

# EUPHORIA

[Arul](#)

[Asuvath](#)

[Gokulraj](#)

[Gokul](#)

[Iswarya](#)

[Kavinkumar](#)

[Kirubaharan](#)

[Prathima](#)

[Ragunath](#)

[Sruthi](#)

[Sobhana](#)

# SOLID Principles



*attention.  
always.*



# SOLID Principles

- S - Single-responsibility Principle
- O - Open-closed Principle
- L - Liskov Substitution Principle
- I - Interface Segregation Principle
- D - Dependency Inversion Principle

*attention.  
always.*



# S - Single-responsibility Principle

```
namespace ConsoleUI
{
    class Program
    {
        static void Main(string[] args)
        {
            StandardMessages.WelcomeMessage();

            Person user = PersonDataCapture.Capture();

            bool isValid = PersonValidator.Validate(user);

            if (!isValid)
            {
                StandardMessages.EndApplication();
                return;
            }

            AccountGenerator.CreateAccount(user);

            StandardMessages.EndApplication();
        }
    }
}
```

```
namespace ConsoleUI
{
    public class PersonDataCapture
    {
        public static Person Capture()
        {
            // Ask for user information
            Person output = new Person();

            Console.WriteLine("What is your first name: ");
            output.FirstName = Console.ReadLine();

            Console.WriteLine("What is your last name: ");
            output.LastName = Console.ReadLine();

            return output;
        }
    }
}
```

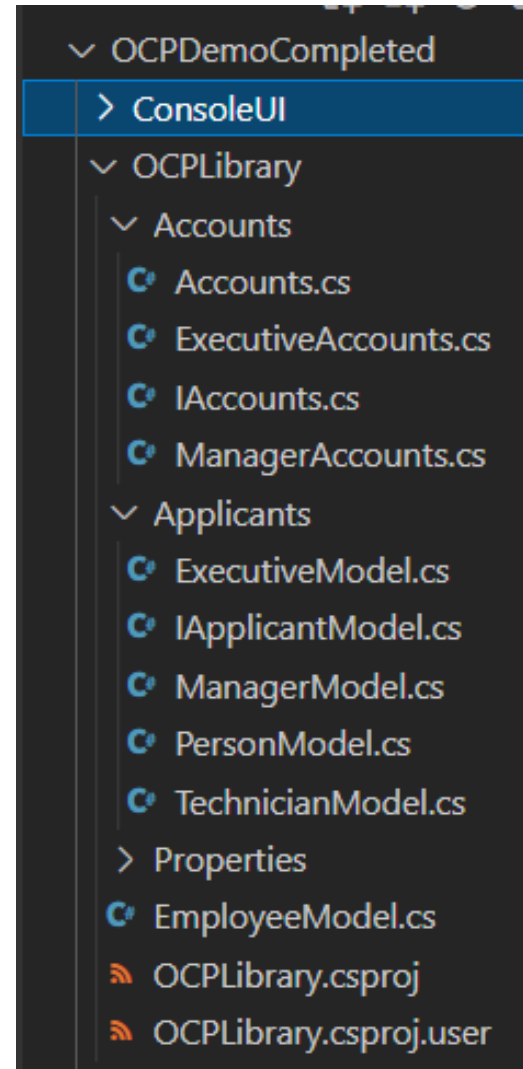
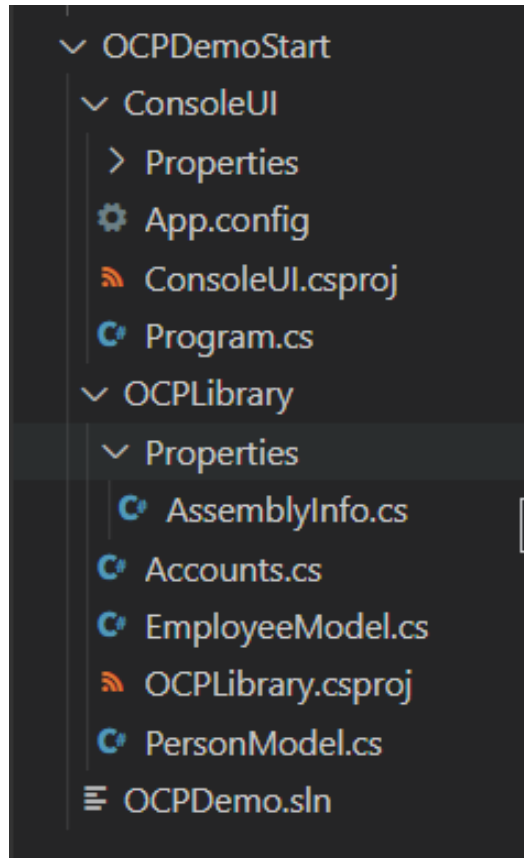
```
namespace ConsoleUI
{
    public class PersonValidator
    {
        public static bool Validate(Person person)
        {
            // Checks to be sure the first and last names are valid
            if (string.IsNullOrEmpty(person.FirstName))
            {
                StandardMessages.DisplayValidationError("first name");
                return false;
            }

            if (string.IsNullOrEmpty(person.LastName))
            {
                StandardMessages.DisplayValidationError("last name");
                return false;
            }

            return true;
        }
    }
}
```

*attention.*  
*always.*

# O - Open-closed Principle



*attention.  
always.*

# L - Liskov Substitution Principle

```
namespace DemoLibrary
{
    public interface IEmployee
    {
        string FirstName { get; set; }
        string LastName { get; set; }
        decimal Salary { get; set; }

        void CalculatePerHourRate(int rank);
    }
}
```

```
namespace DemoLibrary
{
    public interface IManager : IEmployee
    {
        void GeneratePerformanceReview();
    }
}
```

```
namespace DemoLibrary
{
    public interface IManaged : IEmployee
    {
        IEmployee Manager { get; set; }
        void AssignManager(IEmployee manager);
    }
}
```

```
namespace DemoLibrary
{
    public class Manager : Employee, IManager
    {
        public override void CalculatePerHourRate(int rank)
        {
            decimal baseAmount = 19.75M;

            Salary = baseAmount + (rank * 4);
        }

        public void GeneratePerformanceReview()
        {
            // Simulate reviewing a direct report
            Console.WriteLine("I'm reviewing a direct report's performance.");
        }
    }
}
```

```
namespace DemoLibrary
{
    public class Manager : Employee, IManager
    {
        public override void CalculatePerHourRate(int rank)
        {
            decimal baseAmount = 19.75M;

            Salary = baseAmount + (rank * 4);
        }

        public void GeneratePerformanceReview()
        {
            // Simulate reviewing a direct report
            Console.WriteLine("I'm reviewing a direct report's performance.");
        }
    }
}
```

*attention.  
always.*

# I - Interface Segregation Principle

```
namespace DemoLibrary
{
    public interface ILibraryItem
    {
        string Author { get; set; }
        DateTime BorrowDate { get; set; }
        string Borrower { get; set; }
        int CheckOutDurationInDays { get; set; }
        string LibraryId { get; set; }
        int Pages { get; set; }
        string Title { get; set; }

        void CheckIn();
        void CheckOut(string borrower);
        DateTime GetDueDate();
    }
}
```

```
namespace DemoLibrary
{
    public interface ILibraryItem
    {
        string LibraryId { get; set; }
        string Title { get; set; }
    }
}
```

```
namespace DemoLibrary
{
    public interface IBorrowable
    {
        DateTime BorrowDate { get; set; }
        string Borrower { get; set; }
        int CheckOutDurationInDays { get; set; }

        void CheckIn();
        void CheckOut(string borrower);
        DateTime GetDueDate();
    }
}
```

*attention.*  
*always.*



# D - Dependency Inversion Principle

```
namespace ConsoleUI
{
    class Program
    {
        static void Main(string[] args)
        {
            Person owner = new Person
            {
                FirstName = "Tim",
                LastName = "Corey",
                EmailAddress = "tim@iamtimcorey.com",
                PhoneNumber = "555-1212"
            };

            Chore chore = new Chore
            {
                ChoreName = "Take out the trash",
                Owner = owner
            };

            chore.PerformedWork(3);
            chore.PerformedWork(1.5);
            chore.CompleteChore();

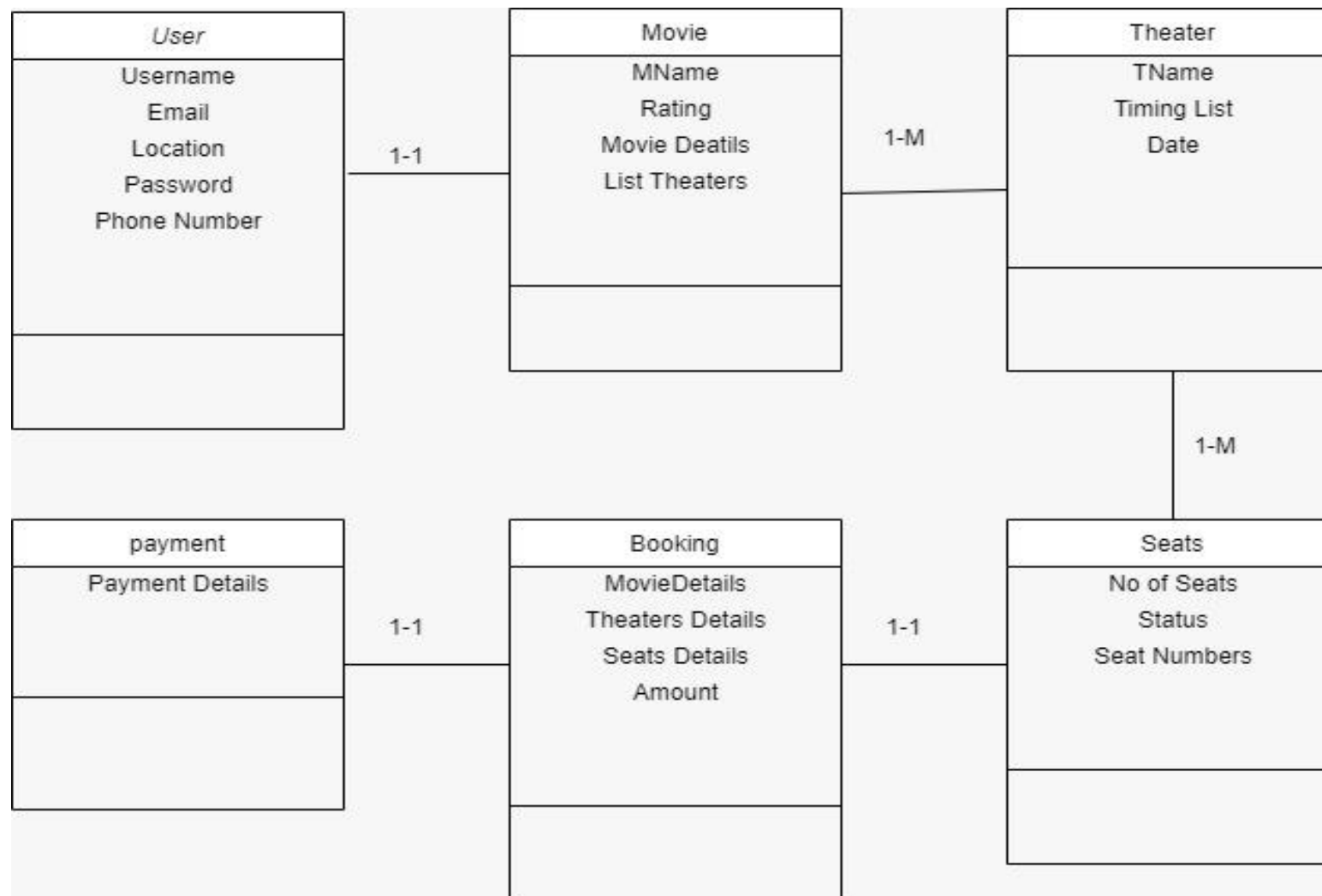
            Console.ReadLine();
        }
    }
}
```

```
8 namespace ConsoleUI
9 {
    0 references
    10 class Program
    11 {
        0 references
        12 static void Main(string[] args)
        13 {
            14     IPerson owner = Factory.CreatePerson();
            15
            16     owner.FirstName = "Tim";
            17     owner.LastName = "Corey";
            18     owner.EmailAddress = "tim@iamtimcorey.com";
            19     owner.PhoneNumber = "555-1212";
            20
            21     IChore chore = Factory.CreateChore();
            22     chore.ChoreName = "Take out the trash";
            23     chore.Owner = owner;
            24
            25     chore.PerformedWork(3);
            26     chore.PerformedWork(1.5);
            27     chore.CompleteChore();
            28
            29     Console.ReadLine();
            30 }
        31 }
    32 }
    33 }
```

```
8 namespace ConsoleUI
9 {
    10 public static class Factory
    11 {
        12     public static IPerson CreatePerson()
        13     {
            14         return new Person();
        15     }
        16
        17     public static IChore CreateChore()
        18     {
            19         return new Chore(CreateLogger(),
            20             CreateMessageSender());
        21     }
        22
        23     public static ILogger CreateLogger()
        24     {
            25         return new Logger();
        26     }
        27
        28     public static IMessageSender CreateMessageSender()
        29     {
            30         return new Texter();
        31     }
    32 }
    33 }
```

*a t t e n t i o n .*  
*a l w a y s .*

# Model Design



*attention.  
always.*



Thank you  
for your time



*attention.  
always.*

*Thank you.*

