PHP Design Patterns Documentation

Release

Sander van Mook

Feb 26, 2018

Contents:

CHAPTER 1

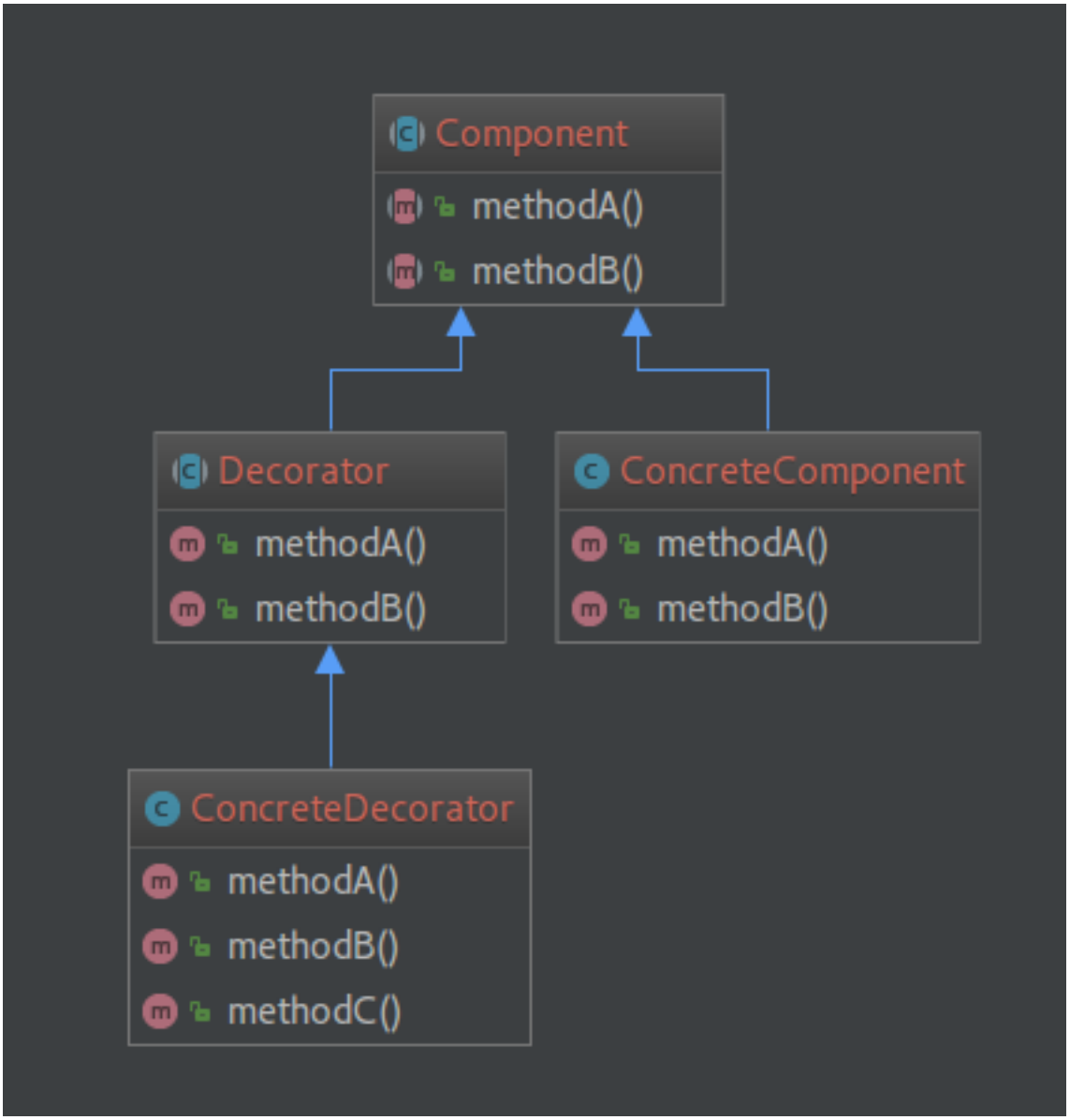
Structural Design Patterns

1.1 Decorator

The GoF describes this pattern as follows:

Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

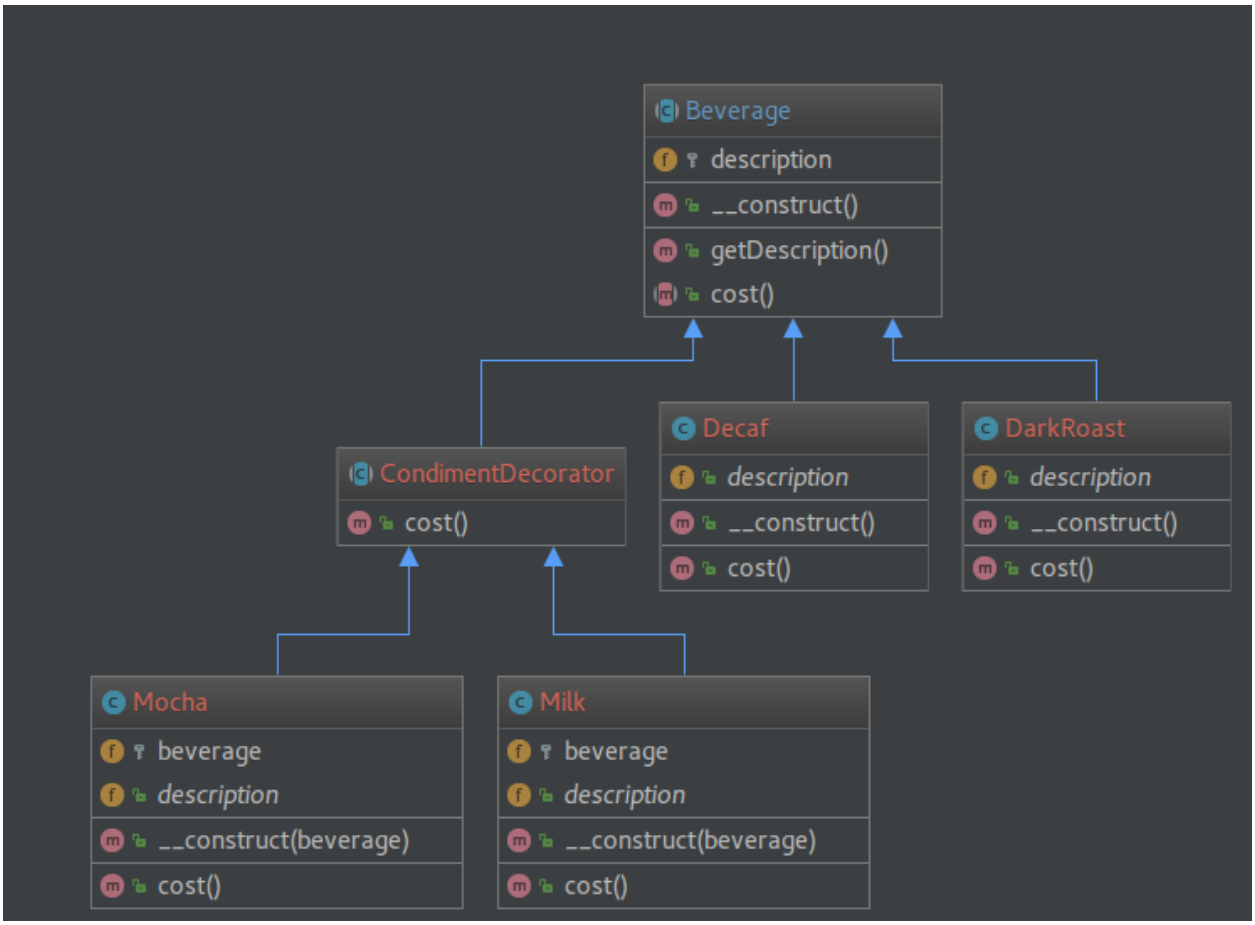
Lets start by looking at the class diagram:



There are two important parts in this pattern, the Components and the Decorators. The decorator holds a reference to the component it's decorating. The component can be an interface or an abstract class, based on your needs. The concrete components implement the component class. The decorators also implement the component, this is a important piece of the decorator pattern. The decorator is usually an abstract class and the concrete decorators implement it. The most important reason both concrete parts implement the component superclass is that we can threat each class in the same way. We want to be able to wrap any decorator around any component.

1.2 Example

Lets make things more clear with a example. In this example we are running a coffee shop. We have different kinds of coffee, these will be our Components. We will add extra's to our coffee which will be our decorators.



Beverage.php

```php
<?php


namespace PHPDesignPatterns\code\Examples\Structural\Decorator;

abstract class Beverage
{
    protected $description;

    public function __construct()
    {
        $this->description = 'Unknown Beverage';
    }

    public function getDescription() : string
    {
        return $this->description;
    }

    abstract public function cost() : float;
}
```

CondimentDecorator.php

```php
<?php


namespace PHPDesignPatterns\code\Examples\Structural\Decorator;

abstract class CondimentDecorator extends Beverage
{
    public function cost(): float
    {
    }
}
```

DarkRoast.php

```php
<?php


namespace PHPDesignPatterns\code\Examples\Structural\Decorator;

class DarkRoast extends Beverage
{
    public function __construct()
    {
        parent::__construct();
        $this->description = 'Dark Roast Coffee';
    }

    public function cost() : float
    {
        return 1.50;
    }
}
```

Decaf.php

```php
<?php


namespace PHPDesignPatterns\code\Examples\Structural\Decorator;

class Decaf extends Beverage
{
    public function __construct()
    {
        parent::__construct();
        $this->description = 'Decaf Coffee';
    }

    public function cost(): float
    {
        return 2.5;
    }
}
```

Milk.php

```php
<?php

```

```php

namespace PHPDesignPatterns\code\Examples\Structural\Decorator;

class Milk extends CondimentDecorator
{
    protected $beverage;

    public function __construct(Beverage $beverage)
    {
        parent::__construct();
        $this->beverage = $beverage;
        $this->description = $this->beverage->getDescription() . ' with Milk';
    }

    public function cost() : float
    {
        return 0.75 + $this->beverage->cost();
    }
}
```

Mocha.php

```php
<?php


namespace PHPDesignPatterns\code\Examples\Structural\Decorator;

class Mocha extends CondimentDecorator
{
    protected $beverage;

    public function __construct(Beverage $beverage)
    {
        parent::__construct();
        $this->beverage = $beverage;
        $this->description = $this->beverage->getDescription() . ' with Mocha';
    }

    public function cost() : float
    {
        return 1.75 + $this->beverage->cost();
    }
}
```

context code here

CHAPTER 2

Indices and tables

- genindex
- modindex
- search