Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине «СПП»

Выполнил
студент 2 курса
группы ПО-5:
Филатов Д.Д.
Проверил:
Крощенко А.А.

Брест, 2020

Вариант 8

Цель: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java.

Задание 1:

Торговый автомат с возможностью выдачи любого выбранного товара (шоколадные батончики, чипсы, пакетированные соки и т.д.)

Абстрактная фабрика — порождающий шаблон проектирования, предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов, не специфицируя их конкретных классов. Шаблон реализуется созданием абстрактного класса Factory, который представляет собой интерфейс для создания компонентов системы (например, для оконного интерфейса он может создавать окна и кнопки). Затем пишутся классы, реализующие этот интерфейс.

Код программы:

Main.java

```
package com.company;
import java.math.BigDecimal;

public class Main {
    public static void main(String[] args) {
        Factory chipsFactory = new ChipsFactory();
        chipsFactory.createProduct().printInformation();
        Factory chocolateBarsFactory = new ChocolateBarsFactory();
        chocolateBarsFactory.createProduct().printInformation();
        Factory packagedJuicesFactory = new PackagedJuicesFactory();
        packagedJuicesFactory.createProduct().printInformation();
    }
}

abstract class Product {
    private BigDecimal price;
    private double weight;
    public Product() {
    }
    public Product(BigDecimal price, double weight) {
        this.price = price;
        this.weight = weight;
    }
    public BigDecimal getPrice() {
        return price;
    }
    public double getWeight() {
        return weight;
    }
    public abstract void printInformation();
}

class Chips extends Product {
    public Chips() {
        super(new BigDecimal("3.25"), 130);
    }
    @Override
    public void printInformation() {
        System.out.println("Chips: " + super.getWeight() + "г., " + super.getPrice() +
                "р.");
    }
}

class ChocolateBars extends Product {
```

```
    public ChocolateBars() {
        super(new BigDecimal("3.19"), 80);
    }
    @Override
    public void printInformation() {
        System.out.println("Chocolate bars: " + super.getWeight() + "г., " +
                super.getPrice() + "р.");
    }
}

class PackagedJuices extends Product {
    public PackagedJuices() {
        super(new BigDecimal("0.79"), 150);
    }
    @Override
    public void printInformation() {
        System.out.println("Packaged juices: " + super.getWeight() + "г., " +
                super.getPrice() + "р.");
    }
}

interface Factory {
    Product createProduct();
}

class ChipsFactory implements Factory {
    @Override
    public Chips createProduct() {
        return new Chips();
    }
}

class ChocolateBarsFactory implements Factory {
    @Override
    public ChocolateBars createProduct() {
        return new ChocolateBars();
    }
}

class PackagedJuicesFactory implements Factory {
    @Override
    public PackagedJuices createProduct() {
        return new PackagedJuices();
    }
}
```

Результат выполнения:

```
Chips: 130.0г., 3.25р.
Chocolate bars: 80.0г., 3.19р.
Packaged juices: 150.0г., 0.79р.
```

Задание 2:

ДУ телевизора. Реализовать иерархию телевизоров для конкретных производителей и иерархию средств дистанционного управления. Телевизоры должны иметь присущие им атрибуты и функции. ДУ имеет набор функций для изменения текущего канала, увеличения/уменьшения громкости, включения/выключения телевизора и т.д. Эти функции должны отличаться для различных устройств ДУ.

Мост — структурный шаблон проектирования, используемый в проектировании программного обеспечения чтобы разделять абстракцию и реализацию так, чтобы они могли

изменяться независимо. Шаблон мост использует инкапсуляцию, агрегирование и может использовать наследование для того, чтобы разделить ответственность между классами.

Код программы:

Main.java

```
package com.company;

public class Main {
    public static void main(String[] args) {
        SamsungTV samsungTV = new SamsungTV(50, 500, true);
        samsungTV.printCharacteristic();
        SamsungRemote samsungRemote = new SamsungRemote(samsungTV);
        samsungRemote.turnOn();
        samsungRemote.increaseVolume();
        LGTV lgtv = new LGTV(48, 320, false);
        lgtv.printCharacteristic();
        LGRemote lgRemote = new LGRemote(lgtv);
        lgRemote.turnOn();
        lgRemote.switchChannel(7);
        lgRemote.turnOff();
    }
}

interface Remote {
    void turnOn();
    void turnOff();
    void switchChannel(int chanelId);
    void increaseVolume();
    void increaseDecrease();
}

interface TV {
    void printCharacteristic();
    void turnOn();
    void turnOff();
}

class LGRemote implements Remote {
    private LGTV tv;
    public LGRemote(LGTV tv) {
        this.tv = tv;
    }
    @Override
    public void turnOn() {
        System.out.println("TV LG is going to turn on by remote.");
    }
    @Override
    public void turnOff() {
        System.out.println("TV LG is going to turn off by remote.");
    }
    @Override
    public void switchChannel(int chanelId) {
        System.out.println("TV LG is going to switch channel on '" + chanelId + "' by
remote.");
    }
    @Override
    public void increaseVolume() {
        System.out.println("TV LG is going to increase volume by remote.");
    }
    @Override
    public void increaseDecrease() {
        System.out.println("TV LG is going to decrease volume by remote.");
    }
}

class LGTV implements TV {
```

```
    private double diagonalSize;
    private int brightness;
    private boolean isWiFiExist;
    public LGTV(double diagonalSize, int brightness, boolean isWiFiExist) {
        this.diagonalSize = diagonalSize;
        this.brightness = brightness;
        this.isWiFiExist = isWiFiExist;
    }
    @Override
    public void printCharacteristic() {
        System.out.println("TV LG:\n\tdiagonal size: " + diagonalSize
                + "\n\tbrightness: " + brightness
                + "\n\tisWiFiExist: " + isWiFiExist);
    }
    @Override
    public void turnOn() {
        System.out.println("TV LG is going to turn on.");
    }
    @Override
    public void turnOff() {
        System.out.println("TV LG is going to turn off.");
    }
}

class SamsungRemote implements Remote {
    private SamsungTV samsungTV;
    public SamsungRemote(SamsungTV samsungTV) {
        this.samsungTV = samsungTV;
    }
    @Override
    public void turnOn() {
        System.out.println("TV Samsung is going to turn on by remote.");
    }
    @Override
    public void turnOff() {
        System.out.println("TV Samsung is going to turn off by remote.");
    }
    @Override
    public void switchChannel(int chanelId) {
        System.out.println("TV Samsung is going to switch channel on '" + chanelId + "'
by remote.");
    }
    @Override
    public void increaseVolume() {
        System.out.println("TV Samsung is going to increase volume by remote.");
    }
    @Override
    public void increaseDecrease() {
        System.out.println("TV Samsung is going to decrease volume by remote.");
    }
}

class SamsungTV implements TV {
    private double diagonalSize;
    private int brightness;
    private boolean isWiFiExist;
    public SamsungTV(double diagonalSize, int brightness, boolean isWiFiExist) {
        this.diagonalSize = diagonalSize;
        this.brightness = brightness;
        this.isWiFiExist = isWiFiExist;
    }
    @Override
    public void printCharacteristic() {
        System.out.println("TV Samsung:\n\tdiagonal size: " + diagonalSize
                + "\n\tbrightness: " + brightness
                + "\n\tisWiFiExist: " + isWiFiExist);
    }
```

```
    @Override
    public void turnOn() {
        System.out.println("TV Samsung is going to turn on.");
    }
    @Override
    public void turnOff() {
        System.out.println("TV Samsung is going to turn off.");
    }
}
```

Результат выполнения:

```
TV Samsung:
    diagonal size: 50.0
    brightness: 500
    isWiFiExist: true
TV Samsung is going to turn on by remote.
TV Samsung is going to increase volume by remote.
TV LG:
    diagonal size: 48.0
    brightness: 320
    isWiFiExist: false
TV LG is going to turn on by remote.
TV LG is going to switch channel on '7' by remote.
TV LG is going to turn off by remote.
```

Задание 3:

Вспомогательная библиотека для работы с текстовыми файлами. Должны быть предусмотрены следующие функции: чтение одного или нескольких файлов, изменение файла(ов), отмена последней выполненной операции (одной в случае простой единичной операции или нескольких в случае сложной операции), последовательное выполнение нескольких операций.

Стратегия — это поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы.

Код программы:

Main.java

```
package com.company;

public class Main {
    public static void main(String[] args) {
        Keyboard keyboardButton1 = new Keyboard(new Digital("4"));
        Keyboard keyboardButton2 = new Keyboard(new Arithmetic("+"));
        Keyboard keyboardButton3 = new Keyboard(new Customize("6"));
        System.out.println(keyboardButton1.getSymbol());
        System.out.println(keyboardButton2.getSymbol());
        System.out.println(keyboardButton3.getSymbol());
        System.out.println();
        keyboardButton1 = customize(keyboardButton1);
        keyboardButton2 = customize(keyboardButton2);
        keyboardButton3 = customize(keyboardButton3);
        System.out.println(keyboardButton1.getSymbol());
        System.out.println(keyboardButton2.getSymbol());
        System.out.println(keyboardButton3.getSymbol());
    }
    private static Keyboard customize(Keyboard keyboardButton) {
        if(keyboardButton.isCustomized()) {
```

```
            keyboardButton = new Keyboard(new Customize("*"));
        }
        return keyboardButton;
    }
}

interface Button {
    String getSymbol();
    boolean isCustomized();
}

class Arithmetic implements Button {
    private String symbol;
    Arithmetic(String symbol) {
        this.symbol = symbol;
    }
    @Override
    public String getSymbol() {
        return "Arithmetic button " + symbol;
    }
    @Override
    public boolean isCustomized() {
        return false;
    }
}

class Customize implements Button {
    private String symbol;
    Customize(String symbol) {
        this.symbol = symbol;
    }
    @Override
    public String getSymbol() {
        return "Custom button " + symbol;
    }
    @Override
    public boolean isCustomized() {
        return true;
    }
}

class Digital implements Button {
    private String symbol;
    Digital(String symbol) {
        this.symbol = symbol;
    }
    @Override
    public String getSymbol() {
        return "Digital button " + symbol;
    }
    @Override
    public boolean isCustomized() {
        return false;
    }
}

class Keyboard {
    Button button;
    Keyboard(Button button) {
        this.button = button;
    }
    String getSymbol() {
        return button.getSymbol();
    }
    boolean isCustomized() {
        return button.isCustomized();
```

```
    }
}
```

Результат выполнения:

```
Digital button 4
Arithmetic button +
Custom button 6

Digital button 4
Arithmetic button +
Custom button *
```

Вывод: Освоила возможности языка программирования Java в построении графических приложений.