUser> users)
{
    foreach (MUser storedUser in users)
    {
        if (user.name == storedUser.name && user.password == storedUser.password)
        {
            return storedUser;
        }
    }
    return null;
}
```

Main Function

```
static void Main(string[] args){
    List<MUser> users = new List<MUser>();
    string path = "textfile.txt";
    int option;
    bool check = readData(path, users);
    if (check)
        Console.WriteLine("Data Loaded SuccessFully");
    else
        Console.WriteLine("Data Not Loaded");
    Console.ReadKey();
    do{
        Console.Clear();
        option = menu();
        Console.Clear();
        if (option == 1){
            MUser user = takeInputWithoutRole();
            if (user != null){
                user = signIn(user, users);
                if (user == null)
                    Console.WriteLine("Invalid User");
                else if (user.isAdmin())
                    Console.WriteLine("Admin Menu");
                else
                    Console.WriteLine("User Menu");
            }
        }
        else if (option == 2){
            MUser user = takeInputWithRole();
            if (user != null){
                storeDataInFile(path, user);
                storeDataInList(users, user);
            }
        }
        Console.ReadKey();
    }
    while (option < 3);
}
```

Learning Objective

Write class with appropriate behaviour on the data and separate functions with single responsibility.