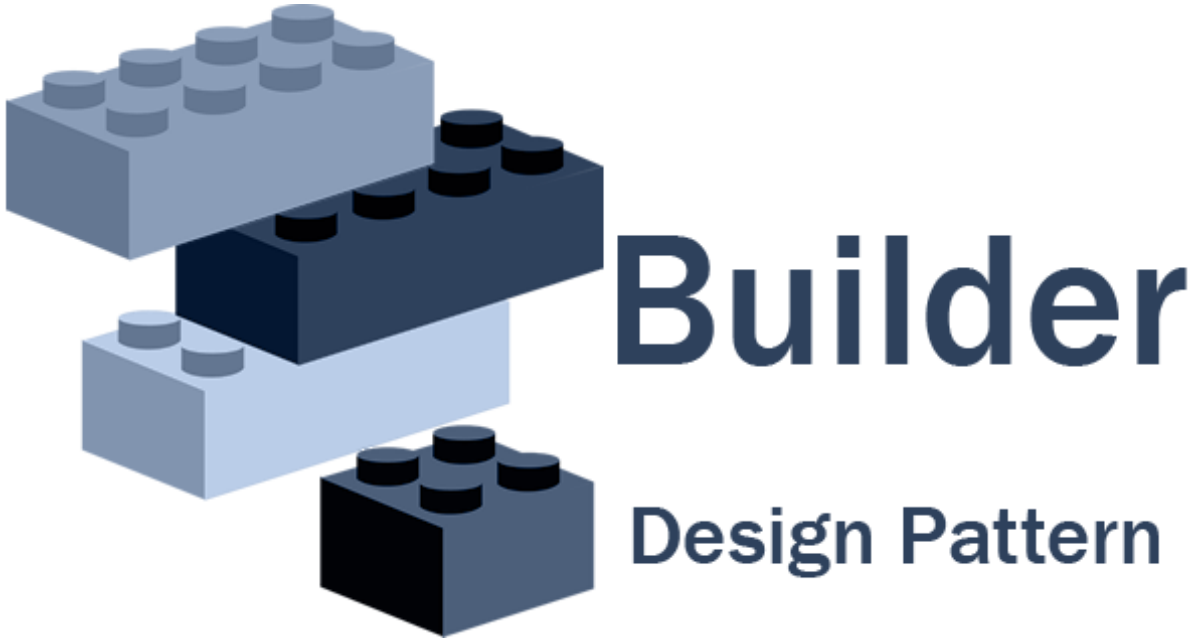
构建者(Builder)

波波老师~研发总监/资深架构师

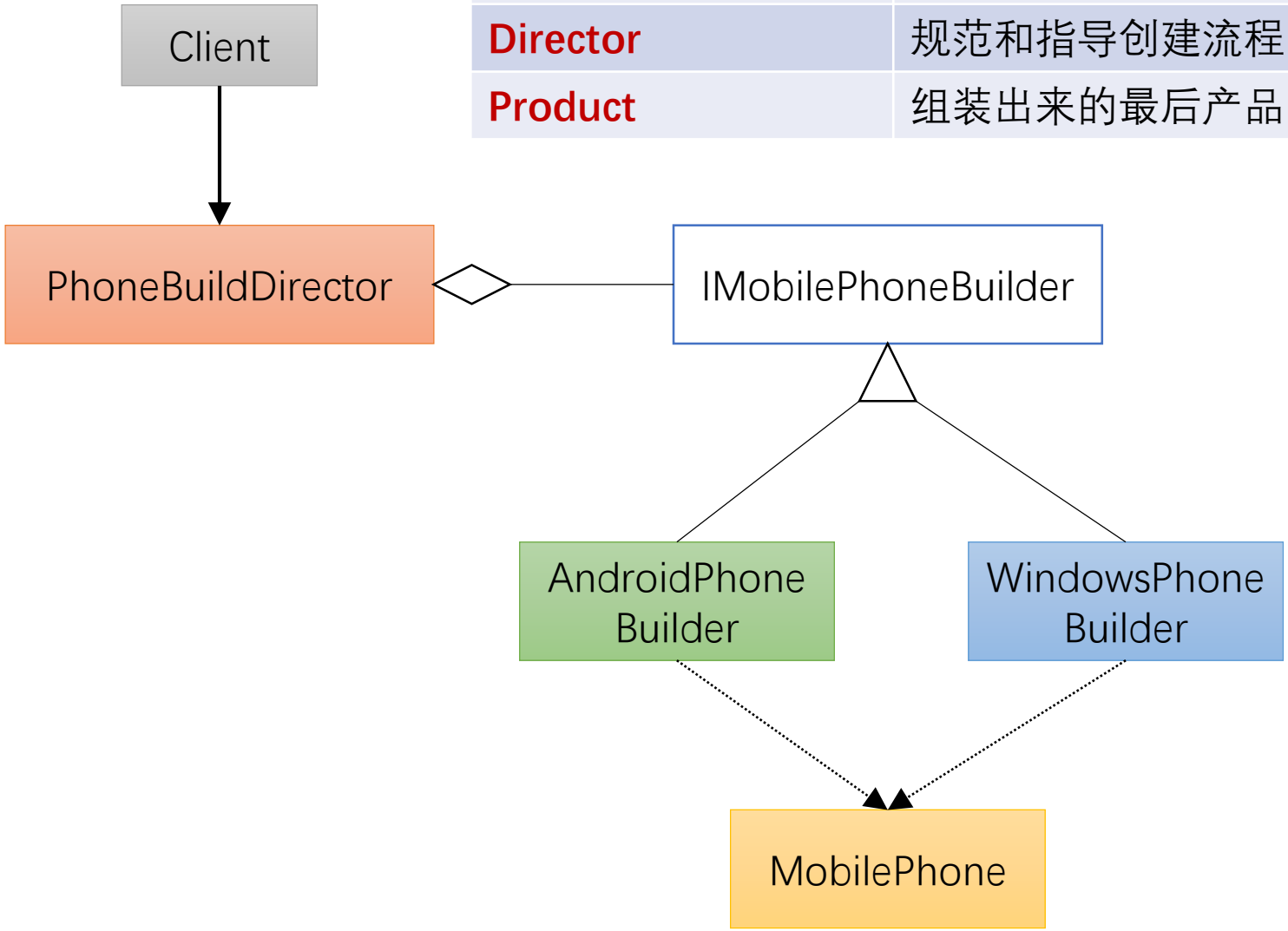


经典构造者模式定义

- 将复杂对象的构造和表示分离，相同的构造流程可以创建不同的表示。

关系图

角色	职责
ConcreteBuilder	负责创建复杂产品的具体类，知道每个部件的构造细节
Builder	创建实际产品的接口
Director	规范和指导创建流程
Product	组装出来的最后产品



代码~部件类型定义

```java
package io.spring2go.corespring.classicbuilder;

// 屏幕类型
public enum ScreenType {
    SCREENTYPE_TOUCH_CAPACITIVE, // 电容式
    SCREENTYPE_TOUCH_RESISTIVE, // 电阻式
    SCREENTYPE_NON_TOUCH
}
```

```java
package io.spring2go.corespring.classicbuilder;

// 触控笔
public enum Stylus {
    YES,
    NO
}
```

```java
package io.spring2go.corespring.classicbuilder;

// 电池容量
public enum Battery {
    MAH_1000,
    MAH_1500,
    MAH_2000
}
```

```java
package io.spring2go.corespring.classicbuilder;

// 操作系统
public enum OperatingSystem {
    ANDROID,
    WINDOWS_MOBILE,
    WINDOWS_PHONE,
    SYMBIAN
}
```

代码~产品类

```java
package io.spring2go.corespring.classicbuilder;

// 这是"Product"产品类
public class MobilePhone {
    // 部件类型
    private String phoneName;
    private ScreenType phoneScreen;
    private Battery phoneBattery;
    private OperatingSystem phoneOS;
    private Stylus phoneStylus;

    // region 访问手机组件的getter/setter公开方法

    @Override
    public String toString() {
        return String.format("Name: %s\nScreen: %s\nBattery: %s\nOS: %s\nStylus: %s\n",
                this.phoneName,
                this.phoneScreen,
                this.phoneBattery,
                this.phoneOS,
                this.phoneStylus);
    }
}
```

代码~构造者接口

```java
package io.spring2go.corespring.classicbuilder;

// 这个是构建者"Builder"接口
public interface IMobilePhoneBuilder {
    void buildScreen();
    void buildBattery();
    void buildOS();
    void buildStylus();
    MobilePhone getPhone();
}
```

代码~AndroidPhoneBuilder

```java
package io.spring2go.corespring.classicbuilder;

// 安卓手机具体构建者"ConcreteBuilder"
public class AndroidPhoneBuilder implements IMobilePhoneBuilder {

    private MobilePhone phone;

    public AndroidPhoneBuilder() {
        this.phone = new MobilePhone("Android Phone");
    }

    @Override
    public void buildScreen() {
        phone.setPhoneScreen(ScreenType.SCREENTYPE_TOUCH_RESISTIVE);
    }

    @Override
    public void buildBattery() {
        phone.setPhoneBattery(Battery.MAH_1500);
    }

    @Override
    public void buildOS() {
        phone.setPhoneOS(OperatingSystem.ANDROID);
    }

    @Override
    public void buildStylus() {
        phone.setPhoneStylus(Stylus.YES);
    }

    // 获得最终构建出来的产品
    @Override
    public MobilePhone getPhone() {
        return this.phone;
    }
}
```

代码~WindowsPhoneBuilder

```java
package io.spring2go.corespring.classicbuilder;

// Windows手机具体构建者"ConcreteBuilder"
public class WindowsPhoneBuilder implements IMobilePhoneBuilder  {

    private MobilePhone phone;

    public WindowsPhoneBuilder() {
        this.phone = new MobilePhone("Windows Phone");
    }

    @Override
    public void buildScreen() {
        phone.setPhoneScreen(ScreenType.SCREENTYPE_TOUCH_CAPACITIVE);
    }

    @Override
    public void buildBattery() {
        phone.setPhoneBattery(Battery.MAH_2000);
    }

    @Override
    public void buildOS() {
        phone.setPhoneOS(OperatingSystem.WINDOWS_PHONE);
    }

    @Override
    public void buildStylus() {
        phone.setPhoneStylus(Stylus.NO);
    }

    @Override
    public MobilePhone getPhone() {
        return this.phone;
    }

}
```

代码~导演Director

```java
package io.spring2go.corespring.classicbuilder;

// 这个是导演"Director"
public class Manufacturer {
    public void construct(IMobilePhoneBuilder phoneBuilder) {
        phoneBuilder.buildBattery();
        phoneBuilder.buildOS();
        phoneBuilder.buildScreen();
        phoneBuilder.buildStylus();
    }
}
```

代码~客户程序

```
<terminated> ClassicBuilderMain [Java Application] C:\
A new Phone built:

Name: Android Phone
Screen: SCREENTYPE_TOUCH_RESISTIVE
Battery: MAH_1500
OS: ANDROID
Stylus: YES

A new Phone built:

Name: Windows Phone
Screen: SCREENTYPE_TOUCH_CAPACITIVE
Battery: MAH_2000
OS: WINDOWS_PHONE
Stylus: NO
```

```java
package io.spring2go.corespring.classicbuilder;

// 客户程序
public class ClassicBuilderMain {

    public static void main(String[] args) {
        // 先创建导演Director
        Manufacturer manufacturer = new Manufacturer();
        // 先准备Builder接口
        IMobilePhoneBuilder phoneBuilder = null;

        // 制造一部安卓手机
        phoneBuilder = new AndroidPhoneBuilder();
        manufacturer.construct(phoneBuilder);
        String output = String.format("A new Phone built:\n\n%s", phoneBuilder.getPhone().toString());
        System.out.println(output);

        // 制造一部Windows手机
        phoneBuilder = new WindowsPhoneBuilder();
        manufacturer.construct(phoneBuilder);
        output = String.format("A new Phone built:\n\n%s", phoneBuilder.getPhone().toString());
        System.out.println(output);
    }
}
```

总结

- 复杂对象的构建
- 多步构造流程/算法
- 构造类似种类产品，构造流程相同，表示不同
- 构建和表示分离

问题

- 构建者和抽象工厂区别？

参考

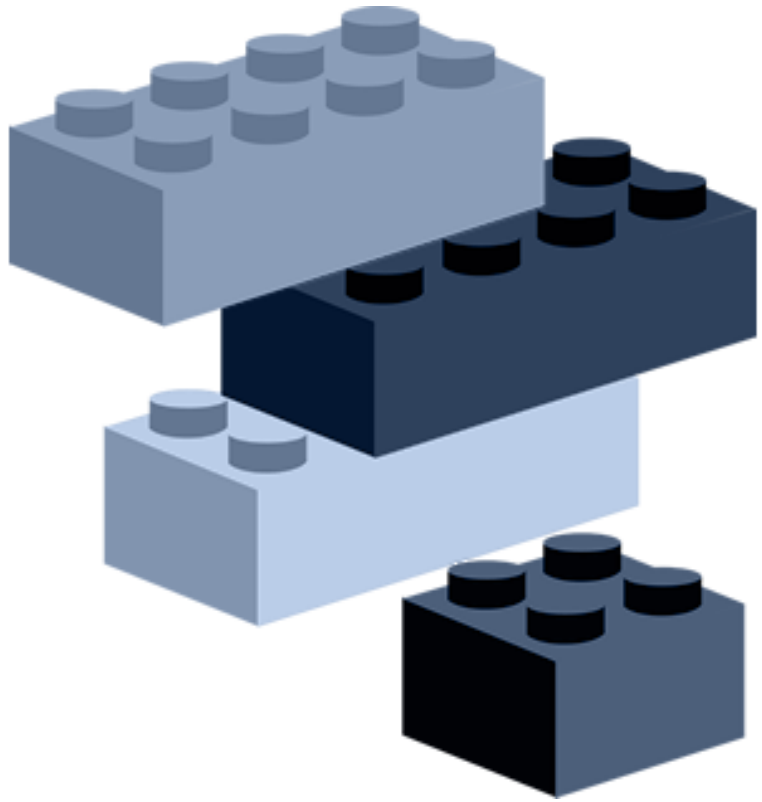
- Understanding and Implementing Builder Pattern
  - https://www.codeproject.com/Articles/470476/Understanding-and-Implementing-Builder-Pattern-in
- Builder Design Pattern
  - https://www.codeproject.com/Articles/1156619/Builder-Design-Pattern-with-Demo

代码

- https://github.com/spring2go/core-spring-patterns

波波微课

spring2go.com



Builder

Design Pattern