МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ

по лабораторной работе №6

Выполнила:
студентка 3 курса
группы ПО-9
Бердникова В.А.

Проверил:
Крощенко А.А.

Брест 2024

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

Вариант 1

Задание 1

Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы. Кофе-автомат с возможностью создания различных кофейных напитков (предусмотреть 5 классов наименований). ***Абстрактная фабрика***

Входные данные:

```
Coffee americano = americanoFactory.createCoffee();
americano.prepare();

Coffee latte = latteFactory.createCoffee();
latte.prepare();

Coffee cappuccino = cappuccinoFactory.createCoffee();
cappuccino.prepare();

Coffee espresso = espressoFactory.createCoffee();
espresso.prepare();

Coffee macchiato = macchiatoFactory.createCoffee();
macchiato.prepare();
```

Выходные данные:

```
Американо готово!
Латте готово!
Капуччино готово!
Эспрессо готово!
Макиато готово!
```

Код программы:

Task1.java

```
interface Coffee {
        void prepare();
    }

    class Americano implements Coffee {
        @Override
        public void prepare() {
            System.out.println("Американо готово!");
        }
    }

    class Latte implements Coffee {
        @Override
        public void prepare() {
            System.out.println("Латте готово!");
        }
```

```
}

class Cappuccino implements Coffee {
    @Override
    public void prepare() {
        System.out.println("Капуччино готово!");
    }
}

class Espresso implements Coffee {
    @Override
    public void prepare() {
        System.out.println("Эспрессо готово!");
    }
}

class Macchiato implements Coffee {
    @Override
    public void prepare() {
        System.out.println("Макиато готово!");
    }
}

interface CoffeeAbstractFactory {
    Coffee createCoffee();
}

class AmericanoFactory implements CoffeeAbstractFactory {
    @Override
    public Coffee createCoffee() {
        return new Americano();
    }
}

class LatteFactory implements CoffeeAbstractFactory {
    @Override
    public Coffee createCoffee() {
        return new Latte();
    }
}

class CappuccinoFactory implements CoffeeAbstractFactory {
    @Override
    public Coffee createCoffee() {
        return new Cappuccino();
    }
}

class EspressoFactory implements CoffeeAbstractFactory {
    @Override
    public Coffee createCoffee() {
        return new Espresso();
    }
}

class MacchiatoFactory implements CoffeeAbstractFactory {
    @Override
    public Coffee createCoffee() {
        return new Macchiato();
    }
}

public class Task1 {
    public static void main(String[] args) {
```

```
            CoffeeAbstractFactory americanoFactory = new AmericanoFactory();
            CoffeeAbstractFactory latteFactory = new LatteFactory();
            CoffeeAbstractFactory cappuccinoFactory = new CappuccinoFactory();
            CoffeeAbstractFactory espressoFactory = new EspressoFactory();
            CoffeeAbstractFactory macchiatoFactory = new MacchiatoFactory();

            Coffee americano = americanoFactory.createCoffee();
            americano.prepare();

            Coffee latte = latteFactory.createCoffee();
            latte.prepare();

            Coffee cappuccino = cappuccinoFactory.createCoffee();
            cappuccino.prepare();

            Coffee espresso = espressoFactory.createCoffee();
            espresso.prepare();

            Coffee macchiato = macchiatoFactory.createCoffee();
            macchiato.prepare();
        }
    }
```

Задание 2

Проект «Часы». В проекте должен быть реализован класс, который дает возможность пользоваться часами со стрелками так же, как и цифровыми часами. В классе «Часы со стрелками» хранятся повороты стрелок. ***Адаптер***

Входные данные:

```
SimpleDigitalClock digitalClock = new SimpleDigitalClock();
AnalogClock analogClock = new AnalogClock();

analogClock.setHourRotation(90);
analogClock.setMinuteRotation(180);
```

Выходные данные:

```
Using digital clock directly:
Digital Clock: displaying time

Using analog clock via adapter:
Digital Time: 3:30
```

Код программы:

Task2.java

```
interface DigitalClock {
    void displayTime();}

class SimpleDigitalClock implements DigitalClock {
    @Override
    public void displayTime() {
        System.out.println("Digital Clock: displaying time");
```

```
    }
}

class AnalogClock {
    private int hourRotation;
    private int minuteRotation;

    public AnalogClock() {
        this.hourRotation = 0;
        this.minuteRotation = 0;
    }

    public void setHourRotation(int rotation) {
        this.hourRotation = rotation;
    }

    public void setMinuteRotation(int rotation) {
        this.minuteRotation = rotation;
    }

    public int getHourRotation() {
        return hourRotation;
    }

    public int getMinuteRotation() {
        return minuteRotation;
    }
}

class AnalogToDigitalAdapter implements DigitalClock {
    private AnalogClock analogClock;

    public AnalogToDigitalAdapter(AnalogClock analogClock) {
        this.analogClock = analogClock;
    }

    @Override
    public void displayTime() {
        int hourRotation = analogClock.getHourRotation();
        int minuteRotation = analogClock.getMinuteRotation();

        int hours = hourRotation / 30;
        int minutes = (minuteRotation / 6) % 60;

        System.out.println("Digital Time: " + hours + ":" + minutes);
    }
}

public class Task2 {
    public static void main(String[] args) {
        SimpleDigitalClock digitalClock = new SimpleDigitalClock();
        AnalogClock analogClock = new AnalogClock();

        analogClock.setHourRotation(90);
        analogClock.setMinuteRotation(180);

        System.out.println("Using digital clock directly:");
        digitalClock.displayTime();

        System.out.println("\nUsing analog clock via adapter:");
        DigitalClock adapter = new AnalogToDigitalAdapter(analogClock);
        adapter.displayTime();
    }
}
```

Задание 3

Проект «Клавиатура настраиваемого калькулятора». Цифровые и арифметические кнопки имеют фиксированную функцию, а остальные могут менять своё назначение. ***Стратегия***

Входные данные:

```
Button digitButton = new Button(new DigitButtonClick(5));
Button additionButton = new Button(new AdditionButtonClick());
CustomButton customButton = new CustomButton();
customButton.setStrategy(() -> System.out.println("Новое действие"));
```

Выходные данные:

```
Нажата цифровая кнопка: 5
Нажата кнопка сложения
Действие по умолчанию
Новое действие
```

Код программы:

Task3.java

```
interface ButtonStrategy {
    void onClick();
}

class Button {
    private ButtonStrategy strategy;

    public Button(ButtonStrategy strategy) {
        this.strategy = strategy;
    }

    public void onClick() {
        strategy.onClick();
    }

    public void setStrategy(ButtonStrategy strategy) {
        this.strategy = strategy;
    }
}

class DigitButtonClick implements ButtonStrategy {
    private int digit;

    public DigitButtonClick(int digit) {
        this.digit = digit;
    }

    @Override
    public void onClick() {
        System.out.println("Нажата цифровая кнопка: " + digit);
    }
}

class AdditionButtonClick implements ButtonStrategy {
    @Override
    public void onClick() {
```

```
        System.out.println("Нажата кнопка сложения");
    }
}

class SubtractionButtonClick implements ButtonStrategy {
    @Override
    public void onClick() {
        System.out.println("Нажата кнопка вычитания");
    }
}

class MultiplicationButtonClick implements ButtonStrategy {
    @Override
    public void onClick() {
        System.out.println("Нажата кнопка умножения");
    }
}

class DivisionButtonClick implements ButtonStrategy {
    @Override
    public void onClick() {
        System.out.println("Нажата кнопка деления");
    }
}

class CustomButton {
    private ButtonStrategy strategy;

    public CustomButton() {
        this.strategy = () -> {
            System.out.println("Действие по умолчанию");
        };
    }

    public void setStrategy(ButtonStrategy strategy) {
        this.strategy = strategy;
    }

    public void onClick() {
        strategy.onClick();
    }
}

public class Task3 {
    public static void main(String[] args) {
        Button digitButton = new Button(new DigitButtonClick(5));
        Button additionButton = new Button(new AdditionButtonClick());
        digitButton.onClick();
        additionButton.onClick();

        CustomButton customButton = new CustomButton();
        customButton.onClick();
        customButton.setStrategy(() -> System.out.println("Новое действие"));
        customButton.onClick();
    }
}
```

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