بررسی الگوهای طراحی

—

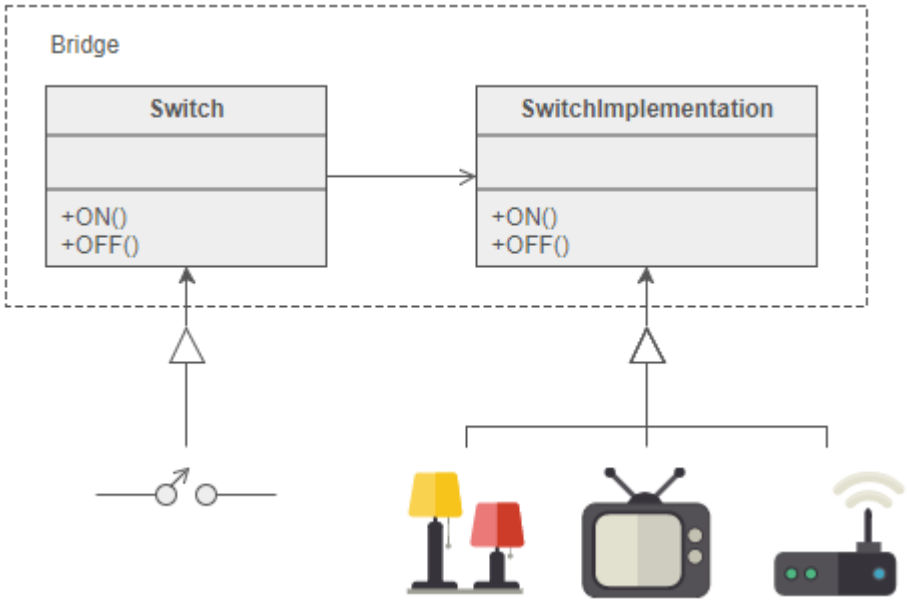
Bridge

توسط : محمد حسین زارع
@HoseinZare67: twitter
telegram
Join Us : @JavaLandCH
http://javaland.blog.ir/

مفاهیم

- جدا کردن abstraction از پیاده سازی به شکلی که مستقل عمل کنند
- انتشار interface در یک ساختار کلاسی ، و پیاده سازی آن در یک ساختار مجزا
- تغییر در interface اصلی تأثیری در پیاده سازی ها ندارد
- نمونه ها :
- Driver ها
- JDBC

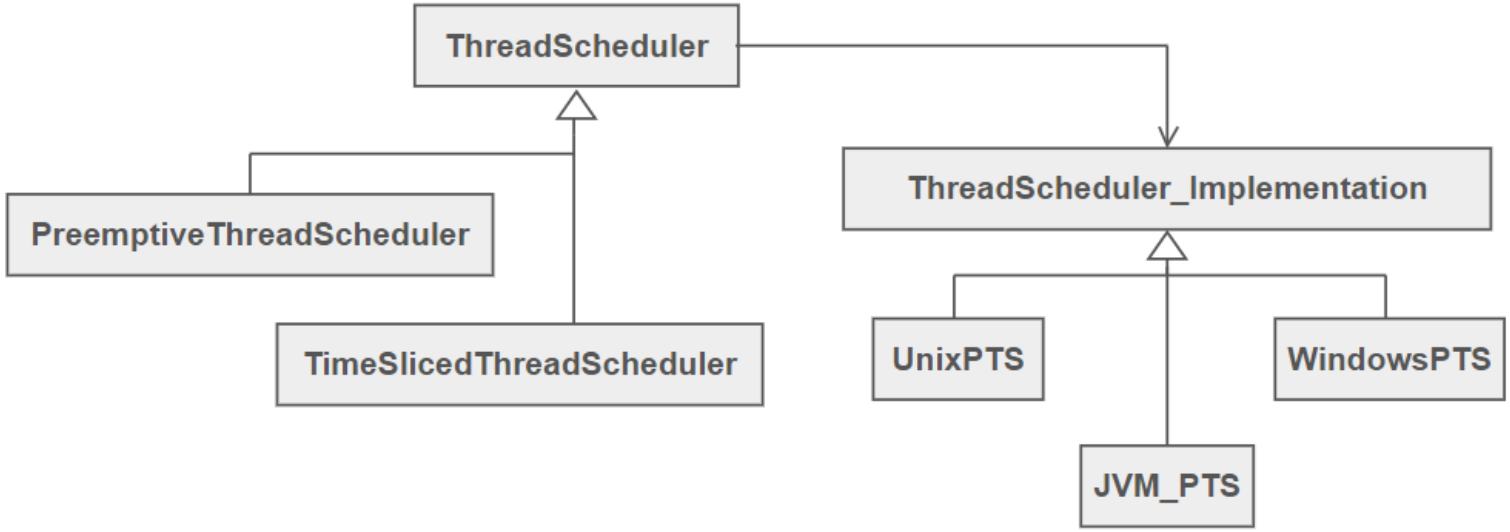
مفاهیم



طراحی

- استفاده از interface ها یا abstract class ها
- استفاده از composition به جای ارث بری در پیاده ساز ها
- هر تغییری در نمای کلی api نیاز به تغییر در هم دو سمت دارد

طراحی

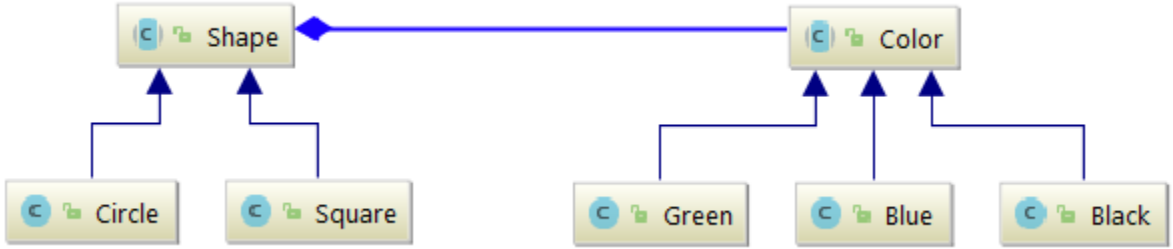

ThreadScheduler
PreemptiveThreadScheduler
TimeSlicedThreadScheduler
ThreadScheduler_Implementation
UnixPTS
WindowsPTS
JVM_PTS

مثال 1

```
try {
    //JDBC is an API
    DriverManager.registerDriver(new
         org.apache.derby.jdbc.EmbeddedDriver());
    String dbUrl = "jdbc:derby:memory:codejava/webdb;create=true";
    Connection conn = DriverManager.getConnection(dbUrl);
    Statement sta = conn.createStatement();
//This client is an abstraction and can work with any database that has a driver
    sta.executeUpdate("CREATE TABLE Address (ID INT, StreetName
VARCHAR(20), City VARCHAR(20))");
    System.out.println("Table created");

} catch (SQLException e) {
    e.printStackTrace();
}
```

یک نمونه



یک نمونه

```java
public abstract class Color {
    public abstract void applyColor();
}


public class Blue extends Color {
    @Override
    public void applyColor() {
        System.out.println("Blue Color");
    }
}
```

یک نمونه

```
public class Green extends Color {
    @Override
    public void applyColor() {
        System.out.println("Green");
    }
}

public class Black extends Color {
    @Override
    public void applyColor() {
        System.out.println("Black");
    }
}
```

یک نمونه

```
public abstract class Shape {
    protected Color color;

    public Shape(Color color) {
        this.color = color;
    }

    public abstract void applyColor();

    public abstract void draw();
}
```

یک نمونه

```
public class Circle extends Shape {
    public Circle(Color color) {
        super(color);
    }
    @Override
    public void applyColor() {
        color.applyColor();
        System.out.println("DRAW");
    }
    @Override
    public void draw() {

    }
}
```

یک نمونه

```
public class Main {

    public static void main(String[] args) {
        Color color = new Blue();

        Shape circle = new Circle(color);
        circle.applyColor();
    }
}
```

جمع بندی

- در برنامه ایجاد انعطاف میکند
- پیاده سازی آن دارای پیچیدگی است
- بیش از یک composition ساده است
- موجب مخفی سازی پیاده سازی ها از استفاده کننده api میشود