Laboratory 1

Creational Design Pattern

Domain: Car Factory Shop

There was implemented 3 creational DPs:

- Builder- Separates object construction from its representation
- Singleton- A class of which only a single instance can exist
- Prototype- A fully initialized instance to be copied or cloned

1. Builder-Object Construction

```
            Factory facory = new Factory();

            builder = new ScooterBuilder();
            facory.Construct(builder);
            Singleton.Singleton.Instance.AddModel(builder);

            builder = new CarBuilder();
            facory.Construct(builder);
            Singleton.Singleton.Instance.AddModel(builder);

            builder = new MotorCycleBuilder();
            facory.Construct(builder);
            Singleton.Singleton.Instance.AddModel(builder);

            string[] list = Singleton.Singleton.Instance.InitList();
            for (int i = 0; i < list.Length; i++)
            {
                listBox1.Items.Add(list[i].ToString());
            }
```

Objects representation:

```
class MotorCycleBuilder : ProxyVehicle
    {
        public MotorCycleBuilder()
        {
            vehicle = new Vehicle("MotorCycle");
        }
        public override void BuildFrame()
        {
            vehicle["frame"] = "MotorCycle Frame";
        }
        public override void BuildEngine()
        {
            vehicle["engine"] = "500 cc";
        }
        public override void BuildWheels()
        {
            vehicle["wheels"] = "2";
        }
        public override void BuildDoors()
        {
            vehicle["doors"] = "0";
        }

    }
```

2. Singleton- a singleton class that create and access vechicles list (car/scooter/motorcycle)

```
  public static Singleton Instance
        {
            get
            {
                lock (testlock)
                    {
                        if (instance == null)
                        {
                            instance = new Singleton();
                        }
                        return instance;
                    }
            }
        }

private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            Singleton.Singleton.Instance.Clear();
            VehicleBuilder builder;
            // Create shop with vehicle builders
            Factory facory = new Factory();

            builder = new ScooterBuilder();
            facory.Construct(builder);
            Singleton.Singleton.Instance.AddModel(builder);

            builder = new CarBuilder();
            facory.Construct(builder);
            Singleton.Singleton.Instance.AddModel(builder);

            builder = new MotorCycleBuilder();
            facory.Construct(builder);
            Singleton.Singleton.Instance.AddModel(builder);

            string[] list = Singleton.Singleton.Instance.InitList();
            for (int i = 0; i < list.Length; i++)
            {
                listBox1.Items.Add(list[i].ToString());
            }

        }
```

3. **Prototype**- specifies the objects to create using a prototypical instance, and create new objects by copying this prototype.

```
abstract class VehicleBuilder
{
    public Vehicle vehicle;
    // Gets vehicle instance
    public Vehicle Vehicle
    {
        get { return vehicle; }
    }
    // Abstract build methods
    public abstract void BuildFrame();
    public abstract void BuildEngine();
    public abstract void BuildWheels();
    public abstract void BuildDoors();

    public abstract VehicleBuilder Clone();
}
```