

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

-----o0o-----



Báo cáo môn học
Đặc tả phần mềm
Cài đặt các mẫu thiết kế

Giảng viên hướng dẫn: Trần Văn Dũng

Sinh viên thực hiện:

Họ tên	Lớp	Mã sinh viên
Nguyễn Quang Phúc	CNTT3 - K60	191200803

Hà Nội tháng 11 năm 2022

MỤC LỤC

1. MẪU THIẾT KẾ FACTORY METHOD	2
1.1. MÔ TẢ BÀI TOÁN	2
1.2. CÀI ĐẶT	2
1.3. KẾT QUẢ CHẠY CHƯƠNG TRÌNH	3
2. MẪU THIẾT KẾ PROTOTYPE	3
2.1. MÔ TẢ BÀI TOÁN	3
2.2. CÀI ĐẶT	4
2.3. KẾT QUẢ CHẠY CHƯƠNG TRÌNH	5
3. MẪU THIẾT KẾ ADAPTER	5
3.1. MÔ TẢ BÀI TOÁN	5
3.2. CÀI ĐẶT	5
3.3. KẾT QUẢ CHẠY CHƯƠNG TRÌNH	6
4. MẪU THIẾT KẾ ITERATOR	7
4.1. MÔ TẢ BÀI TOÁN	7
4.2. CÀI ĐẶT	7
4.3. CHẠY CHƯƠNG TRÌNH	9
5. MẪU THIẾT KẾ MEMENTO	9
5.1. MÔ TẢ BÀI TOÁN	9
5.2. CÀI ĐẶT	9
5.3. CHẠY CHƯƠNG TRÌNH	12
6. MẪU THIẾT KẾ TEMPLATE METHOD	12
6.1. MÔ TẢ BÀI TOÁN	12
6.2. CÀI ĐẶT	13
6.3. CHẠY CHƯƠNG TRÌNH	15

1. Mẫu thiết kế Factory Method

1.1. Mô tả bài toán

Website dự báo thời tiết hiện đang dùng API của hai trang AccuWeather và DarkSky. Tuy nhiên trong tương lai sẽ tích hợp thêm API của các trang khác và mong muốn của website là không cần thay đổi code khi làm vậy. Với yêu cầu như vậy, chúng ta có thể sử dụng mẫu thiết kế Factory Method để giải quyết bài toán này.

1.2. Cài đặt

Lớp IWeather.

```
public interface IWeather {  
    String getName();  
}
```

Lớp AccuWeather và DarkSky implements lại IWeather.

```
public class AccuWeather implements IWeather {  
    @Override  
    public String getName() {  
        return "AccuWeather";  
    }  
}
```

```
public class DarkSky implements IWeather {  
    @Override  
    public String getName() {  
        return "DarkSky";  
    }  
}
```

Lớp WeatherType chứa hai loại API thời tiết đang có.

```
public enum WeatherType {  
    ACCU_WEATHER,  
    DARK_SKY  
}
```

Lớp WeatherFactory sẽ trả về một lớp tương ứng với weatherType truyền vào.

```
public class WeatherFactory {  
  
    private WeatherFactory() {  
    }  
  
    public static IWeather getWeather(WeatherType weatherType) {  
        if (weatherType == WeatherType.ACCU_WEATHER) {  
            return new AccuWeather();  
        }  
        else if (weatherType == WeatherType.DARK_SKY) {  
            return new DarkSky();  
        }  
        else {  
            throw new IllegalArgumentException("Loại thời tiết không hợp lệ");  
        }  
    }  
}
```

Lớp Main chạy chương trình.

```
public class Main {  
  
    public static void main(String[] args) {  
        IWeather weather = WeatherFactory.getWeather(WeatherType.ACCU_WEATHER);  
        System.out.println(weather.getName());  
    }  
}
```

1.3. Kết quả chạy chương trình



```
factory_method.Main ×  
"C:\Program Files\Java\jdk-17.0.5\  
AccuWeather  
  
Process finished with exit code 0
```

2. Mẫu thiết kế Prototype

2.1. Mô tả bài toán

Một chiếc điện thoại khi được xuất xưởng sẽ có cấu hình giống nhau tuy nhiên khác nhau về ngôn ngữ cài sẵn dựa theo thị trường chiếc điện thoại đó được bán. Với yêu cầu như vậy, chúng ta có thể sử dụng mẫu thiết kế Prototype để tạo một chiếc điện thoại chuẩn và clone lại cho các điện thoại khác mà không cần khởi tạo lại từ đầu.

2.2. Cài đặt

Lớp Smartphone chứa thông tin về điện thoại.

```
public class Smartphone implements Cloneable {

    private String name;
    private int storageCapacity;
    private int ram;
    private int batteryCapacity;
    private String language;

    public Smartphone(String name, int storageCapacity, int ram, int
batteryCapacity, String language) {
        this.name = name;
        this.storageCapacity = storageCapacity;
        this.ram = ram;
        this.batteryCapacity = batteryCapacity;
        this.language = language;
    }

    @Override
    protected Smartphone clone() {
        try {
            return (Smartphone) super.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
        return null;
    }

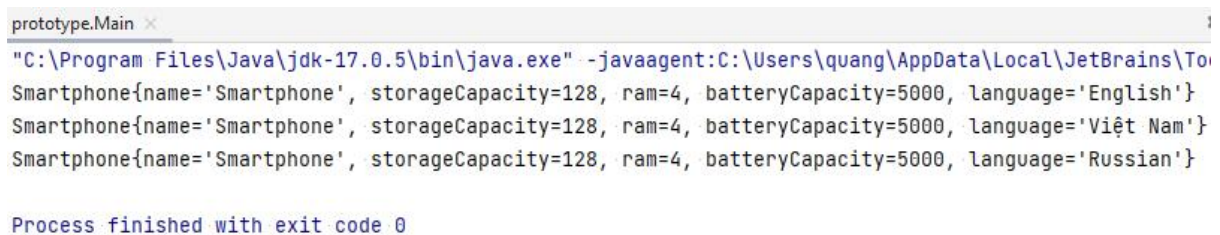
    public void setLanguage(String language) {
        this.language = language;
    }

    @Override
    public String toString() {
        return "Smartphone{" +
            "name='" + name + '\'' +
            ", storageCapacity=" + storageCapacity +
            ", ram=" + ram +
            ", batteryCapacity=" + batteryCapacity +
            ", language='" + language + '\'' +
            '}';
    }
}
```

Lớp Main chạy chương trình.

```
public class Main {  
  
    public static void main(String[] args) {  
        Smartphone prototype = new Smartphone("Smartphone", 128, 4, 5000,  
"English");  
  
        Smartphone vietnamSmartphone = prototype.clone();  
        vietnamSmartphone.setLanguage("Việt Nam");  
  
        Smartphone russiaSmartphone = prototype.clone();  
        russiaSmartphone.setLanguage("Russian");  
  
        System.out.println(prototype);  
        System.out.println(vietnamSmartphone);  
        System.out.println(russiaSmartphone);  
    }  
}
```

2.3. Kết quả chạy chương trình



```
prototype.Main ×  
"C:\Program Files\Java\jdk-17.0.5\bin\java.exe" -javaagent:C:\Users\quang\AppData\Local\JetBrains\To  
Smartphone{name='Smartphone', storageCapacity=128, ram=4, batteryCapacity=5000, language='English'}  
Smartphone{name='Smartphone', storageCapacity=128, ram=4, batteryCapacity=5000, language='Việt Nam'}  
Smartphone{name='Smartphone', storageCapacity=128, ram=4, batteryCapacity=5000, language='Russian'}  
  
Process finished with exit code 0
```

3. Mẫu thiết kế Adapter

3.1. Mô tả bài toán

Một người Việt muốn giao tiếp với một người Anh nhưng hai người này không biết ngôn ngữ của nhau. Điều đó đặt ra cần phải có một người trung gian để phiên dịch từ tiếng Việt sang tiếng Anh. Với yêu cầu như vậy, chúng ta có thể sử dụng mẫu thiết kế Adapter để giải quyết bài toán này.

3.2. Cài đặt

Lớp `IVietnameseTarget` là người nói tiếng Việt.

```
public interface IVietnameseTarget {  
  
    void send(String words);  
}
```

Lớp EnglishAdaptee là nghe tiếng Anh.

```
public class EnglishAdaptee {  
  
    public void receive(String words) {  
        System.out.println("English receive: " + words);  
    }  
}
```

Lớp TranslatorAdapter là người dịch từ tiếng Việt sang tiếng Anh.

```
public class TranslatorAdapter implements IVietnameseTarget {  
  
    private final EnglishAdaptee adaptee;  
  
    public TranslatorAdapter(EnglishAdaptee adaptee) {  
        this.adaptee = adaptee;  
    }  
  
    @Override  
    public void send(String words) {  
        System.out.println("Send: " + words);  
        String vietnameseWords = translate(words);  
        adaptee.receive(vietnameseWords);  
    }  
  
    private String translate(String vietnameseWords) {  
        return "Hello";  
    }  
}
```

Lớp Main chạy chương trình.

```
public class Main {  
  
    public static void main(String[] args) {  
        IVietnameseTarget vietnameseTarget = new TranslatorAdapter(new  
        EnglishAdaptee());  
        vietnameseTarget.send("Xin chào");  
    }  
}
```

3.3. Kết quả chạy chương trình

```
adapter.Main x  
"C:\Program Files\Java\jdk-17.0.5\  
Send: Xin chào  
English receive: Hello  
  
Process finished with exit code 0
```

4. Mẫu thiết kế Iterator

4.1. Mô tả bài toán

Một cửa hàng có danh sách các laptop đang bán và ta cần phải truy cập tuần tự tới danh sách trên. Với yêu cầu như vậy, chúng ta có thể sử dụng mẫu thiết kế Iterator để giải quyết bài toán này.

4.2. Cài đặt

Lớp Laptop chứa thông tin về laptop.

```
public class Laptop {  
  
    private String name;  
    private int price;  
  
    public Laptop(String name, int price) {  
        this.name = name;  
        this.price = price;  
    }  
  
    @Override  
    public String toString() {  
        return "Laptop{" +  
            "name='" + name + '\'' +  
            ", price=" + price +  
            '}';  
    }  
}
```

Lớp Iterator.

```
public interface Iterator<T> {  
  
    boolean hasNext();  
  
    T next();  
}
```


Lớp Shop chứa mảng các laptop cần duyệt.

```
public class Shop {  
  
    private final ArrayList<Laptop> laptops = new ArrayList<>();  
  
    public void addItem(Laptop laptop) {  
        laptops.add(laptop);  
    }  
  
    public LaptopItemIterator iterator() {  
        return new LaptopItemIterator();  
    }  
  
    class LaptopItemIterator implements Iterator<Laptop> {  
  
        private int currentIndex = 0;  
  
        @Override  
        public boolean hasNext() {  
            return currentIndex < laptops.size();  
        }  
  
        @Override  
        public Laptop next() {  
            Laptop laptop = laptops.get(currentIndex);  
            currentIndex++;  
            return laptop;  
        }  
    }  
}
```

Lớp Main chạy chương trình.

```
public class Main {  
  
    public static void main(String[] args) {  
        Shop shop = new Shop();  
        shop.addItem(new Laptop("Asus", 20000000));  
        shop.addItem(new Laptop("Dell", 15000000));  
        shop.addItem(new Laptop("MSI", 18000000));  
  
        Shop.LaptopItemIterator iterator = shop.iterator();  
        while (iterator.hasNext()) {  
            Laptop item = iterator.next();  
            System.out.println(item);  
        }  
    }  
}
```

4.3. Chạy chương trình

```
adapter.Main ×  
"C:\Program Files\Java\jdk-17.0.5"  
Send: Xin chào  
English receive: Hello  
  
Process finished with exit code 0
```

5. Mẫu thiết kế Memento

5.1. Mô tả bài toán

Một điểm có thể di chuyển tới các vị trí khác nhau trên mặt phẳng hai chiều. Ta cần lưu lại các vị trí đã ghé qua để điểm có thể quay lại các vị trí đó. Với yêu cầu như vậy, chúng ta có thể sử dụng mẫu thiết kế Memento để giải quyết bài toán này.

5.2. Cài đặt

Lớp Originator chứa thông tin một điểm và phương thức save, undo.

```
public class Originator {  
  
    private int x;  
    private int y;  
  
    public Originator(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    public int getX() {  
        return x;  
    }  
  
    public int getY() {  
        return y;  
    }  
  
    public void setX(int x) {  
        this.x = x;  
    }  
  
    public void setY(int y) {  
        this.y = y;  
    }  
  
    public Memento save() {  
        return new Memento(this.x, this.y);  
    }  
  
    public void undo(Memento mem) {  
        this.x = mem.getX();  
        this.y = mem.getY();  
    }  
  
    @Override  
    public String toString() {  
        return "Originator{" +  
            "x=" + x +  
            ", y=" + y +  
            '}';  
    }  
}
```

Lớp Memento bảo vệ chống lại sự truy cập của các đối tượng khác ngoài Originator.

```
public class Memento {  
  
    private final int x;  
    private final int y;  
  
    public Memento(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    public int getX() {  
        return x;  
    }  
  
    public int getY() {  
        return y;  
    }  
  
    @Override  
    public String toString() {  
        return "Memento{" +  
            "x=" + x +  
            ", y=" + y +  
            '}';  
    }  
}
```

Lớp Caretaker chứa các điểm đã lưu và phương thức lưu một điểm, lấy một điểm đã lưu.

```
public class CareTaker {  
  
    private final HashMap<String, Memento> savePointStorage = new HashMap<>();  
  
    public void saveMemento(Memento memento, String savedPointName) {  
        System.out.println("Lưu " + savedPointName + ", " + memento);  
        savePointStorage.put(savedPointName, memento);  
    }  
  
    public Memento getMemento(String savedPointName) {  
        Memento memento = savePointStorage.get(savedPointName);  
        System.out.println("Quay lại " + savedPointName + ", " + memento);  
        return memento;  
    }  
}
```

Lớp Main chạy chương trình

```
public class Main {  
  
    public static void main(String[] args) {  
        CareTaker careTaker = new CareTaker();  
        Originator originator = new Originator(5, 10);  
  
        originator.setX(6);  
        careTaker.saveMemento(originator.save(), "Save 1");  
  
        originator.setY(6);  
        careTaker.saveMemento(originator.save(), "Save 2");  
  
        originator.setX(10);  
        careTaker.saveMemento(originator.save(), "Save 3");  
  
        originator.setX(9);  
        originator.setY(8);  
        careTaker.saveMemento(originator.save(), "Save 4");  
  
        originator.undo(careTaker.getMemento("Save 3"));  
        System.out.println(originator);  
    }  
}
```

5.3. Chạy chương trình

```
memento.Main x  
"C:\Program Files\Java\jdk-17.0.5\bin\java.exe"  
Lưu Save 1, Memento{x=6, y=10}  
Lưu Save 2, Memento{x=6, y=6}  
Lưu Save 3, Memento{x=10, y=6}  
Lưu Save 4, Memento{x=9, y=8}  
Quay lại Save 3, Memento{x=10, y=6}  
Originator{x=10, y=6}  
  
Process finished with exit code 0
```

6. Mẫu thiết kế Template Method

6.1. Mô tả bài toán

Một trang web có các phần Header và Footer giống nhau nên ta cần tái sử dụng đoạn code của hai phần này để tránh trùng lặp code. Với yêu cầu như vậy, chúng ta có thể sử dụng mẫu thiết kế Template Method để giải quyết bài toán này.

6.2. Cài đặt

Lớp PageTemplate là template của website

```
public abstract class PageTemplate {  
  
    protected void showHeader() {  
        System.out.println("<header />");  
    }  
  
    protected void showFooter() {  
        System.out.println("<footer />");  
    }  
  
    protected abstract void showBody();  
  
    public final void showPage() {  
        showHeader();  
        showBody();  
        showFooter();  
    }  
}
```

Lớp HomePage là trang chủ

```
public class HomePage extends PageTemplate {  
  
    @Override  
    protected void showBody() {  
        System.out.println("Trang chủ");  
    }  
}
```

Lớp ProductsPage là trang sản phẩm

```
public class ProductsPage extends PageTemplate {  
  
    @Override  
    protected void showBody() {  
        System.out.println("Trang sản phẩm");  
    }  
}
```

Lớp FullscreenMapPage là trang bản đồ toàn màn hình nên không cần header và footer

```
public class FullscreenMapPage extends PageTemplate {

    @Override
    protected void showHeader() {

    }

    @Override
    protected void showFooter() {

    }

    @Override
    protected void showBody() {
        System.out.println("Bản đồ toàn màn hình");
    }
}
```

Lớp Main chạy chương trình

```
public class Main {

    public static void main(String[] args) {
        PageTemplate homePage = new HomePage();
        homePage.showPage();

        System.out.println("\n-----\n");

        PageTemplate productsPage = new ProductsPage();
        productsPage.showPage();

        System.out.println("\n-----\n");

        PageTemplate fullscreenMapPage = new FullscreenMapPage();
        fullscreenMapPage.showPage();
    }
}
```

6.3. Chạy chương trình

```
Main ×
"C:\Program Files\Java\jdk-17.0.5'
<header />
Trang chủ
<footer />

-----

<header />
Trang sản phẩm
<footer />

-----

Bản đồ toàn màn hình

Process finished with exit code 0
```