|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №21, №22**

Выполнил студент группы ИНБО-02-20 Самойленко М. А.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2020г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2020г.

Отметка о выполнении

**Москва – 2020 г.**

**Практическое занятие №21-22**

**Задание:**

В трёх приложениях будет очень много общего.

Главное окно, панель инструментов, команды меню будут весьма схожими. Чтобы не писать повторяющуюся основу трижды, вам поручили разработать основу (каркас) приложения, которую можно использовать во всех трёх редакторах. На совещании в компании была принята следующая архитектура:

Есть некий базовый интерфейс IDocument, представляющий документ неопределённого рода.

От него впоследствии будут унаследованы конкретные документы: TextDocument, ImageDocument, MusicDocument и т.п.

Интерфейс

IDocument перечисляет общие свойства и операции для всех документов.

• При нажатии пунктов меню File -&gt; New и File -&gt; Open требуется создать новый экземпляр документа (конкретного подкласса).

Однако каркас не должен быть привязан ни к какому конкретному виду документов.

• Нужно создать фабричный интерфейс ICreateDocument. Этот интерфейс содержит два абстрактных фабричных метода: CreateNew

и CreateOpen, оба возвращают экземпляр IDocument

• Каркас оперирует одним экземпляром IDocument и одним экземпляром ICreateDocument. Какие конкретные классы будут

подставлены сюда, определяется во время запуска приложения.

Требуется:

1. Создать перечисленные классы. Создать каркас приложения — окно редактора с меню File. В меню File реализовать пункты New,Open, Save, Exit.
2. Продемонстрировать работу каркаса на примере текстового редактора. Потребуется создать конкретный унаследованный класс TextDocument и фабрику для него — CreateTextDocument. В данной практической работе рекомендуется использовать следующие паттерны: Абстрактная фабрика и фабричный метод.

##Рекомендации

Создавать интерфейс в 3 задаче нет необходимости, но это и не запрещается.

**Код программы:**

**Интерфейс IDocument**

package ru.mirea.java.practice21.practice22.document;  
  
public interface IDocument {  
 void open();  
 void save();  
 void exit();  
 void newDocument();  
}

**Класc ImageDocument**

package ru.mirea.java.practice21.practice22.document;  
  
public class ImageDocument implements IDocument {  
 @Override  
 public void open() {  
 System.*out*.println("Открыт файл типа Image");  
 }  
  
 @Override  
 public void save() {  
 System.*out*.println("Сохранен файл типа Image");  
 }  
  
 @Override  
 public void exit() {  
 System.*out*.println("Закрыт файл типа Image");  
 }  
  
 @Override  
 public void newDocument() {  
 System.*out*.println("Создан файл типа Image");  
 }  
}

**Класc** **MusicDocument**

package ru.mirea.java.practice21.practice22.document;  
  
public class MusicDocument implements IDocument {  
 @Override  
 public void open() {  
 System.*out*.println("Открыт файл типа Music");  
 }  
  
 @Override  
 public void save() {  
 System.*out*.println("Сохранен файл типа Music");  
 }  
  
 @Override  
 public void exit() {  
 System.*out*.println("Закрыт файл типа Music");  
 }  
  
 @Override  
 public void newDocument() {  
 System.*out*.println("Создан файл типа Music");  
 }  
}

**Класc** **TextDocument**

package ru.mirea.java.practice21.practice22.document;  
  
public class TextDocument implements IDocument {  
 @Override  
 public void open() {  
 System.*out*.println("Открыт файл типа Text");  
 }  
  
 @Override  
 public void save() {  
 System.*out*.println("Сохранен файл типа Text");  
 }  
  
 @Override  
 public void exit() {  
 System.*out*.println("Закрыт файл типа Text");  
 }  
  
 @Override  
 public void newDocument() {  
 System.*out*.println("Создан файл типа Text");  
 }  
}

**Класc** **CreateImageDocument**

package ru.mirea.java.practice21.practice22.factory;  
  
import ru.mirea.java.practice21.practice22.document.IDocument;  
import ru.mirea.java.practice21.practice22.document.ImageDocument;  
  
public class CreateImageDocument implements ICreateDocument{  
 private ImageDocument image;  
 @Override  
 public IDocument CreateNew() {  
 image = new ImageDocument();  
 return image;  
 }  
  
 @Override  
 public IDocument CreateOpen() {  
 return image;  
 }  
}

**Класc** **CreateMusicDocument**

package ru.mirea.java.practice21.practice22.factory;  
  
import ru.mirea.java.practice21.practice22.document.IDocument;  
import ru.mirea.java.practice21.practice22.document.MusicDocument;  
  
public class CreateMusicDocument implements ICreateDocument{  
 private MusicDocument music;  
 @Override  
 public IDocument CreateNew() {  
 music = new MusicDocument();  
 return music;  
 }  
  
 @Override  
 public IDocument CreateOpen() {  
 return music;  
 }  
}

**Класc CreateTextDocument**

package ru.mirea.java.practice21.practice22.factory;  
  
import ru.mirea.java.practice21.practice22.document.IDocument;  
import ru.mirea.java.practice21.practice22.document.TextDocument;  
  
public class CreateTextDocument implements ICreateDocument{  
 private TextDocument text;  
 @Override  
 public IDocument CreateNew() {  
 text = new TextDocument();  
 return text;  
 }  
  
 @Override  
 public IDocument CreateOpen() {  
 return text;  
 }  
}

**Интерфейс** **ICreateDocument**

package ru.mirea.java.practice21.practice22.factory;  
  
import ru.mirea.java.practice21.practice22.document.IDocument;  
  
public interface ICreateDocument {  
 IDocument CreateNew();  
 IDocument CreateOpen();  
}

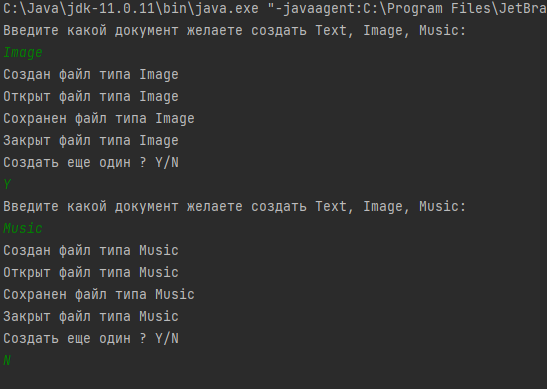
**Класc Frame**

package ru.mirea.java.practice21.practice22;  
  
import ru.mirea.java.practice21.practice22.document.IDocument;  
import ru.mirea.java.practice21.practice22.factory.CreateImageDocument;  
import ru.mirea.java.practice21.practice22.factory.CreateMusicDocument;  
import ru.mirea.java.practice21.practice22.factory.CreateTextDocument;  
import ru.mirea.java.practice21.practice22.factory.ICreateDocument;  
  
import java.util.Scanner;  
  
public class Frame {  
 private IDocument document;  
 private ICreateDocument createDocument;  
 public void newDocument(){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите какой документ желаете создать Text, Image, Music: ");  
 String nameDocument = in.nextLine();  
 if (nameDocument.equals("Text")){  
 createDocument = new CreateTextDocument();  
 document = createDocument.CreateNew();  
 document.newDocument();  
 }  
 if (nameDocument.equals("Image")){  
 createDocument = new CreateImageDocument();  
 document = createDocument.CreateNew();  
 document.newDocument();  
 }  
 if (nameDocument.equals("Music")){  
 createDocument = new CreateMusicDocument();  
 document = createDocument.CreateNew();  
 document.newDocument();  
 }  
 }  
 public void save(){  
 document.save();  
 }  
 public void open(){  
 document.open();  
 }  
 public void exit(){  
 document.exit();  
 }  
}

**Класc Main**

package ru.mirea.java.practice21.practice22;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Frame karkas = new Frame();  
 Scanner in = new Scanner(System.*in*);  
 String YN;  
 do {  
 karkas.newDocument();  
 karkas.open();  
 karkas.save();  
 karkas.exit();  
 System.*out*.println("Создать еще один ? Y/N");  
 YN = in.nextLine();  
 } while(YN.equals("Y"));  
 }  
}

**Тесты**



**Выводы**

Я изучил и научился пользоваться паттернами: фабричный метод, абстрактная фабрика.