Abstract Factory

```
class Program
{
    public static void Main()
    {
        Client client = null;

        client = new Client(new Factory1());
        client.Run();

        client = new Client(new Factory2());
        client.Run();
    }
}
```

```
class Client
{
    AbstractProductA apa;
    AbstractProductB apb;

    public Client(AbstractFactory af)
    {
        apa = af.CreateProductA();
        apb = af.CreateProductB();
    }

    public void Run()
    {
        apb.Interact(apa);
    }
}
```

```
abstract class AbstractFactory
{
    public abstract AbstractProductA CreateProductA();
    public abstract AbstractProductB CreateProductB();
}
```

```
class Factory1 : AbstractFactory
{
    public override AbstractProductA CreateProductA()
    {
        return new ProductA1();
    }

    public override AbstractProductB CreateProductB()
    {
        return new ProductB1();
    }
}
```

```
class Factory2 : AbstractFactory
{
    public override AbstractProductA CreateProductA()
    {
        return new ProductA2();
    }

    public override AbstractProductB CreateProductB()
    {
        return new ProductB2();
    }
}
```

```
abstract class AbstractProductA
{
}
```

```
class ProductA1 : AbstractProductA
{
}
```

```
class ProductA2 : AbstractProductA
{
}
```

```
abstract class AbstractProductB
{
    public abstract void Interact(AbstractProductA apa);
}
```

```
class ProductB1 : AbstractProductB
{
    public override void Interact(AbstractProductA apa)
    {
        Console.WriteLine(this + " interacts with " + apa);
    }
}
```

```
class ProductB2 : AbstractProductB
{
     public override void Interact(AbstractProductA apa)
    {
        Console.WriteLine(this + " interacts with " + apa);
    }
}
```

Builder

```
class Program
{
    public static void Main()
    {
        Builder builder = new ConcreteBuilder();
        Director director = new Director(builder);
        director.Construct();
        Product product = builder.GetResult();
        product.Show();
    }
}
```

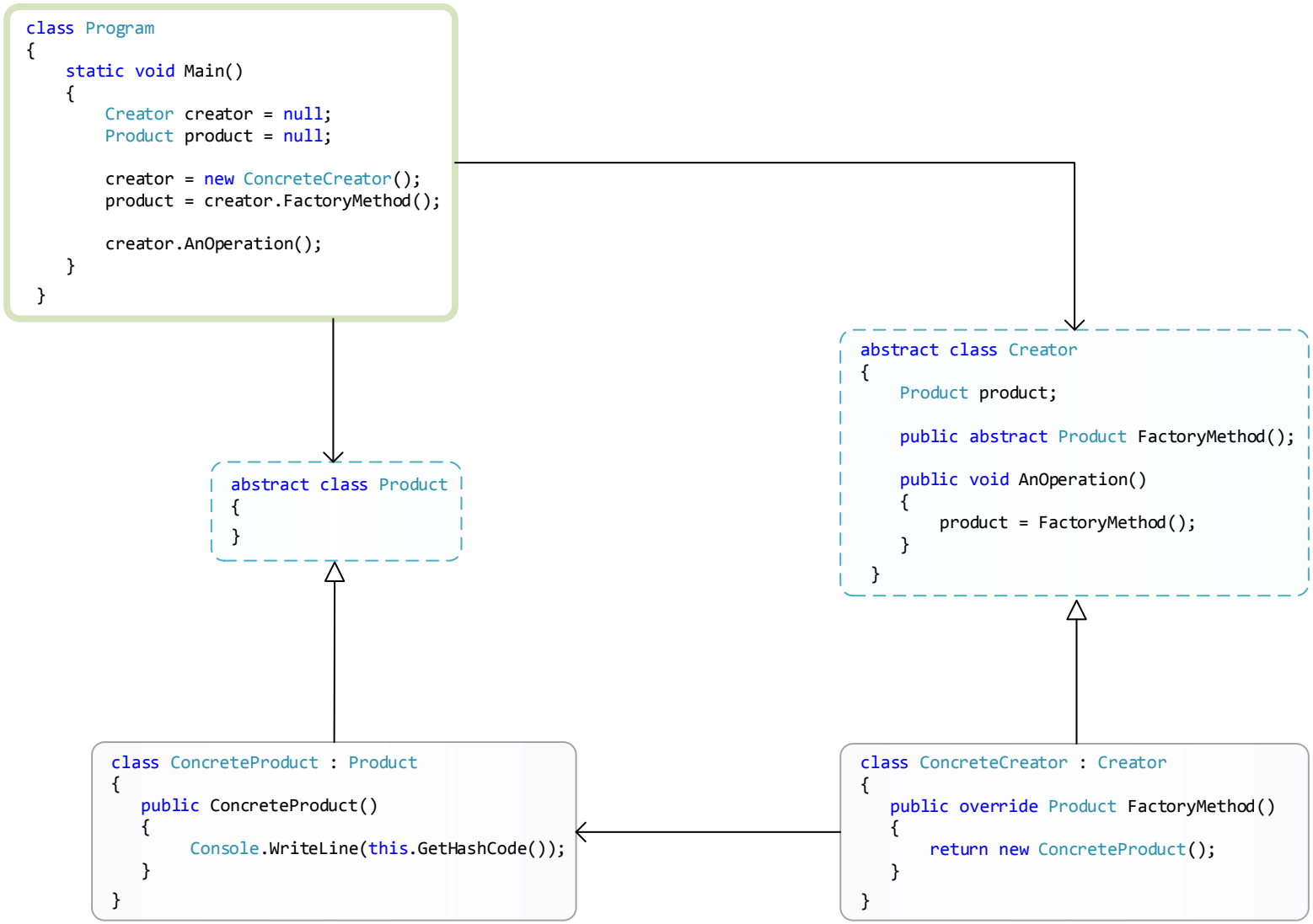
```
class Director
{
    Builder builder;

    public Director(Builder builder)
    {
        this.builder = builder;
    }

    public void Construct()
    {
        builder.BuildPartA();
        builder.BuildPartB();
        builder.BuildPartC();
    }
}
```

```
abstract class Builder
{
    public abstract void BuildPartA();
    public abstract void BuildPartB();
    public abstract void BuildPartC();
    public abstract Product GetResult();
}
```

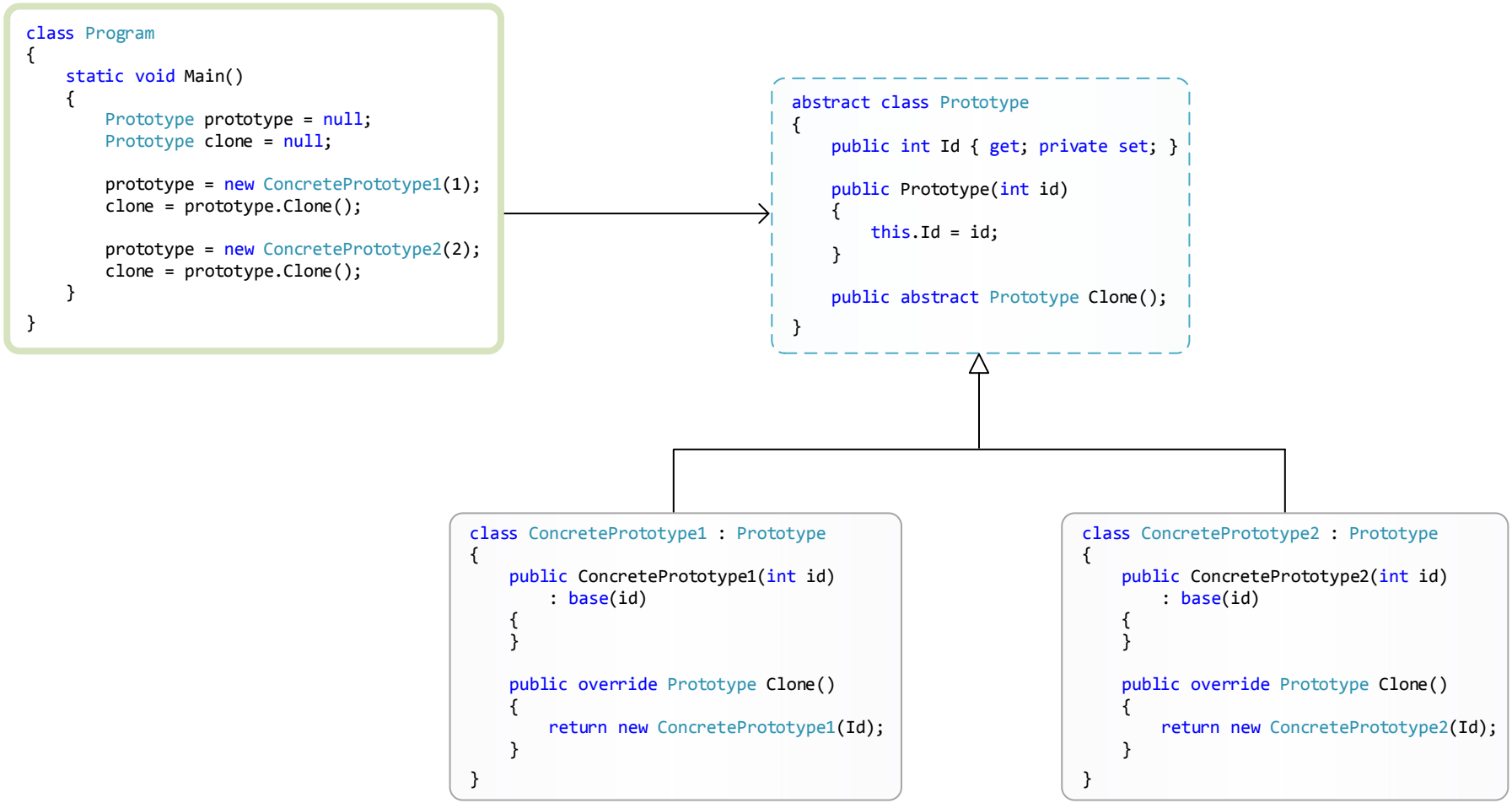
```
class ConcreteBuilder : Builder
{
    Product product = new Product();

    public override void BuildPartA()
    {
        product.Add("Part A");
    }

    public override void BuildPartB()
    {
        product.Add("Part B");
    }

    public override void BuildPartC()
    {
        product.Add("Part C");
    }

    public override Product GetResult()
    {
        return product;
    }
}
```

```
class Product
{
    ArrayList parts = new ArrayList();

    public void Add(string part)
    {
        parts.Add(part);
    }

    public void Show()
    {
        foreach (string part in parts)
            Console.WriteLine(part);
    }
}
```

Factory Method

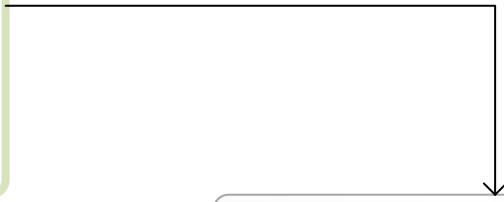


Prototype


class Program
{
static void Main()
{
Prototype prototype = null;
Prototype clone = null;
prototype = new ConcretePrototype1(1);
clone = prototype.Clone();
prototype = new ConcretePrototype2(2);
clone = prototype.Clone();
}
}
abstract class Prototype
{
public int Id { get; private set; }
public Prototype(int id)
{
this.Id = id;
}
public abstract Prototype Clone();
}
class ConcretePrototype1 : Prototype
{
public ConcretePrototype1(int id)
: base(id)
{
}
public override Prototype Clone()
{
return new ConcretePrototype1(Id);
}
}
class ConcretePrototype2 : Prototype
{
public ConcretePrototype2(int id)
: base(id)
{
}
public override Prototype Clone()
{
return new ConcretePrototype2(Id);
}
}

Singleton

```
class Program
{
    static void Main()
    {
        Singleton instance1 = Singleton.Instance();
        Singleton instance2 = Singleton.Instance();
        Console.WriteLine(ReferenceEquals(instance1, instance2));

        instance1.SingletonOperation();
        string singletonData = instance1.GetSingletonData();
        Console.WriteLine(singletonData);
    }
}
```



```
class Singleton
{
    static Singleton uniqueInstance;
    string singletonData = string.Empty;

    protected Singleton()
    {
    }

    public static Singleton Instance()
    {
        if (uniqueInstance == null)
            uniqueInstance = new Singleton();

        return uniqueInstance;
    }

    public void SingletonOperation()
    {
        singletonData = "SingletonData";
    }

    public string GetSingletonData()
    {
        return singletonData;
    }
}
```