|  |
| --- |
|  |
| 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 №10**

Выполнил студент группы ИНБО-02-20 Маркарянц Р.Д.

Принял Степанов П.В.

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

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

Отметка о выполнении

**Москва – 2021 г.**

## **Задание**

**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**

В ходе выполнения работы были получены следующие исходные коды:

package ru.mirea.inbo0220.markaryants.pr10.Exersize1;  
  
public interface ComplexAbstractFactory {  
 Complex createComplex();  
 Complex createComplex(int real, int image);  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize1;  
import java.util.Scanner;  
  
public class ConcreteFactory implements ComplexAbstractFactory{  
  
 @Override  
 public Complex createComplex() {  
 Scanner in = new Scanner(System.*in*);  
 int real = in.nextInt();  
 int image = in.nextInt();  
 Complex com = new Complex(real,image);  
 return com;  
 }  
  
 @Override  
 public Complex createComplex(int real, int image) {  
 Complex com = new Complex(real,image);  
 return com;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize1;  
  
public class Complex {  
 private int image;  
 private int real;  
  
 public Complex(int real,int image) {  
 this.image = image;  
 this.real = real;  
 }  
  
 public Complex() {  
 }  
  
 public int getImage() {  
 return image;  
 }  
  
 public int getReal() {  
 return real;  
 }  
  
 public void setImaginary(int imaginary) {  
 this.image = imaginary;  
 }  
  
 public void setReal(int real) {  
 this.real = real;  
 }  
  
 @Override  
 public String toString() {  
 return "Complex{" + " real=" + real +  
 ", image= " + image +  
 '}';  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize1;  
  
public class Test {  
 public static void main(String[] args) {  
 ConcreteFactory n = new ConcreteFactory();  
 Complex com = n.createComplex();  
 Complex com2 = n.createComplex(5,6);  
 System.*out*.println(com.toString());  
 System.*out*.println(com2.toString());  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public interface AbstractChairFactory {  
 public VictorianChair createVictorianChair();  
 public MagicChair createMagicanChair();  
 public FunctionalChair createFunctionalChair();  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public class ChairFactory implements AbstractChairFactory {  
 @Override  
 public VictorianChair createVictorianChair() {  
 return new VictorianChair();  
 }  
  
 @Override  
 public MagicChair createMagicanChair() {  
 return new MagicChair();  
 }  
  
 @Override  
 public FunctionalChair createFunctionalChair() {  
 return new FunctionalChair();  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public class VictorianChair implements Chair {  
 private int age;  
  
 public int getAge() {  
 return age;  
 }  
  
 public VictorianChair(int age) {  
 this.age = age;  
 }  
  
 public VictorianChair() {  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public class MagicChair implements Chair {  
 public void doMagic(){  
 System.*out*.println("Magic");  
 }  
  
 public MagicChair() {  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public class FunctionalChair implements Chair {  
 public int sum(int a, int b){  
 return a+b;  
 }  
  
 public FunctionalChair() {  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public interface Chair {  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public class Client {  
 public Chair chair;  
 public void sit() {  
 System.*out*.println("Sitting");  
 }  
  
 public Client() {  
 }  
  
 public void setChair(Chair chair) {  
 this.chair = chair;  
 }  
}

package ru.mirea.inbo0220.markaryants.pr10.Exersize2;  
  
public class Test {  
 public static void main(String[] args) {  
 Client c = new Client();  
 ChairFactory factory = new ChairFactory();  
 c.setChair(factory.createMagicanChair());  
 ((MagicChair) c.chair).doMagic();  
 c.setChair(factory.createFunctionalChair());  
 int sum = ((FunctionalChair) c.chair).sum(15, 3);  
 System.*out*.println("Sum: " + sum);  
 c.setChair(factory.createVictorianChair());  
 ((VictorianChair) c.chair).setAge(67);  
 int age = ((VictorianChair) c.chair).getAge();  
 System.*out*.println("Age: " + age);  
 }  
}

## **Вывод**

Выполнив практическую работу, я научился применять порождающие паттерны при разработке программ на Java.