Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6

По дисциплине: «Современные платформы программирования»

Выполнил:
Студент 3 курса
Группы ПО-8
Шлыков А.Л.
Проверил:
Крощенко А.А.

Брест 2024

Цель работы:

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

Задание 1

“Фабричный метод”

1) Кофе-автомат с возможностью создания различных кофейных напитков (предусмотреть 5 классов наименований)

```
abstract class Coffee {
    abstract String prepare();
}


class Espresso extends Coffee {
    String prepare() {
        return "Espresso is ready!";
    }
}

class Latte extends Coffee {
    String prepare() {
        return "Latte is ready!";
    }
}

class Cappuccino extends Coffee {
    String prepare() {
        return "Cappuccino is ready!";
    }
}

class Americano extends Coffee {
    String prepare() {
        return "Americano is ready!";
    }
}

class Mocha extends Coffee {
    String prepare() {
        return "Mocha is ready!";
    }
}

class CoffeeMachine {
    String makeCoffee(Coffee coffee) {
        return coffee.prepare();
    }
}
class CoffeeFactory {
    static Coffee createCoffee(String type) {
        switch (type) {
            case "Espresso":
                return new Espresso();
            case "Latte":
                return new Latte();
            case "Cappuccino":
                return new Cappuccino();
```

```
            case "Americano":
                return new Americano();
            case "Mocha":
                return new Mocha();
            default:
                throw new IllegalArgumentException("Unknown coffee type: " + type);
        }
    }
}

public class task1{
    public static void main(String[] args) {
        CoffeeMachine coffeeMachine = new CoffeeMachine();
        System.out.println(coffeeMachine.makeCoffee(CoffeeFactory.createCoffee("Espresso")));
        System.out.println(coffeeMachine.makeCoffee(CoffeeFactory.createCoffee("Latte")));
        System.out.println(coffeeMachine.makeCoffee(CoffeeFactory.createCoffee("Cappuccino")));
        System.out.println(coffeeMachine.makeCoffee(CoffeeFactory.createCoffee("Americano")));
        System.out.println(coffeeMachine.makeCoffee(CoffeeFactory.createCoffee("Mocha")));
    }
}
```

```
[Running] cd "/Users/harweast/Documents/3kurs/sem2/SPP/laba6/6/src/" && javac task1.java && java task1
Espresso is ready!
Latte is ready!
Cappuccino is ready!
Americano is ready!
Mocha is ready!

[Done] exited with code=0 in 0.429 seconds
```

Задание 2

паттерн проектирования “Адаптер”

```
class DigitalClock {
    private int hours;
    private int minutes;
    private int seconds;

    public DigitalClock(int hours, int minutes, int seconds) {
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
    }

    public String displayTime() {
        return hours + ":" + minutes + ":" + seconds;
    }
}

class AnalogClock {
    private int degrees;

    public AnalogClock(int degrees) {
        this.degrees = degrees;
    }

    public int displayDegrees() {
        return degrees;
    }
}
```

class

```
[Running] cd "/Users/harweast/Documents/3kur
3:0:00

[Done] exited with code=0 in 0.452 seconds
```

```
AnalogClockAdapter extends DigitalClock {
    private AnalogClock analogClock;

    public AnalogClockAdapter(AnalogClock analogClock) {
        super(0, 0, 0);
        this.analogClock = analogClock;
    }
    @Override
    public String displayTime() {
        int degrees = analogClock.displayDegrees();
        int hours = degrees / 30;
        int minutes = (degrees % 30) * 2;
        return hours + ":" + minutes + ":00";
    }
}
public class task2 {
    public static void main(String[] args) {
        AnalogClock analogClock = new AnalogClock(90);
        AnalogClockAdapter adapter = new AnalogClockAdapter(analogClock);
        System.out.println(adapter.displayTime());
    }
}
```

В коде используется шаблон проектирования Адаптер (Adapter).
Класс AnalogClockAdapter служит адаптером, который позволяет работать с объектами класса AnalogClock так, как будто это объекты класса DigitalClock. Это достигается за счет наследования AnalogClockAdapter от DigitalClock и использования экземпляра AnalogClock внутри адаптера.

Задание 3

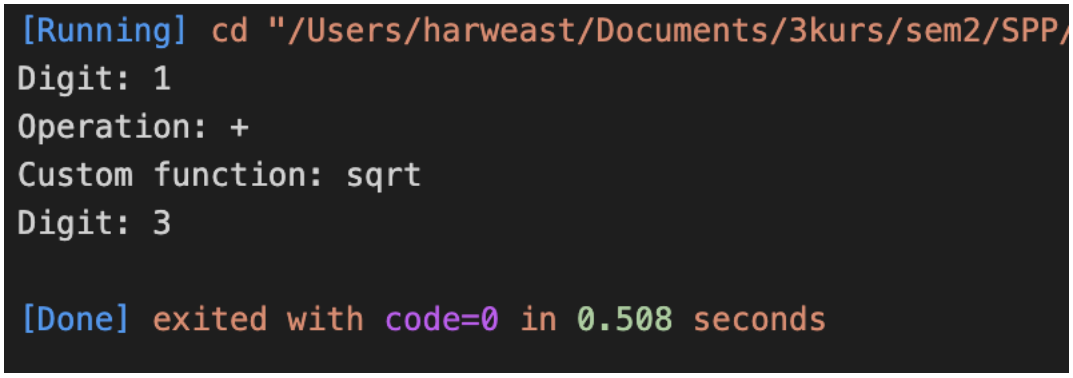
1) Проект «Клавиатура настраимаемого калькулятора». Цифровые и арифметические кнопки имеют фиксированную функцию, а остальные могут менять своё назначение.
паттерн проектирования «**Стратегия**»

```
interface ButtonFunction {
    void execute();
}

class DigitButtonFunction implements ButtonFunction {
    private int digit;

    public DigitButtonFunction(int digit) {
        this.digit = digit;
    }
```

```
    @Override
    public void execute() {
        System.out.println("Digit: " + digit);
    }
}

class ArithmeticButtonFunction implements ButtonFunction {
    private char operation;

    public ArithmeticButtonFunction(char operation) {
        this.operation = operation;
    }

    @Override
    public void execute() {
        System.out.println("Operation: " + operation);
    }
}

class CustomButtonFunction implements ButtonFunction {
    private String function;

    public CustomButtonFunction(String function) {
        this.function = function;
    }

    @Override
    public void execute() {
        System.out.println("Custom function: " + function);
    }
}

class Button {
    private ButtonFunction function;

    public Button(ButtonFunction function) {
        this.function = function;
    }

    public void press() {
        function.execute();
    }

    public void setFunction(ButtonFunction function) {
        this.function = function;
    }
}

public class task3 {
    public static void main(String[] args) {
        Button button1 = new Button(new DigitButtonFunction(1));
        Button button2 = new Button(new ArithmeticButtonFunction('+'));
        Button button3 = new Button(new CustomButtonFunction("sqrt"));

        button1.press();
        button2.press();
        button3.press();

        button3.setFunction(new DigitButtonFunction(3));
        button3.press();
```

```
    }
}
```

Интерфейс ButtonFunction представляет собой стратегию, которая определяет метод execute(). Классы DigitButtonFunction, ArithmeticButtonFunction и CustomButtonFunction реализуют этот интерфейс и предоставляют конкретные реализации алгоритма.



Вывод: приобрёл навыки применения паттернов проектирования при решении практических задач с использованием языка Java