|  |
| --- |
|  |
| 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 ЗАДАНИЮ №10**

Выполнил студент группы ИНБО-02-20 Веселкин.И.А

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва – 2021 г.**

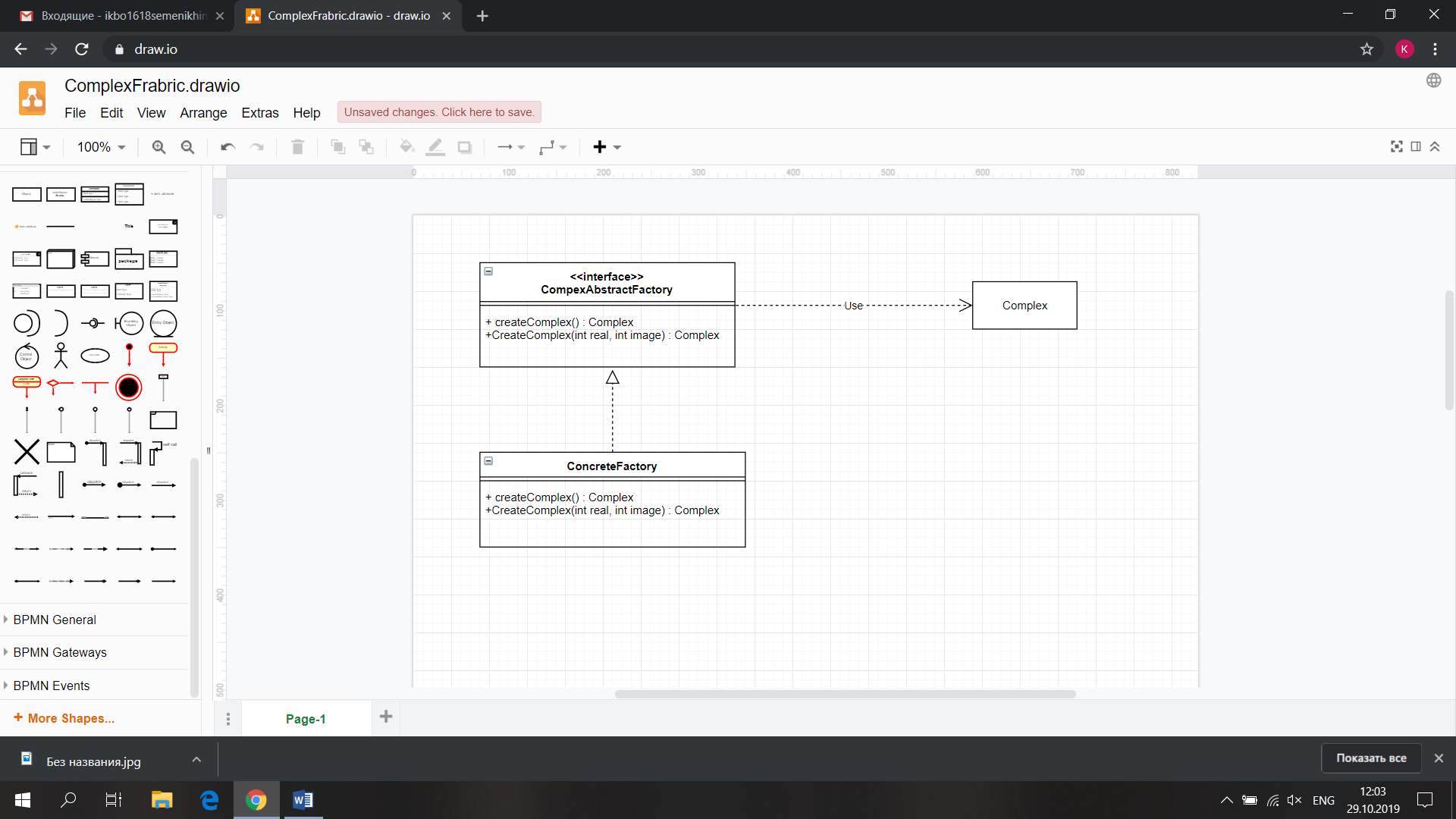
## **Цель практической работы**

Цель данной практической работы – научиться применять порождающие паттерны при разработке программ на Java. В данной практической работе рекомендуется использовать следующие паттерны: Абстрактная фабрика и фабричный метод.

## **Задания**

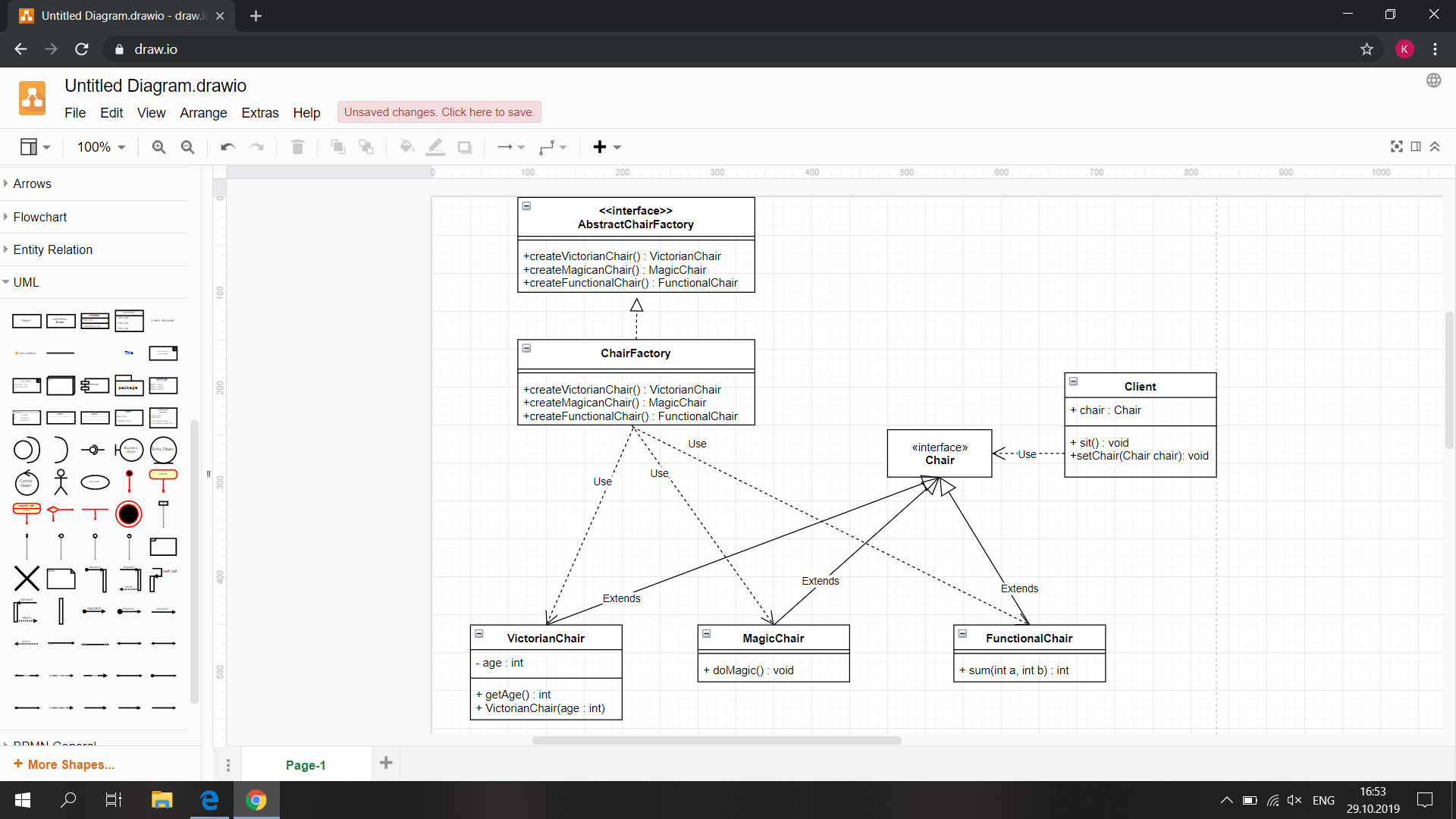
**Упражнение** 1

Реализовать класс Абстрактная фабрика для комплексных чисел



**Упражнение** 2

Реализовать класс Абстрактная фабрика для различных типов стульев: Викторианский стул, Многофункциональный стул, Магический стул, а также интерфейс Стул, от которого наследуются все классы стульев, и класс Клиент, который использует интерфейс стул в своем методе Sit (Chair chair).



## **Ход Работы**

**Реализация упражнения 1**

Реализация класса Complex:

public class Complex {  
 private final int real;  
 private final int image;  
  
 public Complex(int real, int image) {  
 this.real = real;  
 this.image = image;  
 }  
  
 public Complex() {  
 this.real = 1;  
 this.image = 1;  
 }  
  
 @Override  
 public String toString() {  
 if (image == 0) return real + "";  
 if (real == 0) return image + "i";  
 if (image < 0) return real + " - " + (-image) + "i";  
 return real + " + " + image + "i";  
 }  
}

Реализация интерфейса ComplexAbstractFactory:

public interface ComplexAbstractFactory {  
 public Complex CreateComplex(int real, int image);  
}

Реализация класса Concrete:

public class Concrete implements ComplexAbstractFactory {  
  
  
 public Complex CreateComplex(int real, int image) {  
 return new Complex(real, image);  
 }  
}

Реализация класса Test:

public class Test {  
 public static void main(String[] args) {  
 Concrete concreteFactory = new Concrete();  
 Complex complex = concreteFactory.CreateComplex(10,-2);  
 System.*out*.println(complex);  
 }  
}

**Реализация упражнения 2**

Реализация интерфейса AbstractChairFactory:

public interface AbstractChairFactory {  
 public VictorianChair createVictorianChair();  
 public MagicChair createMagicChair();  
 public FunctionalChair createFunctionalChair();  
}

Реализация класса ChairFactory:

public class ChairFactory implements AbstractChairFactory {  
 @Override  
 public VictorianChair createVictorianChair() {  
 return new VictorianChair();  
 }  
  
 @Override  
 public MagicChair createMagicChair() {  
 return new MagicChair();  
 }  
  
 @Override  
 public FunctionalChair createFunctionalChair() {  
 return new FunctionalChair();  
 }  
}

Реализация класса Client:

public class Client {  
 public Chair chair;  
  
 public void sit() {  
 System.*out*.println("Спасибо за заказ!");  
 }  
  
 public void setChair(Chair chair) {  
 this.chair = chair;  
 }  
  
 @Override  
 public String toString() {  
 return "Клиент выбрал " +  
 "стул =" + chair;  
 }  
}

Реализация класса FunctionalChair:

public class FunctionalChair implements Chair{  
 public int sum(int a, int b) {  
 return a+b;  
 }  
  
}

Реализация класса MagicChair:

public class MagicChair implements Chair {  
 public void doMagic() {  
 System.*out*.println("Real Magic!");  
 }  
}

Реализация класса Main:

public class Main {  
 public static void main(String[] args) {  
 Client client= new Client();  
  
 ChairFactory chairFactory= new ChairFactory();  
 FunctionalChair functionalChair=chairFactory.createFunctionalChair();  
 MagicChair magicChair=chairFactory.createMagicChair();  
 VictorianChair victorianChair = chairFactory.createVictorianChair();  
  
 System.*out*.println(functionalChair.sum(12,13));  
 magicChair.doMagic();  
 client.sit();  
 System.*out*.println(victorianChair);  
 }  
}

Реализация класса VictorianChair:

public class VictorianChair implements Chair {  
 private int age;  
  
 public VictorianChair(int age) {  
 this.age = age;  
 }  
  
 public VictorianChair() {  
 this.age = 15;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 @Override  
 public String toString() {  
 return "VictorianChair" + '\''+  
 "age=" + age;  
 }  
}

## **Вывод**

В ходе выполнения работы я научился работать с научиться применять порождающие паттерны при разработке программ на Java.

С полным кодом программы можно ознакомится по ссылке: https://github.com/gost214/Practika-INBO-02-20-Veselkin/tree/main/pract10