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

Отчет по выполнению практической работы 11

**Тема. ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ. ПОРОЖДАЮЩИЕ ПАТТЕРНЫ: АБСТРАКТНАЯ ФАБРИКА,**

**ФАБРИЧНЫЙ МЕТОД**

Дисциплина Программирование на языке Джава

Выполнил

|  |  |
| --- | --- |
| студент | Болотов Михаил |
|  | Фамилия И.О. |
| группа | ИКБО-06-19 |
|  | Номер группы |

Москва 2020

**Содержание**

[​ Теория 2](#__RefHeading___Toc4128_811265792)

[​ Задание 2](#__RefHeading___Toc4136_811265792)

[​ Код 3](#__RefHeading___Toc35859_811265792)

[​ VictorianChair.java 3](#__RefHeading___Toc42698_811265792)

[​ MagicChair.java 4](#__RefHeading___Toc42700_811265792)

[​ FunctionalChair.java 4](#__RefHeading___Toc42702_811265792)

[​ ChairFactory.java 4](#__RefHeading___Toc42704_811265792)

[​ Chair.java 5](#__RefHeading___Toc42706_811265792)

[​ Client.java 5](#__RefHeading___Toc42708_811265792)

[​ AbstractChairFactory.java 6](#__RefHeading___Toc42710_811265792)

[​ Скриншот 6](#__RefHeading___Toc3748_811265792)

[​ Заключение 6](#__RefHeading___Toc3750_811265792)

[​ Библиографический список 6](#__RefHeading___Toc3752_811265792)

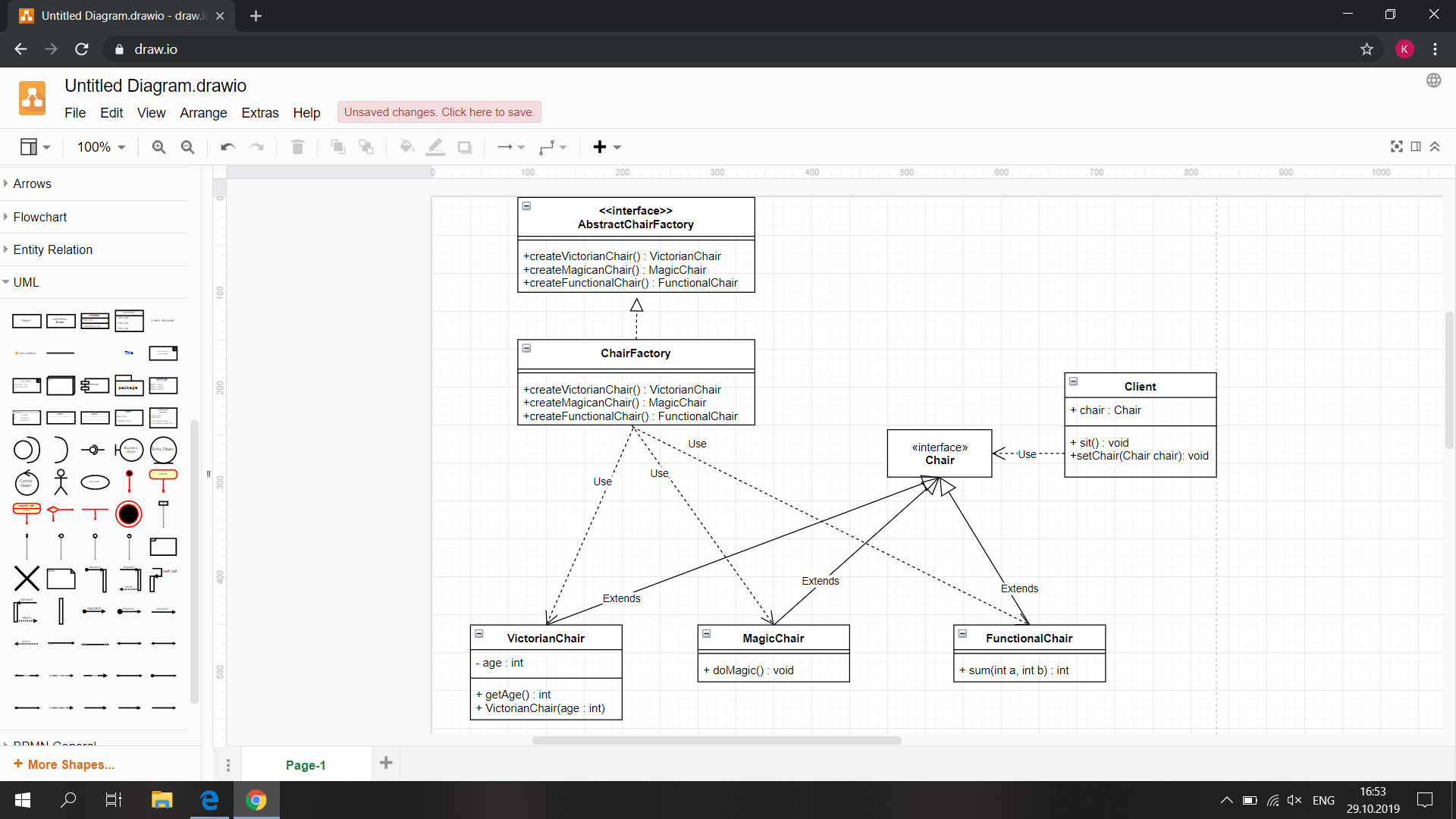
# Теория

Понятие паттерна.

Паттерны (или шаблоны) проектирования описывают типичные способы решения часто встречающихся проблем при проектировании программ.

# Задание

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).



# 

# Код

### VictorianChair.java

package dev.ky3he4ik.pr.pr11;  
  
public class VictorianChair implements Chair {  
 int age;  
  
 public VictorianChair(int age) {  
 this.age = age;  
 }  
  
 @Override  
 public void sit() {  
 System.out.println("You just sat on fancy chair");  
 }  
  
 public int getAge() {  
 return age;  
 }  
}

### MagicChair.java

package dev.ky3he4ik.pr.pr11;  
  
public class MagicChair implements Chair {  
 @Override  
 public void sit() {  
 System.out.println("Oops! You're now a cat!");  
 }  
  
 public void doMagic() {  
 System.out.println("Magic happened somewhere. But where?");  
 }  
}

### FunctionalChair.java

package dev.ky3he4ik.pr.pr11;  
  
public class FunctionalChair implements Chair {  
 public int sum(int a, int b) {  
 return a + b;  
 }  
  
 @Override  
 public void sit() {  
 System.out.println("You opened with your char a beer and sitting here");  
 }  
}

### ChairFactory.java

package dev.ky3he4ik.pr.pr11;  
  
public class ChairFactory implements AbstractChairFactory{  
  
 @Override  
 public VictorianChair createVictorianChair() {  
 return new VictorianChair(42);  
 }  
  
 @Override  
 public FunctionalChair createFunctionalChair() {  
 return new FunctionalChair();  
 }  
  
 @Override  
 public MagicChair createMagicChair() {  
 return new MagicChair();  
 }  
}

### Chair.java

package dev.ky3he4ik.pr.pr11;  
  
public interface Chair {  
 void sit();  
}

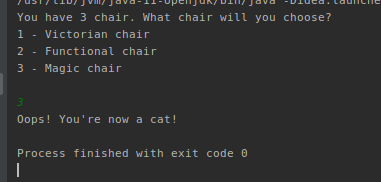
### Client.java

package dev.ky3he4ik.pr.pr11;  
  
import java.util.Scanner;  
  
public class Client {  
 Chair chair;  
  
 public void setChair(Chair chair) {  
 this.chair = chair;  
 }  
  
 public void sit() {  
 chair.sit();  
 }  
  
 public static void main(String[] args) {  
 System.out.println("You have 3 chair. What chair will you choose?\n1 - Victorian chair\n2 - Functional chair\n3 - Magic chair");  
 Scanner scanner = new Scanner(System.in);  
 Client client = new Client();  
 AbstractChairFactory factory = new ChairFactory();  
 switch (scanner.nextInt()) {  
 case 1:  
 client.setChair(factory.createVictorianChair());  
 break;  
 case 2:  
 client.setChair(factory.createFunctionalChair());  
 break;  
 case 3:  
 client.setChair(factory.createMagicChair());  
 break;  
 }  
 client.sit();  
 }  
}

### AbstractChairFactory.java

package dev.ky3he4ik.pr.pr11;  
  
public interface AbstractChairFactory {  
 VictorianChair createVictorianChair();  
 FunctionalChair createFunctionalChair();  
 MagicChair createMagicChair();  
}

# Скриншот



# Заключение

В данной практической работе я научился применять порождающие паттерны при разработке программ на Java

# Библиографический список

1. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.