Section: Master 1 SIGL

Faculty of Mathematics and Computer Science
Department of Computer Science

Object Analysis and Design (OAD)
Lab Session (TP) 03

Section 1. Structural Patterns

1. **Vehicle Proxy:**
   We have an interface *ICar* that defines the function *MoveCar* and we have a class *Driver* (age). We want to use Proxy pattern to verify if the driver is permitted to move a car. The function MoveCar is responsible for verifying if the driver's age must be greater or equal to 18.

   Using the proxy pattern, give a class diagram and a code implementation of this problem.

```
Driver's Age is 17: Sorry the driver is too young to drive
Car has been driven
```

Figure 1: Proxy example output

2. **SmartTv Bridge:**
   We have an interface *IVideoSource* that defines the function *PlayVideo* and we have a class *MySuperSmartTv* that has an attribute of type *IVideoSource* and a function *PlayTv*. We want to use Bridge pattern to make sure that MySuperSmartTv plays a video with respect to the video source defined by the user. We have three types of video sources : three classes implement *IVideoSource* (*LocalCabelTv*, *LocalDishTv* and *IpTvService*).

   Using the Bridge pattern, give a class diagram and a code implementation of this problem.

```
Select A source to get TV Guide and Play
1. Local Cable TV
2. Local Dish TV
3. IP TV
2
Playing - Local DISH TV
```

Figure 2: Bridge example output

3. **Vehicle Decorator:**
   We have an interface *Vehicle* that defines 3 functions (*Make*, *Model*, *Price*). BMW (make) is a vehicle that has a model X6 (model) with price 75000 (price). We want to use Decorator pattern to make a special offer of this vehicle with a discount of 25% .

   Using the decorator pattern, give a class diagram and a code implementation of this problem.

```
BMW X6 base price are : 7500000
With special offer (25 % discount) the price is : 5625000
```

Figure 3: Decorator example output

Answers

Section 1. Structural Patterns

1. **Vehicle Proxy:**
   We have an interface *ICar* that defines the function *MoveCar* and we have a class *Driver* (age). We want to use Proxy pattern to verify if the driver is permitted to move a car. The function MoveCar is responsible for verifying if the driver's age must be greater or equal to 18.

   Using the proxy pattern, give a class diagram and a code implementation of this problem.

```
Driver's Age is 17: Sorry the driver is too young to drive
Car has been driven
```

Figure 4: Proxy example output

Answer:

2. **SmartTv Bridge:**
   We have an interface *IVideoSource* that defines the function *PlayVideo* and we have a class *MySuperSmartTv* that has an attribute of type *IVideoSource* and a function *PlayTv*. We want to use Bridge pattern to make sure that MySuperSmartTv plays a video with respect to the video source defined by the user. We have three types of video sources : three classes implement *IVideoSource* (*LocalCabelTv*, *LocalDishTv* and *IpTvService*).

   Using the Bridge pattern, give a class diagram and a code implementation of this problem.

   Answer:

3. **Vehicle Decorator:**
   We have an interface *Vehicle* that defines 3 functions (*Make*, *Model*, *Price*). BMW (make) is a vehicle that has a model X6 (model) with price 75000 (price). We want to use Decorator pattern to make a special offer of this vehicle with a discount of 25% .

   Using the decorator pattern, give a class diagram and a code implementation of this problem.

   Answer:

```
interface ICar
{
    void MoveCar();
}

class ProxyCar : ICar
{
    private Driver driver;
    private ICar _realCar;

    public ProxyCar(Driver driver)
    {
        this.driver = driver;
        _realCar = new Car();
    }

    void ICar.MoveCar()
    {
        if (driver.Age < 18)
            Console.WriteLine("Driver's Age is " + driver.Age.ToString() + ": "  + "Sorry the driver is too young to drive")
        else
          _realCar.MoveCar();
    }
}
```

```
//the real object
class Car : ICar
{
    void ICar.MoveCar()
    {
        Console.WriteLine("Car has been driven");
    }
}

class Driver
{
    public int Age { get; set; }

    public Driver(int age)
    {
        this.Age = age;
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        ICar car = new ProxyCar(new Driver(17));
        car.MoveCar();

        car = new ProxyCar(new Driver(20));
        car.MoveCar();

        Console.ReadKey();
    }
}
```

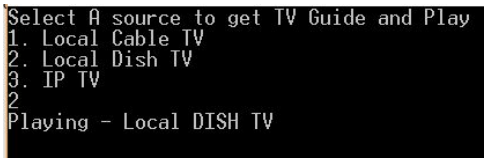

Figure 5: Bridge example output

```
public interface IVideoSource
{
    string PlayVideo();
}

class LocalCabelTv : IVideoSource
{
    private const string SourceName = "Local Cabel TV";

    public string PlayVideo()
    {
        return string.Format("Playing - {0}", SourceName);
    }
}

class LocalDishTv : IVideoSource
{
    private const string SourceName = "Local DISH TV";

    public string PlayVideo()
    {
        return string.Format("Playing - {0}", SourceName);
    }
}
```

```
class MySuperSmartTv
{
    IVideoSource _currentVideoSource;

    public IVideoSource VideoSource
    {
        get
        {
            return _currentVideoSource;
        }
        set
        {
            _currentVideoSource = value;
        }
    }

    public void PlayTv()
    {
        if (_currentVideoSource != null)
        {
            Console.WriteLine(_currentVideoSource.PlayVideo());
        }
        else
        {
            Console.WriteLine("Please select a Video Source to play");
        }
    }
}


 static void Main(string[] args)
 {
     MySuperSmartTv myTv = new MySuperSmartTv();

     Console.WriteLine("Select A source to get TV Guide and Play");
     Console.WriteLine("1. Local Cable TV\n2. Local Dish TV\n3. IP TV");

     ConsoleKeyInfo input = Console.ReadKey();

     // Let us see what user has selected and select the video source accrodingly
     switch (input.KeyChar)
     {
         case '1':
             myTv.VideoSource = new LocalCabelTv();
             break;

         case '2':
             myTv.VideoSource = new LocalDishTv();
             break;

     }

     Console.WriteLine();
     //Let us now play the selected TV source.
     myTv.PlayTv();

     Console.WriteLine(); //some whitespace on output for readability
     Console.ReadKey();
 }
}
```

```
BMW X6 base price are : 7500000
With special offer (25 % discount) the price is : 5625000
```

Figure 6: Decorator example output

```
public interface IVehicle
{
    string Make { get; }
    string Model { get; }
    double Price { get; }
}

/// <summary>
/// The 'ConcreteComponent' class
/// </summary>
public class Bmw : IVehicle
{
    public string Make
    {
        get { return "BMW"; }
    }

    public string Model
    {
        get { return "X6"; }
    }

    public double Price
    {
        get { return 7500000; }
    }
}
```

```
public class SpecialOfferDecorator : IVehicle
{
    private IVehicle _vehicle;

    public SpecialOfferDecorator(IVehicle vehicle)
    {
        this._vehicle = vehicle;
    }
    public int DiscountPercentage { get; set; }
    public string Offer { get; set; }

    public string Make
    {
        get { return _vehicle.Make; }
    }
    public string Model
    {
        get { return _vehicle.Model; }
    }
    public double Price
    {
        get
        {
            double price = _vehicle.Price;
            int percentage = 100 - DiscountPercentage;
            return Math.Round((price * percentage) / 100, 2);
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        // Basic vehicle
        Bmw car = new Bmw();
        // basic price
        Console.WriteLine(car.Make + " " + car.Model + " base price is : {0}", car.Price);

        // Special offer (Decorated)
        SpecialOfferDecorator offer = new SpecialOfferDecorator(car);
        offer.DiscountPercentage = 25;
        offer.Offer = "25 % discount";

        Console.WriteLine("With special offer ({1}) the price is : {0} ", offer.Price, offer.Offer);

        Console.ReadKey();
    }
}
```