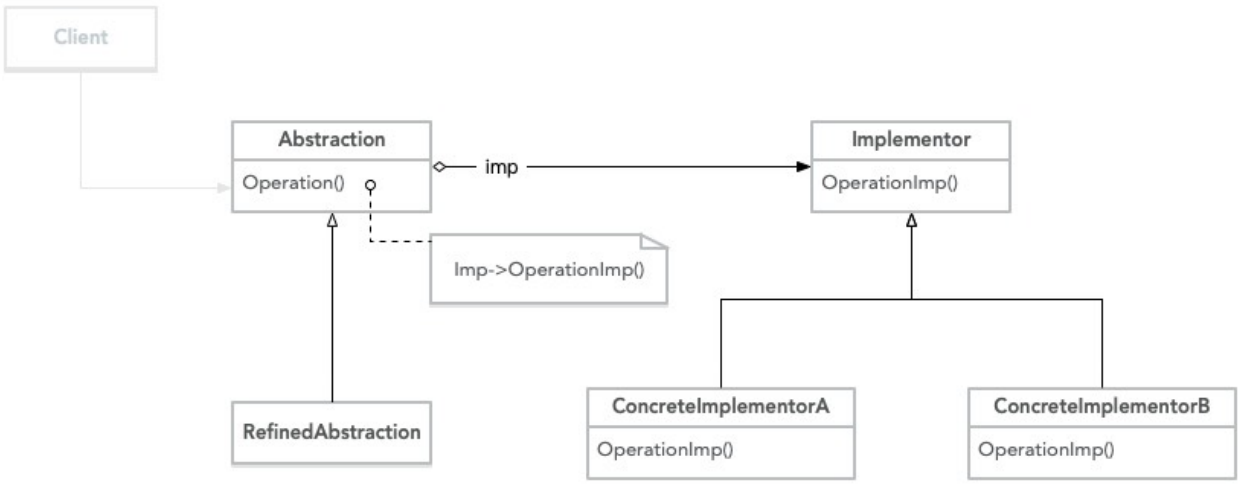
PHP设计模式之桥接模式

桥接模式，在程序世界中，其实就是组合/聚合的代名词。为什么这么说呢？熟悉面向对象的我们都知道继承的好处，子类可以共享父类的很多属性、功能。但是，继承也会带来一个问题，那就是严重的耦合性。父类的修改多少都会对子类产生影响，甚至一个方法或属性的修改都有可能让所有子类都去修改一遍。这样就违背了开放封装原则。而桥接就是为了解决这个问题，它强调的是用组合/聚合的方式来共享一些能用的方法。相信大家一定想到了php中的trait，如果你在工作中使用过这个特性，那么你就已经用过桥接模式了！

Gof类图及解释

GoF定义：将抽象部分与它的实现部分分离，使它们都可以独立地变化。

> GoF类图



> 代码实现

```php
interface Implementor
{
    public function OperationImp();
}

class ConcreteImplementorA implements Implementor
{
    public function OperationImp()
    {
        echo '具体实现A', PHP_EOL;
    }
}

```

```php
class ConcreteImplementorB implements Implementor
{
    public function OperationImp()
    {
        echo '具体实现B', PHP_EOL;
    }
}
```

我们先来定义实现接口以及它们具体的实现，也就是真正要执行的功能。就像是适配器模式中的Adaptee。

```php
abstract class Abstraction
{
    protected $imp;
    public function SetImplementor(Implementor $imp)
    {
        $this->imp = $imp;
    }
    abstract public function Operation();
}

class RefinedAbstraction extends Abstraction
{
    public function Operation()
    {
        $this->imp->OperationImp();
    }
}
```

定义抽象类的接口，并维护一个对实现的引用。具体的抽象类的实现方法中，我们直接调用实现接口的真实操作方法。类似于适配器中的Adapter。

```php
$impA = new ConcreteImplementorA();
$impB = new ConcreteImplementorB();

$ra = new RefinedAbstraction();

```

```php
$ra->SetImplementor($impA);
$ra->Operation();

$ra->SetImplementor($impB);
$ra->Operation();
```

客户端调用，我们的抽象类使用不用的实现类就可以让操作方法变成多态的感觉。

- 在源码解释中，我们会发现，这个模式和适配器模式非常相似。但是，适配器的目的是为了帮助两个不太相关的类，让它们能够协同工作，实现中间转换工作。而桥接则是为了让方法的行为解除继承耦合，方便地添加、修改，动态调用行为，让抽象接口和实现部分可以独立进行改变
- 让抽象接口和实现部分可以独立进行改变的意思是，只要维护了实现接口的引用，我们的实现接口的具体实现类可以是完全不同的类，里面有不同的功能，并且可以任意改变。让实现来自己决定它自己是什么。
- 桥接模式的优点：分享接口及其实现部分、提高可扩充性、实现细节对客户透明
- 桥接模式最主要解决的问题就是继承的不断增长而带来的紧耦合问题
- 组合与聚合：聚合是弱关系，A可以包含B，但B不是A的一部分；组合是强关系，A包含B，B也是A的一部分，整体和部分的关系

我们的手机有不同的型号，每个型号又要生产大致相同但不同的配件。比如X1手机壳、贴膜、耳机；X2的手机壳、贴膜、耳机等。受限于成本的问题，我们不会给每一个型号的手机都去生产完全不一样的配套配件。而是去尽量使用外部通用的配件（Implementor），让每一种型号的手机（Abstraction）去进行组合（Bridge），搭配售卖给消费者。这样，才不至于让我们的手机品牌太早的消耗完融资关门大吉。看来，做企业和学设计模式还真是有很多相关之处哦！！

完整代码：https://github.com/zhangyue0503/designpatterns-php/blob/master/18.bridge/source/bridge.php

实例

我们的短信发送也可以用桥接来实现。假设我们有很多的短信模板，然后搭配不同的短信提供商进行短信的发送。这时，我们就可以用桥接模式来形成各种不同的组合。

> 短信发送类图

完整源码：https://github.com/zhangyue0503/designpatterns-php/blob/master/18.bridge/source/bridge-message.php

```php
<?php

interface MessageTemplate
{
```

```php
    public function GetTemplate();
}

class LoginMessage implements MessageTemplate
{
    public function GetTemplate()
    {
        echo '您的登录验证码是【AAA】，请不要泄露给他人【XXX公司】！', PHP_EOL;
    }
}
class RegisterMessage implements MessageTemplate
{
    public function GetTemplate()
    {
        echo '您的注册验证码是【BBB】，请不要泄露给他人【XXX公司】！', PHP_EOL;
    }
}
class FindPasswordMessage implements MessageTemplate
{
    public function GetTemplate()
    {
        echo '您的找回密码验证码是【CCC】，请不要泄露给他人【XXX公司】！', PHP_EOL;
    }
}

abstract class MessageService
{
    protected $template;
    public function SetTemplate($template)
    {
        $this->template = $template;
    }
    abstract public function Send();
}

class AliYunService extends MessageService
{
    public function Send()
    {
        echo '阿里云开始发送短信：';
```

```php
        $this->template->GetTemplate();
    }
}

class JiGuangService extends MessageService
{
    public function Send()
    {
        echo '极光开始发送短信：';
        $this->template->GetTemplate();
    }
}

// 三个短信模板
$loginTemplate = new LoginMessage();
$registerTemplate = new RegisterMessage();
$findPwTemplate = new FindPasswordMessage();

// 两个短信服务商
$aliYun = new AliYunService();
$jg = new JiGuangService();

// 随意组合
// 极光发注册短信
$jg->SetTemplate($registerTemplate);
$jg->Send();

// 阿里云发登录短信
$aliYun->SetTemplate($loginTemplate);
$aliYun->Send();

// 阿里云发找回密码短信
$aliYun->SetTemplate($findPwTemplate);
$aliYun->Send();

// 极光发登录短信
$jg->SetTemplate($loginTemplate);
$jg->Send();

```

```
84 // ......
```

> 说明

- 这就是一种聚合模式。模板并不是短信发送的一部分，我们不使用模板直接发送也可以，它们没有强关系
- 短信发送商的发送方法无需改变，只需要传入不同的短信模板就可以实现各种模板的快速发送
- 在不确定是否一定是is-a的关系的情况下，更推荐用桥接模式这种组合/聚合形式的设计方法，如果确定当前的类关系是is-a，那么就不要犹豫的用继承吧