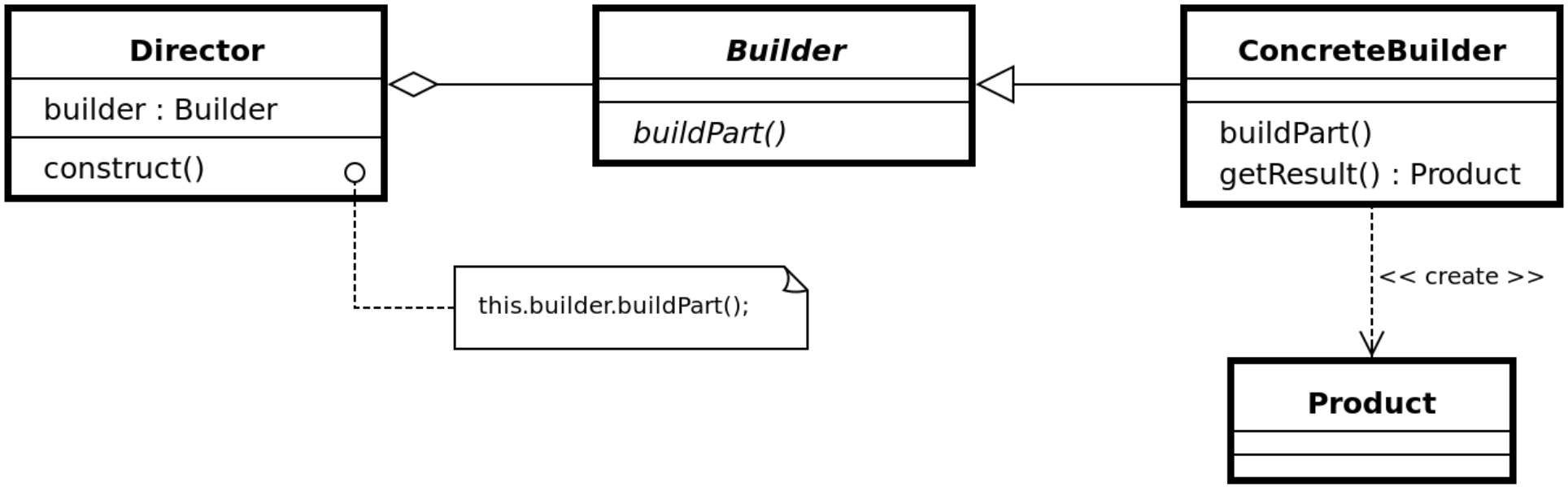
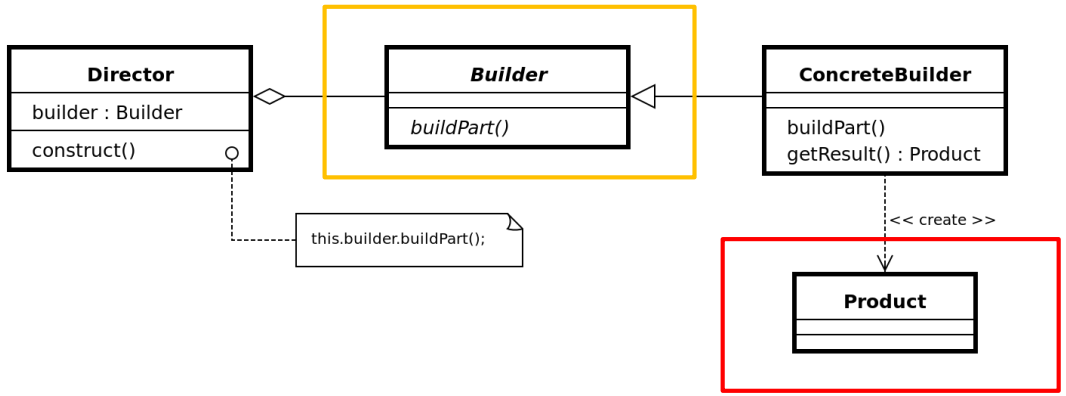
C# 빌더패턴

빌더 패턴

- 복합 객체의 생성 과정과 표현 방법을 분리하여 동일한 생성 절차에서 서로 다른 표현 결과를 만들 수 있게 하는 패턴



빌더 패턴의 구현

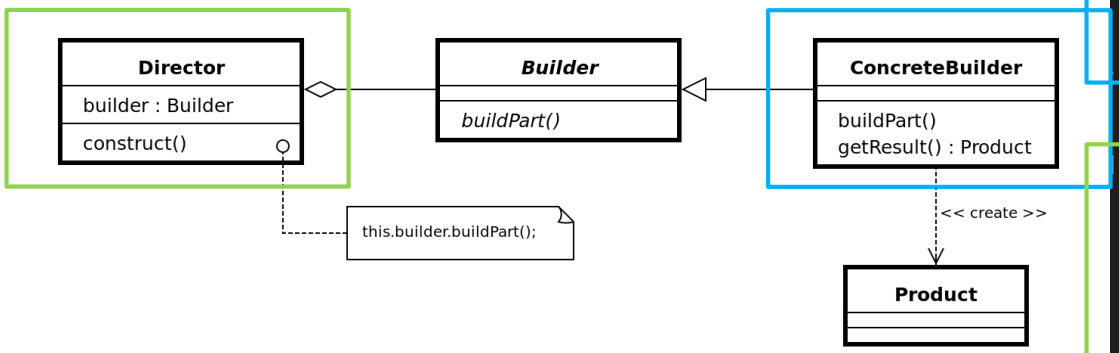



```
class Pizza
{
    string dough;
    string sauce;
    string topping;
    public Pizza() { }
    public void SetDough(string d) { dough = d; }
    public void SetSauce(string s) { sauce = s; }
    public void SetTopping(string t) { topping = t; }
}
```

```
//Abstract Builder
abstract class PizzaBuilder
{
    protected Pizza pizza;
    public PizzaBuilder() { }
    public Pizza GetPizza() { return pizza; }
    public void CreateNewPizza() { pizza = new Pizza(); }

    public abstract void BuildDough();
    public abstract void BuildSauce();
    public abstract void BuildTopping();
}
```

빌더 패턴의 구현




```
//Concrete Builder
class HawaiianPizzaBuilder : PizzaBuilder
{
    public override void BuildDough() { pizza.SetDough("cross"); }
    public override void BuildSauce() { pizza.SetSauce("mild"); }
    public override void BuildTopping() { pizza.SetTopping("ham+pineapple"); }
}
```

```
//Concrete Builder
class SpicyPizzaBuilder : PizzaBuilder
{
    public override void BuildDough() { pizza.SetDough("pan baked"); }
    public override void BuildSauce() { pizza.SetSauce("hot"); }
    public override void BuildTopping() { pizza.SetTopping("pepparoni+salami"); }
}
```

```
/** "Director" */
class Waiter
{
    private PizzaBuilder pizzaBuilder;

    public void SetPizzaBuilder(PizzaBuilder pb) { pizzaBuilder = pb; }
    public Pizza GetPizza() { return pizzaBuilder.GetPizza(); }

    public void ConstructPizza()
    {
        pizzaBuilder.CreateNewPizza();
        pizzaBuilder.BuildDough();
        pizzaBuilder.BuildSauce();
        pizzaBuilder.BuildTopping();
    }
}
```