EUPHORIA

Arul
Asuvath
Gokulraj
Gokul
Iswarya
Kavinkumar
Kirubaharan
Prathima
Ragunath
Sruthi

SOLID Principles

aspire SYSTEMS

attention.
always.

SOLID Principles

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

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

S - Single-responsibility Principle

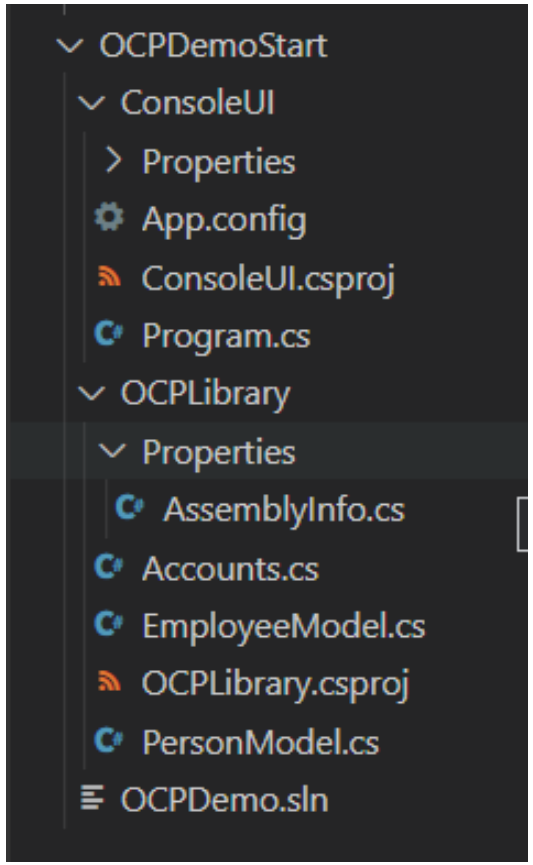
```
namespace ConsoleUI
{
    class Program
    {
        static void Main(string[] args)
        {
            StandardMessages.WelcomeMessage();

            Person user = PersonDataCapture.Capture();

            bool isUserValid = PersonValidator.Validate(user);

            if (isUserValid == false)
            {
                StandardMessages.EndApplication();
                return;
            }

            AccountGenerator.CreateAccount(user);

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

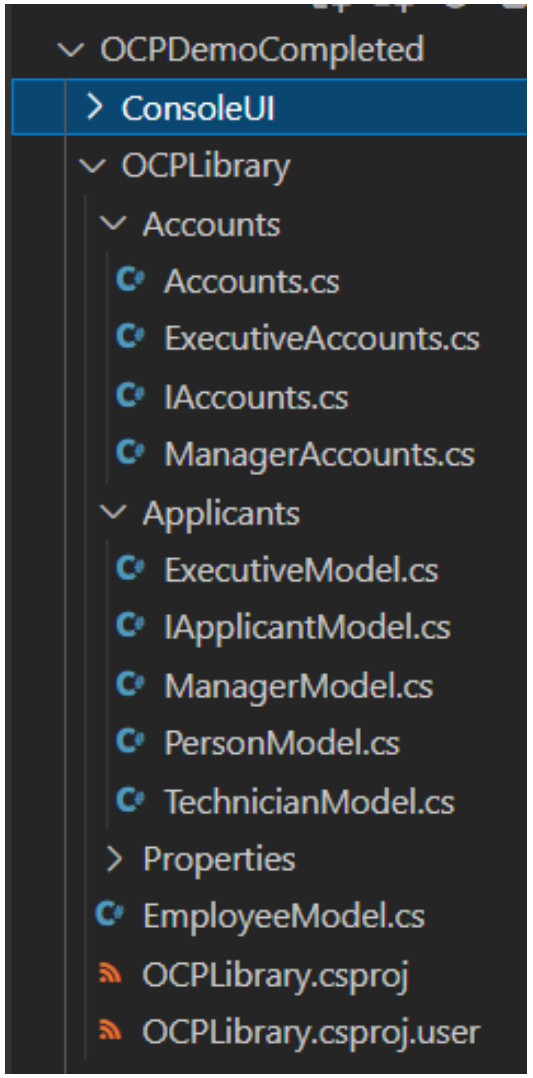
```
namespace ConsoleUI
{
    public class PersonDataCapture
    {
        public static Person Capture()
        {
            // Ask for user information
            Person output = new Person();

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

            Console.Write("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.IsNullOrWhiteSpace(person.FirstName))
            {
                StandardMessages.DisplayValidationError("first name");
                return false;
            }

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

            return true;
        }
    }
}
```

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

O - Open-closed Principle





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

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.");
        }
    }
}
```

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

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();
    }
}
```

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

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();
        }
    }
}
```

```
namespace ConsoleUI
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            IPerson owner = Factory.CreatePerson();

            owner.FirstName = "Tim";
            owner.LastName = "Corey";
            owner.EmailAddress = "tim@iamtimcorey.com";
            owner.PhoneNumber = "555-1212";

            IChore chore = Factory.CreateChore();
            chore.ChoreName = "Take out the trash";
            chore.Owner = owner;

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

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

```
namespace ConsoleUI
{
    public static class Factory
    {
        public static IPerson CreatePerson()
        {
            return new Person();
        }

        public static IChore CreateChore()
        {
            return new Chore(CreateLogger(),
            CreateMessageSender());
        }

        public static ILogger CreateLogger()
        {
            return new Logger();
        }

        public static IMessageSender CreateMessageSender()
        {
            return new Texter();
        }
    }
}
```

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

Thank you
for your time

aspire SYSTEMS | attention. always.