**Pattern Name:**

Builder Pattern

**Short Description:**

Separate representation and object construction

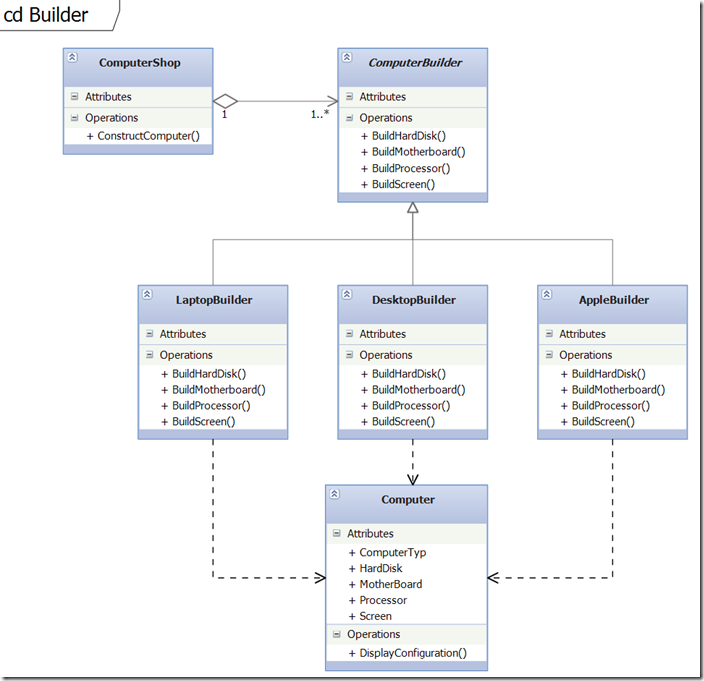
**Usage:**

Rarely used, only useful if complex objects consisting of multiple parts need to be constructed (composite objects for example)

**Complexity:**

1 / 5

**UML Class Diagram:**

[](http://download.codeplex.com/download?ProjectName=csharpdesignpatterns&DownloadId=236765)

**Explanation:**

* The director (ComputerShop) implements a method that is responsible for the sequence of steps of an object creation process. It takes an abstract builder class as input parameter and delegates the real creation to it.
* The abstract builder class defines the interface that all inheriting concrete builders will use for object creation.

    public class ComputerShop  
    {  
        public void ConstructComputer(ComputerBuilder computerBuilder)  
        {  
            computerBuilder.BuildMotherboard();  
            computerBuilder.BuildProcessor();  
            computerBuilder.BuildHardDisk();  
            computerBuilder.BuildScreen();  
        }  
    }  
   
    public abstract class ComputerBuilder  
    {  
        public Computer Computer { get; set; }  
   
        public abstract void BuildMotherboard();  
        public abstract void BuildProcessor();  
        public abstract void BuildHardDisk();  
        public abstract void BuildScreen();  
    }

* The concrete builder implementations contain the parts that are assembled and that build the objects.

    public class LaptopBuilder : ComputerBuilder  
    {  
        public LaptopBuilder()  
        {  
            Computer = new Computer(ComputerTyp.Laptop);  
        }  
   
        public override void BuildMotherboard()  
        {  
            Computer.MotherBoard = "DELL MotherBoard";  
        }  
   
        public override void BuildProcessor()  
        {  
            Computer.Processor = "Intel Core 2 Duo";  
        }  
   
        public override void BuildHardDisk()  
        {  
            Computer.HardDisk = "250GB";  
        }  
   
        public override void BuildScreen()  
        {  
            Computer.Screen = "15.4-inch (1280 x 800)";  
        }  
    }  
   
    public class DesktopBuilder : ComputerBuilder  
    {  
        public DesktopBuilder()  
        {  
            Computer = new Computer(ComputerTyp.Desktop);  
        }  
   
        public override void BuildMotherboard()  
        {  
            Computer.MotherBoard = "Asus P6X58D Premium";  
        }  
   
        public override void BuildProcessor()  
        {  
            Computer.Processor = "Intel Xeon 7500";  
        }  
   
        public override void BuildHardDisk()  
        {  
            Computer.HardDisk = "2TB";  
        }  
   
        public override void BuildScreen()  
        {  
            Computer.Screen = "21 inch (1980 x 1200)";  
        }  
    }  
   
    public class AppleBuilder : ComputerBuilder  
    {  
        public AppleBuilder()  
        {  
            Computer = new Computer(ComputerTyp.Apple);  
        }  
   
        public override void BuildMotherboard()  
        {  
            Computer.MotherBoard = "iMac G5 PowerPC";  
        }  
   
        public override void BuildProcessor()  
        {  
            Computer.Processor = "Intel Core 2 Duo";  
        }  
   
        public override void BuildHardDisk()  
        {  
            Computer.HardDisk = "320GB";  
        }  
   
        public override void BuildScreen()  
        {  
            Computer.Screen = "24 inch (1980 x 1200)";  
        }  
    }

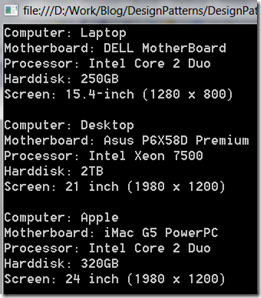
* The final object contains all different parts that get assembled by the concrete builder classes. Those may differ from each other depending on the implementations.
* A method was added that prints out the characteristics of the different parts to be able to validate the design.

    public class Computer  
    {  
        private ComputerTyp \_computerTyp;  
   
        public string MotherBoard { get; set; }  
        public string Processor { get; set; }  
        public string HardDisk { get; set; }  
        public string Screen { get; set; }  
   
        public Computer(ComputerTyp computerTyp)  
        {  
            \_computerTyp = computerTyp;  
        }  
   
        public void DisplayConfiguration()  
        {  
            string message;  
   
            message = string.Format("Computer: {0}", \_computerTyp);  
            Console.WriteLine(message);  
   
            message = string.Format("Motherboard: {0}", MotherBoard);  
            Console.WriteLine(message);  
   
            message = string.Format("Processor: {0}", Processor);  
            Console.WriteLine(message);  
   
            message = string.Format("Harddisk: {0}", HardDisk);  
            Console.WriteLine(message);  
   
            message = string.Format("Screen: {0}", Screen);  
            Console.WriteLine(message);  
   
            Console.WriteLine();  
        }  
    }  
   
    public enum ComputerTyp  
    {  
        Laptop,  
        Desktop,  
        Apple  
    }

* In the last step we add some code to test the software design and the Builder implementation.

    private static void Builder()  
    {  
        ComputerShop computerShop = new ComputerShop();  
        ComputerBuilder computerBuilder;  
   
        computerBuilder = new LaptopBuilder();  
        computerShop.ConstructComputer(computerBuilder);  
        computerBuilder.Computer.DisplayConfiguration();  
   
        computerBuilder = new DesktopBuilder();  
        computerShop.ConstructComputer(computerBuilder);  
        computerBuilder.Computer.DisplayConfiguration();  
   
        computerBuilder = new AppleBuilder();  
        computerShop.ConstructComputer(computerBuilder);  
        computerBuilder.Computer.DisplayConfiguration();  
        Console.ReadKey();  
    }

* When running the example you can see that everything is working as expected and that the correct classes are instantiated during runtime.

[](http://download.codeplex.com/download?ProjectName=csharpdesignpatterns&DownloadId=236278)