МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

Специальность ПО5

Выполнил:
А.А. Игнатюк,
студент группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2021 г.

Брест 2021

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

Вариант 9.

Задание 1.
Требования к выполнению:
• Прочитать задания, взятые из каждой группы.
• Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.
• Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

9) 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.

Выбран паттерн строитель. Паттерн строитель - это паттерн проектирования, который позволяет поэтапно создавать сложные объекты с помощью четко определенной последовательности действий.

Спецификация ввода:
-

Спецификация вывода:
<параметры функций System.out.println() (содержимое полей объектов)>
…

Структура проекта:

Рисунок 1.1 - Структура проекта.

Код программы:

Burger.java

Java > Builder > src > build > Burger.java > ...

```java
package build;

public final class Burger extends Dish {
    private BurgerType m_BurgerType = null;

    public Burger(final Double c_Price, final String c_Name, final Double c_Weight, final BurgerType c_BurgerType) {
        super(c_Price, c_Name, c_Weight, Dish.Type.BURGER);
        this.m_BurgerType = c_BurgerType;
    }

    public final BurgerType f_get_burger_type() {
        return this.m_BurgerType;
    }

    public final void f_set_burger_type(final BurgerType c_BurgerType) {
        this.m_BurgerType = c_BurgerType;
    }

    public static enum BurgerType {
        VEGAN, CHICKEN
    }
}

```

Рисунок 1.2 - Содержимое файла Burger.java.

ColdDrink.java

Java > Builder > src > build > ColdDrink.java > ...

```java
package build;

public final class ColdDrink extends Drink {
    private ColdDrinkType m_ColdDrinkType = null;

    public ColdDrink(final Double c_Price, final String c_Name, final Double c_Volume,
            final ColdDrinkType c_ColdDrinkType) {
        super(c_Price, c_Name, c_Volume, Drink.Type.HOT);
        this.m_ColdDrinkType = c_ColdDrinkType;
    }

    public final ColdDrinkType f_get_cold_drink_type() {
        return this.m_ColdDrinkType;
    }

    public final void f_set_cold_drink_type(final ColdDrinkType c_ColdDrinkType) {
        this.m_ColdDrinkType = c_ColdDrinkType;
    }

    public static enum ColdDrinkType {
        COLA, PEPSI
    }
}

```

Рисунок 1.3 - Содержимое файла ColdDrink.java.

Dish.java ×

Java > Builder > src > build > Dish.java > ...

```java
package build;

public abstract class Dish extends Product {
    private Double m_Weight = 0.0;
    private Type m_Type = null;

    public Dish(final Double c_Price, final String c_Name, final Double c_Weight, final Type c_Type) {
        super(c_Price, c_Name);
        this.m_Weight = c_Weight;
        this.m_Type = c_Type;
    }

    public final Double f_get_weight() {
        return this.m_Weight;
    }

    public final void f_set_wight(final Double c_Weight) {
        this.m_Weight = c_Weight;
    }

    public final Type f_get_type() {
        return this.m_Type;
    }

    public final void f_set_type(final Type c_Type) {
        this.m_Type = c_Type;
    }

    public static enum Type {
        BURGER, SALAD
    }
}

```

Рисунок 1.4 - Содержимое файла Dish.java.

Main.java ×

Java > Builder > src > build > Main.java > ...

```java
package build;

public final class Main {
    Run | Debug
    public final static void main(final String[] c_Args) throws Exception {
        Order.Builder v_Builder = Order.Builder.f_new_instance();
        v_Builder.f_set_dish(new Burger(12.99, "Chicken Burger", 400.0, Burger.BurgerType.CHICKEN));
        v_Builder.f_set_drink(new ColdDrink(4.38, "Coca-Cola", 200.0, ColdDrink.ColdDrinkType.COLA));
        v_Builder.f_set_need_to_pack(Boolean.TRUE);
        v_Builder.f_set_pack_price(0.89);

        final Order c_Order = v_Builder.f_build();
        System.out.println("Total price: " + c_Order.f_get_total_price().toString());
    }
}

```

Рисунок 1.5 - Содержимое файла Main.java.

Drink.java

Java > Builder > src > build > Drink.java > ...

```java
package build;

public abstract class Drink extends Product {
    private Double m_Volume = 0.0;
    private Type m_Type = null;

    public Drink(final Double c_Price, final String c_Name, final Double c_Volume, final Type c_Type) {
        super(c_Price, c_Name);
        this.m_Volume = c_Volume;
        this.m_Type = c_Type;
    }

    public final Double f_get_volume() {
        return this.m_Volume;
    }

    public final void f_set_volume(final Double c_Volume) {
        this.m_Volume = c_Volume;
    }

    public final Type f_get_type() {
        return this.m_Type;
    }

    public final void f_set_type(final Type c_Type) {
        this.m_Type = c_Type;
    }

    public static enum Type {
        HOT, COLD
    }
}

```

Рисунок 1.6 - Содержимое файла Drink.java.

Salad.java

Java > Builder > src > build > Salad.java > ...

```java
package build;

public final class Salad extends Dish {
    private SaladType m_SaladType = null;

    public Salad(final Double c_Price, final String c_Name, final Double c_Weight, final SaladType c_SaladType) {
        super(c_Price, c_Name, c_Weight, Dish.Type.SALAD);
        this.m_SaladType = c_SaladType;
    }

    public final SaladType f_get_salad_type() {
        return this.m_SaladType;
    }

    public final void f_set_salad_type(final SaladType c_SaladType) {
        this.m_SaladType = c_SaladType;
    }

    public static enum SaladType {
        VEGETABLE, SEAFOOD
    }
}

```

Рисунок 1.7 - Содержимое файла Salad.java.

HotDrink.java

Java > Builder > src > build > HotDrink.java > ...

```java
package build;

public final class HotDrink extends Drink {
    private HotDrinkType m_HotDrinkType = null;

    public HotDrink(final Double c_Price, final String c_Name, final Double c_Volume,
            final HotDrinkType c_HotDrinkType) {
        super(c_Price, c_Name, c_Volume, Drink.Type.HOT);
        this.m_HotDrinkType = c_HotDrinkType;
    }

    public final HotDrinkType f_get_hot_drink_type() {
        return this.m_HotDrinkType;
    }

    public final void f_set_hot_drink_type(final HotDrinkType c_HotDrinkType) {
        this.m_HotDrinkType = c_HotDrinkType;
    }

    public static enum HotDrinkType {
        TEA, COFFEE
    }
}

```

Рисунок 1.8 - Содержимое файла HotDrink.java.

Product.java

Java > Builder > src > build > Product.java > ...

```java
package build;

public abstract class Product {
    private Double m_Price = 0.0;
    private String m_Name = new String();

    public Product(final Double c_Price, final String c_Name) {
        this.m_Price = c_Price;
        this.m_Name = c_Name;
    }

    public final Double f_get_price() {
        return this.m_Price;
    }

    public final void f_set_price(final Double c_Price) {
        this.m_Price = c_Price;
    }

    public final String f_get_name() {
        return this.m_Name;
    }

    public final void f_set_name(final String c_Name) {
        this.m_Name = c_Name;
    }
}

```

Рисунок 1.9 - Содержимое файла Product.java.

Order.java ×

Java > Builder > src > build > Order.java > ...

```
1   package build;
2
3   public final class Order {
4       final private Dish m_Dish;
5       final private Drink m_Drink;
6
7       final private Double m_PackPrice;
8       final private Boolean m_NeedToPack;
9
10      final private Double m_TotalPrice;
11
12      public Order(final Builder c_Builder) {
13          this.m_Dish = c_Builder.m_Dish;
14          this.m_Drink = c_Builder.m_Drink;
15          this.m_PackPrice = c_Builder.m_PackPrice;
16          this.m_NeedToPack = c_Builder.m_NeedToPack;
17          this.m_TotalPrice = c_Builder.m_TotalPrice;
18      }
19
20      public final Dish f_get_dish() {
21          return this.m_Dish;
22      }
23
24      public final Drink f_get_drink() {
25          return this.m_Drink;
26      }
27
28      public final Double f_get_pack_price() {
29          return this.m_PackPrice;
30      }
31
32      public final Boolean f_get_need_to_pack() {
33          return this.m_NeedToPack;
34      }
35
36      public final Double f_get_total_price() {
37          return this.m_TotalPrice;
38      }
39
40      public static class Builder {
41          private Dish m_Dish = null;
42          private Drink m_Drink = null;
43
44          private Double m_PackPrice = 0.0;
45          private Boolean m_NeedToPack = Boolean.FALSE;
46
47          private Double m_TotalPrice = 0.0;
48
49          public static Builder f_new_instance() {
50              return new Builder();
51          }
52
53          private Builder() {
54          }
55
```

Рисунок 1.10 - Содержимое файла Order.java.

Продолжение рисунка 1.10.

```
56          public final void f_set_dish(final Dish c_Dish) {
57              this.m_Dish = c_Dish;
58              this.m_TotalPrice = this.f_total_price();
59          }
60
61          public final void f_set_drink(final Drink c_Drink) {
62              this.m_Drink = c_Drink;
63              this.m_TotalPrice = this.f_total_price();
64          }
65
66          public final void f_set_pack_price(final Double c_PackPrice) {
67              this.m_PackPrice = c_PackPrice;
68              this.m_TotalPrice = this.f_total_price();
69          }
70
71          public final void f_set_need_to_pack(final Boolean c_NeedToPack) {
72              this.m_NeedToPack = c_NeedToPack;
73              this.m_TotalPrice = this.f_total_price();
74          }
75
76          private final Double f_total_price() {
77              Double v_TotalPrice = 0.0;
78
79              if (this.m_Dish != null) {
80                  v_TotalPrice += this.m_Dish.f_get_price();
81              }
82
83              if (this.m_Drink != null) {
84                  v_TotalPrice += this.m_Drink.f_get_price();
85              }
86
87              if (this.m_NeedToPack) {
88                  v_TotalPrice += this.m_PackPrice;
89              }
90
91              return v_TotalPrice;
92          }
93
94          public final Order f_build() {
95              return new Order(this);
96          }
97      }
98  }
99
```

```
PS C:\Users\User\Documents\Visual Studio Code>  & 'c:\Users\User\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\launcher.bat' 'C:\Program Files\Eclipse Foundation\jdk-11.0.12.7-hotspot\bin\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\User\AppData\Roaming\Code\User\workspaceStorage\35645f84fe309ce1d5ab2a9af314d21a\redhat.java\jdt_ws\Visual Studio Code_9125e193\bin' 'build.Main'
Total price: 18.26
PS C:\Users\User\Documents\Visual Studio Code>
```

Рисунок 1.11 - Результат выполнения программы.

Задание 2.

Требования к выполнению:

• Прочитать задания, взятые из каждой группы.

• Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.

• Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

9) 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 стрелкамии” хранятся повороты стрелок.

Выбран паттерн фасад. Шаблон фасад - структурный шаблон проектирования, позволяющий скрыть сложность системы путём сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

Спецификация ввода:

-

Спецификация вывода:

<параметры функций System.out.println() (содержимое полей объектов)>

…

Структура проекта:

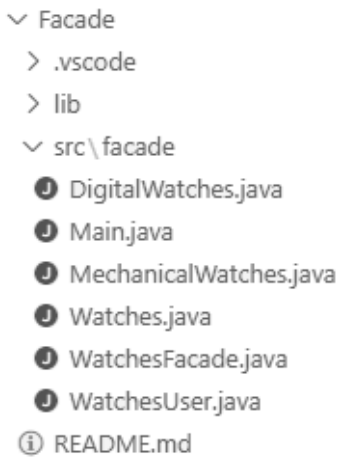

Рисунок 2.1 - Структура проекта.

Код программы:

DigitalWatches.java

Java > Facade > src > facade > DigitalWatches.java > ...

```java
package facade;

public final class DigitalWatches implements Watches {
    private Integer m_Hours = 0, m_Minutes = 0;

    public DigitalWatches(final Integer c_Hours, final Integer c_Minutes) {
        this.m_Hours = c_Hours;
        this.m_Minutes = c_Minutes;
    }

    @Override
    public final String f_get_time() {
        return new String((this.m_Hours < 10 ? '0' + this.m_Hours.toString() : this.m_Hours.toString()) + ':'
                + (this.m_Minutes < 10 ? '0' + this.m_Minutes.toString() : this.m_Minutes.toString()));
    }

    @Override
    public final void f_set_time(final Integer c_Hours, final Integer c_Minutes) {
        this.m_Hours = c_Hours;
        this.m_Minutes = c_Minutes;
    }

    @Override
    public final Integer f_get_hours() {
        return this.m_Hours;
    }

    @Override
    public final void f_set_hours(final Integer c_Hours) {
        this.m_Hours = c_Hours;
    }

    @Override
    public final Integer f_get_minutes() {
        return this.m_Minutes;
    }

    @Override
    public final void f_set_minutes(final Integer c_Minutes) {
        this.m_Minutes = c_Minutes;
    }
}

```

Рисунок 2.2 - Содержимое файла DigitalWatches.java.

Main.java

Java > Facade > src > facade > Main.java > ...

```java
package facade;

public final class Main {
    Run | Debug
    public final static void main(final String[] c_Args) throws Exception {
        final Integer c_Hours = 9, c_Minutes = 40;

        final MechanicalWatches c_MechanicalWatches = new MechanicalWatches(c_Hours, c_Minutes);
        final WatchesFacade c_FirstFacade = new WatchesFacade(c_MechanicalWatches);
        final WatchesUser c_FirstUser = new WatchesUser(c_FirstFacade);

        final DigitalWatches v_DigitalWatches = new DigitalWatches(c_Hours, c_Minutes);
        final WatchesFacade c_SecondFacade = new WatchesFacade(v_DigitalWatches);
        final WatchesUser c_SecondUser = new WatchesUser(c_SecondFacade);

        System.out.println("Time on mechanical watches: " + c_FirstUser.f_get_time());
        System.out.println("Angle of rotation of the hour hand: " + c_MechanicalWatches.f_get_hour_degrees());
        System.out.println("Angle of rotation of the minute hand: " + c_MechanicalWatches.f_get_minute_degrees());
        System.out.println();
        System.out.println("Time on digital watches: " + c_SecondUser.f_get_time());
    }
}

```

Рисунок 2.3 - Содержимое файла Main.java.

MechanicalWatches.java ×

Java > Facade > src > facade > MechanicalWatches.java > ...

```
1   package facade;
2
3   public final class MechanicalWatches implements Watches {
4       private final static Integer c_HoursToDegreesK = 30, c_MinutesToDegreesK = 3;
5       private Integer m_HourDegrees = 0, m_MinuteDegrees = 0;
6
7       public MechanicalWatches(final Integer c_Hours, final Integer c_Minutes) {
8           this.m_HourDegrees = c_Hours * c_HoursToDegreesK;
9           this.m_MinuteDegrees = c_Minutes * c_MinutesToDegreesK;
10      }
11
12      @Override
13      public final String f_get_time() {
14          final Integer c_Hours = this.m_HourDegrees / c_HoursToDegreesK,
15                        c_Minutes = this.m_MinuteDegrees / c_MinutesToDegreesK;
16
17          return new String((c_Hours < 10 ? '0' + c_Hours.toString() : c_Hours.toString()) + ':'
18                  + (c_Minutes < 10 ? '0' + c_Minutes.toString() : c_Minutes.toString()));
19      }
20
21      @Override
22      public final void f_set_time(final Integer c_Hours, final Integer c_Minutes) {
23          this.m_HourDegrees = c_Hours * c_HoursToDegreesK;
24          this.m_MinuteDegrees = c_Minutes * c_MinutesToDegreesK;
25      }
26
27      @Override
28      public final void f_set_hours(final Integer c_Hours) {
29          this.m_HourDegrees = c_Hours * c_HoursToDegreesK;
30      }
31
32      @Override
33      public final Integer f_get_hours() {
34          return this.m_HourDegrees / c_HoursToDegreesK;
35      }
36
37      @Override
38      public final void f_set_minutes(final Integer c_Minutes) {
39          this.m_MinuteDegrees = c_Minutes * c_MinutesToDegreesK;
40      }
41
42      @Override
43      public final Integer f_get_minutes() {
44          return this.m_MinuteDegrees / c_MinutesToDegreesK;
45      }
46
47      public final void f_set_hour_degrees(final Integer c_HourDegrees) {
48          this.m_HourDegrees = c_HourDegrees;
49      }
50
51      public final Integer f_get_hour_degrees() {
52          return this.m_HourDegrees;
53      }
54
55      public final void f_set_minute_degrees(final Integer c_MinuteDegrees) {
56          this.m_MinuteDegrees = c_MinuteDegrees;
57      }
58
59      public final Integer f_get_minute_degrees() {
60          return this.m_MinuteDegrees;
61      }
62  }
63
```

Рисунок 2.4 - Содержимое файла MechanicalWatches.java.

Watches.java

Java > Facade > src > facade > Watches.java > ...

```java
package facade;

public interface Watches {
    public String f_get_time();

    public void f_set_time(final Integer c_Hours, final Integer c_Minutes);

    public Integer f_get_hours();

    public void f_set_hours(final Integer c_Hours);

    public Integer f_get_minutes();

    public void f_set_minutes(final Integer c_Minutes);
}

```

Рисунок 2.5 - Содержимое файла Watches.java.

WatchesFacade.java

Java > Facade > src > facade > WatchesFacade.java > ...

```java
package facade;

public final class WatchesFacade {
    private final DigitalWatches m_DigitalWatches;
    private final MechanicalWatches m_MechanicalWatches;

    public WatchesFacade(final DigitalWatches c_DigitalWatches) {
        this.m_DigitalWatches = c_DigitalWatches;
        this.m_MechanicalWatches = null;
    }

    public WatchesFacade(final MechanicalWatches c_MechanicalWatches) {
        this.m_DigitalWatches = null;
        this.m_MechanicalWatches = c_MechanicalWatches;
    }

    public final String f_get_time() {
        if (m_DigitalWatches != null) {
            return m_DigitalWatches.f_get_time();
        }

        return m_MechanicalWatches.f_get_time();
    }
}

```

Рисунок 2.6 - Содержимое файла WatchesFacade.java.

WatchesUser.java

Java > Facade > src > facade > WatchesUser.java > ...

```java
package facade;

public final class WatchesUser {
    private final WatchesFacade m_WatchesFacade;

    public WatchesUser(final WatchesFacade c_WatchesFacade) {
        this.m_WatchesFacade = c_WatchesFacade;
    }

    public final String f_get_time() {
        return m_WatchesFacade.f_get_time();
    }
}

```

Рисунок 2.7 - Содержимое файла WatchesUser.java.

```
Time on digital watches: 09:40
PS C:\Users\User\Documents\Visual Studio Code>  c:; cd 'c:\Users\User\Documents\Visual Studio Code'; & 'c:\Use
rs\User\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\launcher.bat' 'C:\Program Files\Eclipse Fo
undation\jdk-11.0.12.7-hotspot\bin\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\User\AppData\Roaming\Code
\User\workspaceStorage\35645f84fe309ce1d5ab2a9af314d21a\redhat.java\jdt_ws\Visual Studio Code_9125e193\bin' 'f
acade.Main'
Time on mechanical watches: 09:40
Angle of rotation of the hour hand: 270
Angle of rotation of the minute hand: 120

Time on digital watches: 09:40
PS C:\Users\User\Documents\Visual Studio Code>
```

Рисунок 2.8 - Результат выполнения программы.

Задание 3.

Требования к выполнению:

• Прочитать задания, взятые из каждой группы.

• Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.

• Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

9) 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.

Выбран паттерн абстрактная фабрика. 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. Затем пишутся классы, реализующие этот интерфейс.

Спецификация ввода:

-

Спецификация вывода:

<параметры функций System.out.println() (содержимое полей объектов)>

…

Структура проекта:

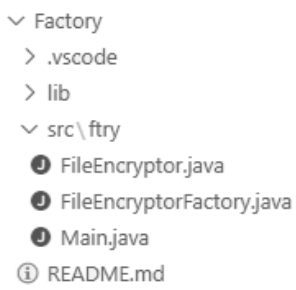

Рисунок 3.1 - Структура проекта.

Код программы:

FileEncryptor.java

Java > Factory > src > ftry > FileEncryptor.java > ...

```java
package ftry;

public class FileEncryptor {
    private final FileEncryptor.Encryptor m_Encryptor;

    protected FileEncryptor(final FileEncryptor.Encryptor c_Encryptor) {
        this.m_Encryptor = c_Encryptor;
    }

    public final String f_encrypt(final String c_Input) {
        return m_Encryptor.f_encrypt(c_Input);
    }

    protected static interface Encryptor {
        public String f_encrypt(final String c_Input);
    }

    protected final static class VowelsEncryptor implements Encryptor {
        public VowelsEncryptor() {
        }

        @Override
        public final String f_encrypt(final String c_Input) {
            return c_Input.replaceAll("[AEIOUaeiou]", "");
        }
    }

    protected final static class XorEncryptor implements Encryptor {
        private final Integer m_XorValue;

        public XorEncryptor(final Integer c_XorValue) {
            this.m_XorValue = c_XorValue;
        }

        @Override
        public final String f_encrypt(final String c_Input) {
            char[] v_Result = c_Input.toCharArray();

            for (Integer v_I = 0, v_Size = v_Result.length; v_I < v_Size; ++v_I) {
                v_Result[v_I] = (char) (((int) v_Result[v_I]) ^ this.m_XorValue);
            }

            return new String(v_Result);
        }
    }

    protected final static class ShiftEncryptor implements Encryptor {
        private final Integer m_ShiftValue;

        public ShiftEncryptor(final Integer c_ShiftValue) {
            this.m_ShiftValue = c_ShiftValue;
        }

        @Override
        public final String f_encrypt(final String c_Input) {
            char[] v_Result = c_Input.toCharArray();

            for (Integer v_I = 0, v_Size = v_Result.length; v_I < v_Size; ++v_I) {
                v_Result[v_I] = (char) (((int) v_Result[v_I]) + this.m_ShiftValue);
            }

            return new String(v_Result);
        }
    }
}

```

Рисунок 3.2 - Содержимое файла FileEncryptor.java.

FileEncryptorFactory.java

Java > Factory > src > ftry > FileEncryptorFactory.java > ...

```java
package ftry;

public final class FileEncryptorFactory extends FileEncryptor {
    private FileEncryptorFactory(final Encryptor c_Encryptor) {
        super(c_Encryptor);
    }

    public final static FileEncryptor f_get_vowels_file_encryptor() {
        return new FileEncryptor(new FileEncryptor.VowelsEncryptor());
    }

    public final static FileEncryptor f_get_xor_file_encryptor(final Integer c_XorValue) {
        return new FileEncryptor(new FileEncryptor.XorEncryptor(c_XorValue));
    }

    public final static FileEncryptor f_get_shift_file_encryptor(final Integer c_ShiftValue) {
        return new FileEncryptor(new FileEncryptor.ShiftEncryptor(c_ShiftValue));
    }
}

```

Рисунок 3.3 - Содержимое файла FileEncryptorFactory.java.

Main.java

Java > Factory > src > ftry > Main.java > ...

```java
package ftry;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;

public final class Main {
    Run | Debug
    public final static void main(final String[] c_Args) throws IOException {
        final Integer c_ShiftValue = 5, c_XorValue = 2;

        final FileEncryptor c_VowelsFileEncryptor = FileEncryptorFactory.f_get_vowels_file_encryptor(),
                c_XorFileEncryptor = FileEncryptorFactory.f_get_xor_file_encryptor(c_XorValue),
                c_ShiftFileEncryptor = FileEncryptorFactory.f_get_shift_file_encryptor(c_ShiftValue);

        final String c_Input = new String(Files.readString(Path.of("test.txt")));

        System.out.println("Source: " + c_Input + '\n');
        System.out.println("After vowels encryption: " + c_VowelsFileEncryptor.f_encrypt(c_Input));
        System.out.println("After xor encryption: " + c_XorFileEncryptor.f_encrypt(c_Input));
        System.out.println("After shift encryption: " + c_ShiftFileEncryptor.f_encrypt(c_Input));
    }
}

```

Рисунок 3.4 - Содержимое файла Main.java.

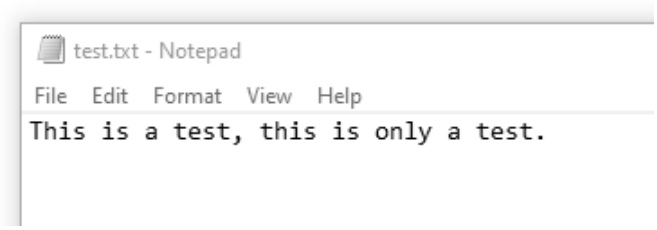

Рисунок 3.5 - Содержимое файла test.txt.

```
PS C:\Users\User\Documents\Visual Studio Code>  c:; cd 'c:\Users\User\Documents\Visual Studio Code'; & 'c:\Users\User\.v
scode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\launcher.bat' 'C:\Program Files\Eclipse Foundation\jdk-11.0.12
.7-hotspot\bin\java.exe' '-Dfile.encoding=UTF-8' '-cp' 'C:\Users\User\AppData\Roaming\Code\User\workspaceStorage\35645f8
4fe309ce1d5ab2a9af314d21a\redhat.java\jdt_ws\Visual Studio Code_9125e193\bin' 'ftry.Main'
Source: This is a test, this is only a test.

After vowels encryption: Ths s  tst, ths s nly  tst.
After xor encryption: Vjkq"kq"c"vgqv."vjkq"kq"mln{"c"vgqv,
After shift encryption: Ymnx%nx%f%yjxy1%ymnx%nx%tsq~%f%yjxy3
PS C:\Users\User\Documents\Visual Studio Code>
```

Рисунок 3.6 - Результат выполнения программы.

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