Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине «Современные платформы программирования»

Выполнила:

Студентка 3 курса

Группы ПО-3

Пивчик В.Г.

Проверил:

Крощенко А.А.

Брест 2020 г.

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

**Вариант 9**

Постановка задачи:

Задание 1:

Проект «Бургер-закусочная». Реализовать возможность формирования заказа из определенных позиций (тип бургера (веганский, куриный и т.д.)), напиток (холодный – пепси, кока-кола и т.д.; горячий – кофе, чай и т.д.), тип упаковки – с собой, на месте. Должна формироваться итоговая стоимость заказа.

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.

Задание 3:

Шифрование текстового файла. Реализовать класс-шифровщик текстового файла с поддержкой различных алгоритмов шифрования. Возможные варианты шифрования: удаление всех гласных букв из текста, изменение букв текста на буквы, получаемые фиксированным сдвигом из алфавита (например, шифром буквы а будет являться буква д для сдвига 4 и т.д.), применение операции исключающее или с заданным ключом.

**Ход работы**

**Текст программы:**

**Задание 1**

**Код программы**

**Main**package com.company;

public class Main {

public static void main(String[] args) {

Order myOrder = new Order.Builder()

.withBurger(BurgerType.CHICKEN)

.withBurger(BurgerType.CHICKEN)

.withDrink(DrinkType.COLA)

.withPacking(PackingType.TAKEAWAY)

.build();

System.out.println(myOrder);

Order myOrderTwo = new Order.Builder()

.withBurger(BurgerType.VEGAN)

.withDrink(DrinkType.COFFEE)

.withPacking(PackingType.IN)

.build();

System.out.println(myOrderTwo); }

}

**BurgerType**package com.company;

public enum BurgerType {

VEGAN(2.99),

CHICKEN(3.99);

private Double price;

BurgerType(Double price) {

this.price = price;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**DrinkType**package com.company;

public enum DrinkType {

TEA(0.99),

COFFEE(1.99),

COLA(1.29),

PEPSI(1.29);

private Double price;

DrinkType(Double price) {

this.price = price;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**PackingType**package com.company;

public enum PackingType {

TAKEAWAY(0.49),

IN(0.00);

private Double price;

PackingType(Double price) {

this.price = price;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**Order**

package com.company;

import java.util.ArrayList;

public class Order {

private ArrayList<BurgerType> burgers;

private ArrayList<DrinkType> drinks;

private PackingType packing;

private Double totalCost;

public Order() {

this.burgers = new ArrayList<>();

this.drinks = new ArrayList<>();

this.totalCost = 0.00;

}

public ArrayList<BurgerType> getBurgers() {

return burgers;

}

public ArrayList<DrinkType> getDrinks() {

return drinks;

}

public PackingType getPacking() {

return packing;

}

public Double getTotalCost() {

return totalCost;

}

public static class Builder {

private Order newOrder;

public Builder() {

newOrder = new Order();

}

public Builder withBurger(BurgerType burger) {

newOrder.burgers.add(burger);

return this;

}

public Builder withDrink(DrinkType drink) {

newOrder.drinks.add(drink);

return this;

}

public Builder withPacking(PackingType packing) {

newOrder.packing = packing;

return this;

}

public Order build() {

for (BurgerType burger : newOrder.burgers) {

newOrder.totalCost += burger.getPrice();

}

for (DrinkType drink : newOrder.drinks) {

newOrder.totalCost += drink.getPrice();

}

newOrder.totalCost += newOrder.packing.getPrice();

return newOrder;

}

}

@Override

public String toString() {

return "Task1.Order{" +

"burgers=" + burgers +

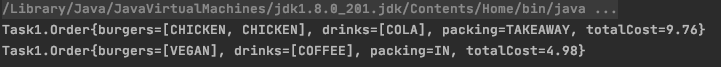
", drinks=" + drinks +

", packing=" + packing +

", totalCost=" + totalCost + '}';}

}

**Рисунок с результатом работы программы**

****

**Задание 2**

**Код программы**

**Main**

package com.company;

public class Main {

public static void main(String[] args) {

DigitalClock digitalTime = new AnalogClockAdapter(new AnalogClock(300, 180));

digitalTime.showTime();

}

}

**DigitalClock**package com.company;

public interface DigitalClock {

void showTime();

}

**DigitalClockImpl**package com.company;

public class DigitalClockImpl implements DigitalClock {

private Integer hours;

private Integer minutes;

DigitalClockImpl(Integer hours, Integer minutes) {

this.hours = hours;

this.minutes = minutes;

}

@Override

public void showTime() {

if (minutes >= 10) {

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

}

else {

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

}

}

}

**AnalogClock**package com.company;

public class AnalogClock {

private Integer hourDegrees;

private Integer minuteDegrees;

AnalogClock(Integer hoursDegrees, Integer minutesDegrees) {

this.hourDegrees = hoursDegrees;

this.minuteDegrees = minutesDegrees;

}

public Integer getHourDegrees() {

return hourDegrees;

}

public Integer getMinuteDegrees() {

return minuteDegrees;

}

}

**AnalogClockAdapter**package com.company;

public class AnalogClockAdapter implements DigitalClock{

private AnalogClock analogClock;

AnalogClockAdapter(AnalogClock analogClock) {

this.analogClock = analogClock;

}

@Override

public void showTime() {

if (analogClock.getHourDegrees() >= 30 && analogClock.getMinuteDegrees() != 360) {

DigitalClock digitalClock = new DigitalClockImpl(

analogClock.getHourDegrees() / 30 ,

analogClock.getMinuteDegrees() / 6

);

digitalClock.showTime();

} else if (analogClock.getMinuteDegrees() == 360) {

DigitalClock digitalClock = new DigitalClockImpl(

analogClock.getHourDegrees() / 30 + 1,

0

);

digitalClock.showTime();

} else {

DigitalClock digitalClock = new DigitalClockImpl(

12,

analogClock.getMinuteDegrees() / 6

);

digitalClock.showTime();

}

}

}

**Рисунок с результатом работы программы**

****

**Задание 3**

**Код программы**

**Main**

package com.company;

import java.io.IOException;

public class Main {

public static void main(String[] args) throws IOException {

Encryptor encryptorOne = new Encryptor(new DeleteLetterAlgorithm());

encryptorOne.encryptFile("/Users/valeriadmitruk/Desktop/SPP/lab63/File.txt", "deleteletter.txt");

Encryptor encryptorTwo = new Encryptor(new MoveLetterAlgorithm());

encryptorTwo.encryptFile("/Users/valeriadmitruk/Desktop/SPP/lab63/File.txt", "moveletter.txt");

Encryptor encryptorThree = new Encryptor(new XorAlgorithm());

encryptorThree.encryptFile("/Users/valeriadmitruk/Desktop/SPP/lab63/File.txt", "xorletter.txt");

System.out.println("Encryption is done. Please check files.");

}

}

**EncryptorAlgorithm**

package com.company;

public interface EncryptionAlgorithm {

String encryptData(String plainText);

}

**Encryptor**package com.company;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

public class Encryptor {

private EncryptionAlgorithm algorithm;

public Encryptor(EncryptionAlgorithm algorithm) {

this.algorithm = algorithm;

}

public void encryptFile(String filePath, String outputFile) throws IOException {

List<String> strings = Files.readAllLines(Paths.get(filePath));

List<String> encryptedStrings = new ArrayList<>();

for (String string : strings) {

encryptedStrings.add(algorithm.encryptData(string));

}

Files.write(Paths.get(outputFile), encryptedStrings); }

}

**MoveLetterAlgorithm**

package com.company;

public class MoveLetterAlgorithm implements EncryptionAlgorithm {

@Override

public String encryptData(String plainText) { StringBuffer result = new StringBuffer();

for (int i = 0; i < plainText.length(); i++) {

if (Character.isUpperCase(plainText.charAt(i))) {

char ch = (char) (((int) plainText.charAt(i) + 4 - 65) % 26 + 65); result.append(ch);

} else {

char ch = (char) (((int) plainText.charAt(i) + 4 - 97) % 26 + 97); result.append(ch);

}

}

return result.toString();

}

}

**DeleteLetterAlgorithm**

package com.company;

public class DeleteLetterAlgorithm implements EncryptionAlgorithm {

@Override

public String encryptData(String plainText) {

return plainText.replaceAll("[AEIOUaeiou]", "");

}

}

**XorAlgorithm**

package com.company;

public class XorAlgorithm implements EncryptionAlgorithm {

@Override

public String encryptData(String plainText) {

String key = "010101";

byte[] bytes = plainText.getBytes(); byte[] keys = key.getBytes();

byte[] out = new byte[bytes.length];

for (int i = 0; i < bytes.length; i++) {

out[i] = (byte) (bytes[i] ^ keys[i % keys.length]);

}

return new String(out);

}

}

File.txt



deleteletter.txt 

moveletter.txt 

xorletter.txt

****

**Рисунок с результатом работы программы**

****

**Выводы**:  
Я приобрела навыки применения паттернов проектирования при решении практических задач с использованием языка Java.