

# 桥接模式

提供起子把手的抽象类

项目

Screwdriver C:\IDEA\TEXT\Screwdriver

> .idea

> src

> main

> java

> org.example

BridgePatternDemo

FlatHead

Handle

PhillipsHead

ScrewdriverHandle095

ScrewdriverHandle110

ScrewdriverHead

resources

test

target

.gitignore

pom.xml

```
1 package org.example;
2
3 public abstract class Handle {
4     private ScrewdriverHead head;
5
6     public void ScrewdriverHead(ScrewdriverHead head) {
7         this.head = head;
8     }
9
10    public abstract void rotate();
11 }
```

实现两个不同的起子把手对象

项目

Screwdriver C:\IDEA\TEXT\Screwdriver

> .idea

> src

> main

> java

> org.example

BridgePatternDemo

FlatHead

Handle

PhillipsHead

ScrewdriverHandle095

ScrewdriverHandle110

ScrewdriverHead

resources

test

target

.gitignore

pom.xml

外部库

临时文件和控制台

```
1 package org.example;
2
3 public class ScrewdriverHandle110 extends Handle{
4     protected ScrewdriverHead head;
5
6     public void ScrewdriverHead(ScrewdriverHead head) {
7         this.head = head;
8     }
9
10    public void rotate(){
11        System.out.println("使用110的把手" );
12        head.screwIn();
13        head.screwOut();
14    }; // 旋转手柄以拧紧或拧松螺丝
15 }
```

项目

Screwdriver C:\IDEA\TEXT\Screwdriver

> .idea

> src

> main

> java

> org.example

BridgePatternDemo

FlatHead

Handle

PhillipsHead

ScrewdriverHandle095

ScrewdriverHandle110

ScrewdriverHead

resources

test

target

.gitignore

pom.xml

外部库

临时文件和控制台

```
1 package org.example;
2
3 public class ScrewdriverHandle095 extends Handle{
4     private ScrewdriverHead head;
5
6     public void ScrewdriverHead(ScrewdriverHead head) {
7         this.head = head;
8     }
9
10    public void rotate(){
11        System.out.println("使用095的把手" );
12        head.screwIn();
13        head.screwOut();
14    }; // 旋转手柄以拧紧或拧松螺丝
15 }
```

## 起子头的接口

The screenshot shows an IDE with a project named 'Screwdriver' at 'C:\IDEATEXT\Screwdriver'. The project structure on the left includes 'src/main/java/org/example' with files like 'BridgePatternDemo', 'FlatHead', 'Handle', 'PhillipsHead', 'ScrewdriverHandle095', 'ScrewdriverHandle110', and 'ScrewdriverHead'. The 'ScrewdriverHead' file is selected and its code is displayed on the right:

```
1 package org.example;
2
3 8 个用法 2 个实现
4 public interface ScrewdriverHead {
5     2 个用法 2 个实现
6     void screwIn(); // 拧入螺丝
7     2 个用法 2 个实现
8     void screwOut(); // 拧出螺丝
9 }
10
```

## 两个不同起子头的对象

The screenshot shows the same IDE with the 'PhillipsHead' file selected. The project structure on the left is the same, but 'PhillipsHead' is highlighted. The code for 'PhillipsHead.java' is shown on the right:

```
1 package org.example;
2
3 2 个用法
4 public class PhillipsHead implements ScrewdriverHead {
5     2 个用法
6     @Override
7     public void screwIn() {
8         System.out.println("Phillips head screwing in...");
9     }
10
11     2 个用法
12     @Override
13     public void screwOut() {
14         System.out.println("Phillips head screwing out...");
15     }
16 }
17
```

The screenshot shows the same IDE with the 'FlatHead' file selected. The project structure on the left is the same, but 'FlatHead' is highlighted. The code for 'FlatHead.java' is shown on the right:

```
1 package org.example;
2
3 2 个用法
4 public class FlatHead implements ScrewdriverHead {
5     2 个用法
6     @Override
7     public void screwIn() {
8         System.out.println("Flat head screwing in...");
9     }
10
11     2 个用法
12     @Override
13     public void screwOut() {
14         System.out.println("Flat head screwing out...");
15     }
16 }
17
```

## 实现



```
1 package org.example;
2
3 public class BridgePatternDemo {
4     public static void main(String[] args) {
5         // 创建一个带有平头螺丝刀头的手柄
6         Handle flatHeadHandle = new ScrewdriverHandle095();
7         flatHeadHandle.ScrewdriverHead(new FlatHead());
8         flatHeadHandle.rotate();
9         flatHeadHandle.ScrewdriverHead(new PhillipsHead());
10        flatHeadHandle.rotate();
11        // 创建一个带有十字头螺丝刀头的手柄
12        Handle phillipsHeadHandle = new ScrewdriverHandle110();
13        phillipsHeadHandle.ScrewdriverHead(new FlatHead());
14        phillipsHeadHandle.rotate();
15        phillipsHeadHandle.ScrewdriverHead(new PhillipsHead());
16        phillipsHeadHandle.rotate();
17    }
18 }
```

桥接模式是一种将抽象部分与实现部分分离的设计模式，上述代码实现了起子头和起子把手的分离，实现的不同的起子把手和不同的起子头的结合，减少了类的生成，更降低了耦合度。