生成器模式

生成器模式和工厂模式和抽象工厂模式一样也是创建型模式。这个模式用来解决当对象包含大量的属性时使用工厂模式和抽象工厂模式存在的一些问题。

当对象拥有大量属性时，使用工厂模式和抽象工厂模式存在三个主要的问题：

- 过多的参数从客户端程序传给工厂类，这样容易发生错误，因为多数时候参数的类型都是一样的，在客户端这边很难操作参数的顺序。
- 一些参数可能是可选的，但是在工厂类中，我们不得不传入所有的参数然后把可选的参数设置为NULL。
- 如果对象是重量级的且创建复杂，那么这些复杂性都会变成工厂类的一部分，整个工厂类变得很混乱。

我们可以通过提供一个构造器来构造需要的参数来解决大量参数的问题，然后使用不同的setter方法来设置可选的参数，但同样也会带来一个问题，对象的状态会变得不完整，除非显式设置了所有的属性。

生成器模式可以解决大量可选参数和状态不一致的问题，通过提供一种方式逐步构建对象，提供了一个方法返回最终的对象。

生成器模式实现

1.首先你需要创建一个静态内部类，然后拷贝所有的参数从外部类到建造者类。我们应该如下的命名约定，如果类名为`Computer`，那么其对应的生成器类应该命名为`ComputerBuilder`。

2.生成器类应该有一个公共的构造器，使用所有必要的属性作为参数。

3.生成器类应该有用来设置可选参数的方法，在设置完可选属性之后，应该返回同一个生成器对象。

4.最后一步，在建造者类中提供一个`build()`方法返回客户端程序需要的对象。为此，我们需要在类中提供一个私有的构造器，并使用建造者类作为参数。

```
package com.journaldev.design.builder;

public class Computer {

    //required parameters
    private String HDD;
    private String RAM;

    //optional parameters
    private boolean isGraphicsCardEnabled;
    private boolean isBluetoothEnabled;
```

```
    public String getHDD() {
        return HDD;
    }

    public String getRAM() {
        return RAM;
    }

    public boolean isGraphicsCardEnabled() {
        return isGraphicsCardEnabled;
    }

    public boolean isBluetoothEnabled() {
        return isBluetoothEnabled;
    }

    private Computer(ComputerBuilder builder) {
        this.HDD=builder.HDD;
        this.RAM=builder.RAM;
        this.isGraphicsCardEnabled=builder.isGraphicsCardEnabled;
        this.isBluetoothEnabled=builder.isBluetoothEnabled;
    }

    //Builder Class
    public static class ComputerBuilder{

        // required parameters
        private String HDD;
        private String RAM;

        // optional parameters
        private boolean isGraphicsCardEnabled;
        private boolean isBluetoothEnabled;

        public ComputerBuilder(String hdd, String ram){
            this.HDD=hdd;
            this.RAM=ram;
        }

        public ComputerBuilder setGraphicsCardEnabled(boolean
isGraphicsCardEnabled) {
            this.isGraphicsCardEnabled = isGraphicsCardEnabled;
            return this;
        }

        public ComputerBuilder setBluetoothEnabled(boolean
isBluetoothEnabled) {
            this.isBluetoothEnabled = isBluetoothEnabled;
            return this;
        }

        public Computer build(){
            return new Computer(this);
```

```
        }

    }

}
```

注意：Computer类仅有getter方法，没有公有构造器。因此，获得Computer对象的唯一方法就是通过ComputerBuilder类。

```
package com.journaldev.design.test;

import com.journaldev.design.builder.Computer;

public class TestBuilderPattern {

    public static void main(String[] args) {
        //Using builder to get the object in a single line of
code and
                //without any inconsistent state or arguments
management issues
        Computer comp = new Computer.ComputerBuilder(
                "500 GB", "2 GB").setBluetoothEnabled(true)
                .setGraphicsCardEnabled(true).build();
    }

}
```