

بررسی الگوهای طراحی –

Factory Method

توسط : محمد حسین زارع

@zare88r: twitter

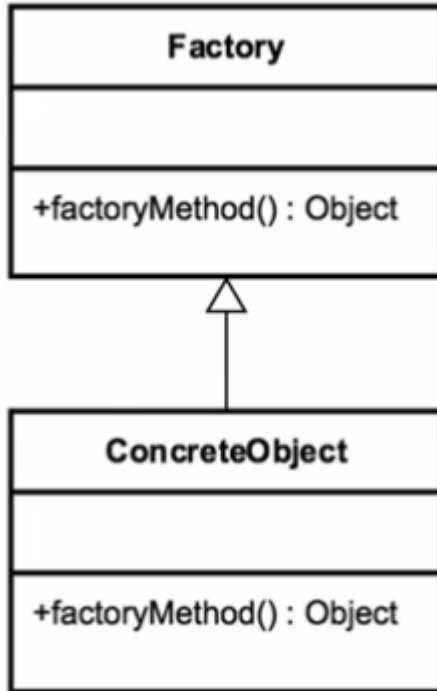
telegram

Join Us : @JavaLandCH

مفاهیم

- مخفی سازی لاجیک ساخت اشیاء
- درگیر شدن خود **factory** با کلاس های فرزند
- **Interface** های مشترک
- نمونه ها:
- **Calendar**
- **NumberFormat**
- **ResourceBundle**

طراحی



- الگویی برای ساخت اشیا مختلف با یک ساختار
- دارای interface برای ارائه استاندارد
- دارای یک یا چند کلاس پیاده سازی برای هر زیر ساختار
- دارای متود یا متودهایی با پارامتر های گوناگون

نمونه موجود

```
import java.util.Calendar;
```

```
Calendar calendar = Calendar.getInstance();  
System.out.println(calendar);  
System.out.println(calendar.get(Calendar.SECOND))
```

یک نمونه

```
public abstract class Calculation {  
    protected int amountPerMonth;  
    protected int taxPercent;  
    protected String product;  
  
    public Calculation(int amountPerMonth, int taxPercent, String product) {  
        this.amountPerMonth = amountPerMonth;  
        this.taxPercent = taxPercent;  
        this.product = product;  
    }  
  
    public abstract int calculate();  
}
```

یک نمونه

```
public class Calculation1 extends Calculation {  
  
    public Calculation1(int amountPerMonth, int taxPercent, String product) {  
        super(amountPerMonth, taxPercent, product);  
    }  
  
    @Override  
    public int calculate() {  
        return amountPerMonth * taxPercent + 200;  
    }  
}
```

یک نمونه

```
public class Calculation2 extends Calculation {  
    public Calculation2(int amountPerMonth, int taxPercent, String product) {  
        super(amountPerMonth, taxPercent, product);  
    }  
  
    @Override  
    public int calculate() {  
        return amountPerMonth * taxPercent ;  
    }  
}
```

یک نمونه

```
public class CalculationFactory {  
    public Calculation createCalculation(int amountPerMonth, int taxPercent,  
                                         String product, boolean ferstival) {  
        if (ferstival) {  
            return new Calculation1(amountPerMonth, taxPercent, product);  
        } else {  
            return new Calculation2(amountPerMonth, taxPercent, product);  
        }  
    }  
}
```


جمع بندی

Factory method نقطه مقابل **Singleton** است

پیچیدگی ساخت **sub class** ها را رفع میکند

به کد شما قابلیت گسترش میدهد

پیاده سازی آن منوط به نوشتن کد بیشتری است بنابراین خودش پیچیدگی دارد