МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №6

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Пищик А.В

Проверил:

Крощенко А. А.

Брест 2021

Цель: приобрести навыки применения паттернов проектирования при решении

практических задач с использованием языка Java.

**Задание 1:**

Кофе-автомат с возможностью создания различных кофейных напитков (предусмотреть 5

классов наименований).

**Паттерн: Абстрактная фабрика.**

**Текст программы:**

import java.util.ArrayList;

import java.util.List;

public class task1lab6 {

public static void main(String[] args) {

CoffeeMachine macchiatoFactory = new MacchiattoFactory();

CoffeeMachine latteFactory = new LatteFactory();

CoffeeMachine cappuccinoFactory = new CappuccinoFactory();

CoffeeMachine espressoFactory = new EspressoFactory();

CoffeeMachine americanoFactory = new AmericanoFactory();

Coffee macchiato = macchiatoFactory.makeCoffee();

Coffee latte = latteFactory.makeCoffee();

Coffee cappuccino = cappuccinoFactory.makeCoffee();

Coffee espresso = espressoFactory.makeCoffee();

Coffee americano = americanoFactory.makeCoffee();

System.out.println(macchiato);

System.out.println(latte);

System.out.println(cappuccino);

System.out.println(espresso);

System.out.println(americano);

}

}

interface CoffeeMachine {

Coffee makeCoffee();

}

class MacchiattoFactory implements CoffeeMachine {

@Override

public Coffee makeCoffee() {

return new Machiatto();

}

}

class LatteFactory implements CoffeeMachine {

@Override

public Coffee makeCoffee() {

return new Latte();

}

}

class CappuccinoFactory implements CoffeeMachine {

@Override

public Coffee makeCoffee() {

return new Cappuccino();

}

}

class EspressoFactory implements CoffeeMachine {

@Override

public Coffee makeCoffee() {

return new Espresso();

}

}

class AmericanoFactory implements CoffeeMachine {

@Override

public Coffee makeCoffee() {

return new Americano();

}

}

abstract class Coffee {

String name;

public List<String> ingredients = new ArrayList<>();

public String toString() {

return name + " - Ingredients: " + ingredients;

}

}

class Machiatto extends Coffee {

Machiatto() {

name = "Machiatto";

ingredients.add("espresso");

ingredients.add("whipped milk");

}

}

class Latte extends Coffee {

Latte() {

name = "Latte";

ingredients.add("espresso");

ingredients.add("milk");

ingredients.add("whipped milk");

}

}

class Cappuccino extends Coffee {

Cappuccino() {

name = "Cappuccino";

ingredients.add("espresso");

ingredients.add("milk");

ingredients.add("whipped milk");

}

}

class Espresso extends Coffee {

Espresso() {

name = "Espresso";

ingredients.add("espresso");

}

}

class Americano extends Coffee {

Americano() {

name = "Americano";

ingredients.add("espresso");

ingredients.add("water");

}

}

**Результат выполнения:**

Machiatto - Ingredients: [espresso, whipped milk]

Latte - Ingredients: [espresso, milk, whipped milk]

Cappuccino - Ingredients: [espresso, milk, whipped milk]

Espresso - Ingredients: [espresso]

Americano - Ingredients: [espresso, water]

**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.

**Паттерн: Адаптер.**

**Текст программы:**

public class task2lab6 {

public static void main(String[] args) {

DigitalClock clock = new DialToDigital(new DialWatch(300, 180));

clock.showTime();

}

}

interface DigitalClock {

void showTime();

}

class DigitalWatch implements DigitalClock {

private Integer hours;

private Integer minutes;

DigitalWatch(Integer hours, Integer minutes) {

this.hours = hours;

this.minutes = minutes;

}

@Override

public void showTime() {

if (minutes >= 10)

System.out.println(hours.toString() + ":" + minutes.toString());

else

System.out.println(hours.toString() + ":0" + minutes.toString());

}

}

class DialWatch {

private Integer hoursDegrees;

private Integer minutesDegrees;

DialWatch(Integer hoursDegrees, Integer minutesDegrees) {

this.hoursDegrees = hoursDegrees;

this.minutesDegrees = minutesDegrees;

}

public Integer getHoursDegrees() {

return hoursDegrees;

}

public Integer getMinutesDegrees() {

return minutesDegrees;

}

}

class DialToDigital implements DigitalClock {

private DialWatch dialWatch;

DialToDigital(DialWatch dialWatch) {

this.dialWatch = dialWatch;

}

@Override

public void showTime() {

if (dialWatch.getHoursDegrees() >= 30 && dialWatch.getMinutesDegrees() != 360) {

DigitalClock digitalWatch = new DigitalWatch(dialWatch.getHoursDegrees() / 30, dialWatch.getMinutesDegrees() / 6);

digitalWatch.showTime();

} else if (dialWatch.getMinutesDegrees() == 360) {

DigitalClock digitalWatch = new DigitalWatch(dialWatch.getHoursDegrees() / 30 + 1, 0);

digitalWatch.showTime();

} else {

DigitalClock digitalWatch = new DigitalWatch(12, dialWatch.getMinutesDegrees() / 6);

digitalWatch.showTime();

}

}

}

**Результат выполнения:**

10:30

**Задание 3:**

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

**Паттерн: Стратегия.**

**Текст программы:**

public class task3lab6 {

public static void main(String[] args) {

Keyboard keyboardButton1 = new Keyboard(new Digital("2"));

Keyboard keyboardButton2 = new Keyboard(new Arithmetic("-"));

Keyboard keyboardButton3 = new Keyboard(new Customize("3"));

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 Keyboard {

Button button;

Keyboard(Button button) {

this.button = button;

}

String getSymbol() {

return button.getSymbol();

}

boolean isCustomized() {

return button.isCustomized();

}

}

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 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;

}

}

**Результат выполнения:**

Digital button 2

Arithmetic button -Custom button 3

Digital button 2

Arithmetic button -

Custom button \*

Вывод: В ходе выполненной работы приобрел навыки применения паттернов

проектирования при решении практических задач с использованием языка Java.