

本 科 生 实 验 报 告

**实验课程 软件体系结构与设计**

**学院名称 计算机与网络安全学院（牛津布鲁克斯学院）**

**专业名称 软件工程**

**学生姓名 徐睿航**

**学生学号 202013160210**

**指导教师 单武扬**

**实验地点 C075-05-32**

**实验成绩**

**二〇二二年四月­——二〇二二年六月**

实验八 创建型模式的应用

一、实验目的

1. 理解“创建型模式”的基本概念、基本原理和应用场景

2. 掌握简单工厂模式(Simple Factory)、工厂方法模式(Factory Method)、抽象工厂模式(Abstract Factory)的使用方法

二、实验器材

1.计算机一台。

2.Rational Rose 工具软件。

3.Eclipse集成开发环境

三、实验内容

1. 简单工厂模式

利用简单工厂模式，完成以下程序设计（UML+JAVA）

游戏人物创建，设计一个人物创建仓：

1. 当用户在仓门口喊一声：“汉子！”该仓就能创建一个男性角色
2. 当用户喊”妹子!”,该仓就创建一个女性角色
3. 当用户喊"随便!”，该仓就随机创建一个角色性别
4. 工厂模式

利用工厂模式，完成以下程序设计(UML+JAVA)

游戏职业构造器：

1. 设计一个游戏职业的构造器，可以为用户构造不同的职业
2. 职业有，建筑师、魔法师、医师、金工技师、厨师
3. 请根据用户传递的角色职业信息和姓名，为用户生成不同的职业
4. 抽象工厂模式

利用抽象工厂模式，完成以下程序设计(UML+JAVA)

游戏人物外观构造器

1. 为游戏人物创建不同的外观
2. 分为:服饰、头发、眼睛、肤色、身鎬
3. 要求不同的职业有不同的外观范围，同一职业外观相同
4. 提高版本:你也可以设计成不同职业不同外观，同一职业根据等级不同，外观也有细微差异(如头发长短、瞳孔颜色深浅、身高高矮、服饰配饰多少的差异等等)

每个设计作业都要求包含2部分：UML类图和代码实现（java）

四、实验步骤

1. 简单工厂模式

分析：

在游戏人物构建中，根据传入的参数不同，返回不同性别角色类的实例，专门定义一个RoleFactory类（工厂角色）来负责创建其他FemaleRole类和MaleRole类（具体产品角色）的实例，被创建的实例都具有共同的父类Role类（抽象产品角色）。

1. 画UML类图

添加Role类、FemaleRole类、MaleRole类、RoleFactory类；

建立Role类与FemaleRole类、MaleRole类之间为继承关系，RoleFactory类与FemaleRole类、MaleRole类之间为依赖关系。

文本, 信件

描述已自动生成 图示

描述已自动生成

1. 编写Java代码

编写Role类，为抽象类，其中有抽象方法play()；

编写MaleRole类继承Role类，并实现play()，FemaleRole类与其类似；

编写RoleFactory类，其中produceRole(String role)负责创建其他类的实例；

编写Client类，根据传入的参数不同，创建不同性别的角色。

1. 工厂模式

分析：

在游戏职业构造器中，根据用户传递的角色职业信息和姓名，为用户生成不同的职业，定义一个ProfessionFactory类（工厂父类）负责定义创建Profession（产品角色）的公共接口，而ArchitectFactory、ChefFactory、MagicianFactory、PhysicianFactory、MetaltechnicianFactory类（工厂子类）负责生成Architect、Chef、Magician、Physician、Metaltechnician（具体的产品角色）。

1. 画UML类图

添加工厂父类、抽象产品角色类、工厂子类、具体的产品角色类；

图片包含 文本

描述已自动生成

建立工厂父类与工厂子类之间的继承关系、工厂子类与具体产品角色类之间的依赖关系、具体产品角色类与抽象产品角色类之间的继承关系。

图示

描述已自动生成

1. 编写Java代码

编写Profeesion类，为抽象类，其中有抽象方法build()；

编写Architect类继承Profession类，实现build()方法，Chef、Magician、Physician、Metaltechnician类与其类似；

编写ProfessionFactory类，抽象类，其中有抽象方法produceProfession(String name）；

编写ArchitectFactory类继承ProfessionFactory类，实现produceProfession(String name)方法，ChefFactory、MagicianFactory、PhysicianFactory、MetaltechnicianFactory类与其类似；

编写Client类，根据传递的角色职业信息和姓名，生成不同的职业。

1. 抽象工厂模式

分析：

在游戏人物外观构造器中，不同的职业有不同的外观，Clothes抽象类（抽象产品），其子类有architectClothes、chefClothes类（同一产品等级结构的具体产品）等，则抽象类与其具体类之间构成一个产品等级结构，Complexion、Height、Hair、Eyes类和其子类与之类似，而architectClothes、architectComlexion等类（同一产品族的具体产品）是由同一个工厂architectFactory类（具体工厂）生产的，位于不同产品等级结构中的一组产品中，其他与之类似，这些具体工厂继承professionFactory类（抽象工厂）。

1. 画UML类图

添加抽象产品类、具体产品类、抽象工厂类、具体工厂类、Client类，并建立类与类之间的关系。

图示

描述已自动生成

1. 编写Java代码

编写Clothes类，抽象类，其中有抽象方法displayClothes()，Complexion、Height、Hair、Eyes类与其类似；

编写architectClothes类，继承Clothes类，实现displayClothes()方法，其它具体产品类与其类似；

编写professionFactory类，抽象类，有抽象方法produceClothes()、produceEyes()等；

编写architectFactory类继承professionFactory类，实现produceClothes()、produceEyes()方法，chefFactory、magicianFactory、metaltechnicianFactory、physicianFactory类与其类似；

编写Client类，根据不同的职业，生成不同的外观。

五、实验结果或核心代码

1. 简单工厂模式

图形用户界面

低可信度描述已自动生成

Role.java

**package** simplefactory;

**public** **abstract** **class** Role {

**public** **abstract** **void** play();

}

FemaleRole.java

**package** simplefactory;

**public** **class** FemaleRole **extends** Role {

**public** **void** play() {

System.***out***.println("创建了一个女性角色！");

}

}

MaleRole.java

**package** simplefactory;

**public** **class** MaleRole **extends** Role {

**public** **void** play() {

System.***out***.println("创建了一个男性角色！");

}

}

RoleFactory.java

**package** simplefactory;

**public** **class** RoleFactory {

**public** **static** Role produceRole(String role) {

**if**(role.equals("汉子")) {

**return** **new** MaleRole();

}

**else** **if**(role.equals("妹子")) {

**return** **new** FemaleRole();

}

**else** **if**(role.equals("随便")) {

**int** n = (**int**)((Math.*random*()\*10)%2);

**if**(n == 0) {

**return** **new** MaleRole();

}

**else** {

**return** **new** FemaleRole();

}

}

**return** **null**;

}

}

Client.java

**package** simplefactory;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

Role role;

role = RoleFactory.*produceRole*("汉子");

role.play();

}

}

运行结果：

图示, 示意图

中度可信度描述已自动生成

1. 工厂模式

图示

描述已自动生成

Profession.java

**package** factory;

**public** **abstract** **class** Profession {

**protected** String name;

**public** Profession(String name) {

**this**.name = name;

}

**public** **abstract** **void** build();

}

Architect.java

**package** factory;

**public** **class** Architect **extends** Profession {

**public** Architect(String name) {

**super**(name);

}

**public** **void** build() {

System.***out***.println("生成了一个建筑师！名为："+name);

}

}

Chef.java

**package** factory;

**public** **class** Chef **extends** Profession {

**public** Chef(String name) {

**super**(name);

}

**public** **void** build() {

System.***out***.println("生成了一个厨师！名为："+name);

}

}

Magician.java

**package** factory;

**public** **class** Magician **extends** Profession {

**public** Magician(String name) {

**super**(name);

}

**public** **void** build() {

System.***out***.println("生成了一个魔法师！名为："+name);

}

}

Metaltechnician.java

**package** factory;

**public** **class** Metaltechnician **extends** Profession {

**public** Metaltechnician(String name) {

**super**(name);

}

**public** **void** build() {

System.***out***.println("生成了一个金工技师！名为："+name);

}

}

Physician.java

**package** factory;

**public** **class** Physician **extends** Profession {

**public** Physician(String name) {

**super**(name);

}

**public** **void** build() {

System.***out***.println("生成了一个医师！名为："+name);

}

}

ProfessionFactory.java

**package** factory;

**public** **abstract** **class** ProfessionFactory {

**public** **abstract** Profession produceProfession(String name);

}

ArchitectFactory.java

**package** factory;

**public** **class** ArchitectFactory **extends** ProfessionFactory {

**public** Profession produceProfession(String name) {

**return** **new** Architect(name);

}

}

ChefFactory.java

**package** factory;

**public** **class** ChefFactory **extends** ProfessionFactory {

**public** Profession produceProfession(String name) {

**return** **new** Chef(name);

}

}

MagicianFactory.java

**package** factory;

**public** **class** MagicianFactory **extends** ProfessionFactory {

**public** Profession produceProfession(String name) {

**return** **new** Magician(name);

}

}

MetaltechnicianFactory.java

**package** factory;

**public** **class** MetaltechnicianFactory **extends** ProfessionFactory {

**public** Profession produceProfession(String name) {

**return** **new** Metaltechnician(name);

}

}

PhysicianFactory.java

**package** factory;

**public** **class** PhysicianFactory **extends** ProfessionFactory {

**public** Profession produceProfession(String name) {

**return** **new** Physician(name);

}

}

Client.java

**package** factory;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

ProfessionFactory factory = **null**;

Profession pro;

String profession = "厨师";

String name = "张三";

**if**(profession.equals("建筑师")) {

factory = **new** ArchitectFactory();

}

**else** **if**(profession.equals("厨师")) {

factory = **new** ChefFactory();

}

**else** **if**(profession.equals("魔术师")) {

factory = **new** MagicianFactory();

}

**else** **if**(profession.equals("金工技师")) {

factory = **new** MetaltechnicianFactory();

}

**else** **if**(profession.equals("医师")) {

factory = **new** MagicianFactory();

}

pro = factory.produceProfession(name);

pro.build();

}

}

运行结果：

图片包含 文本

描述已自动生成

1. 抽象工厂模式

图示

描述已自动生成

Clothes.java

**package** abstractfactory;

**public** **abstract** **class** Clothes {

**public** **abstract** **void** displayClothes();

}

Complexion.java

**package** abstractfactory;

**public** **abstract** **class** Complexion {

**public** **abstract** **void** displayComplexion();

}

Eyes.java

**package** abstractfactory;

**public** **abstract** **class** Eyes {

**public** **abstract** **void** displayEyes();

}

Hair.java

**package** abstractfactory;

**public** **abstract** **class** Hair {

**public** **abstract** **void** displayHair();

}

Height.java

**package** abstractfactory;

**public** **abstract** **class** Height {

**public** **abstract** **void** displayHeight();

}

architectClothes.java

**package** abstractfactory;

**public** **class** architectClothes **extends** Clothes {

**public** **void** displayClothes() {

System.***out***.println("休闲服");

}

}

architectComplexion.java

**package** abstractfactory;

**public** **class** architectComplexion **extends** Complexion {

**public** **void** displayComplexion() {

System.***out***.println("黑皮肤");

}

}

architectEyes.java

**package** abstractfactory;

**public** **class** architectEyes **extends** Eyes {

**public** **void** displayEyes() {

System.***out***.println("黑眼睛");

}

}

architectHair.java

**package** abstractfactory;

**public** **class** architectHair **extends** Hair {

**public** **void** displayHair() {

System.***out***.println("黑头发");

}

}

architectHeight.java

**package** abstractfactory;

**public** **class** architectHeight **extends** Height {

**public** **void** displayHeight() {

System.***out***.println("170cm");

}

}

其他具体产品类的Java文件类似；

professionFactory.java

**package** abstractfactory;

**public** **abstract** **class** professionFactory {

**public** **abstract** Clothes produceClothes();

**public** **abstract** Eyes produceEyes();

**public** **abstract** Hair produceHair();

**public** **abstract** Height produceHeight();

**public** **abstract** Complexion produceComplexion();

}

architectFactory.java

**package** abstractfactory;

**public** **class** architectFactory **extends** professionFactory {

**public** Clothes produceClothes() {

**return** **new** architectClothes();

}

**public** Eyes produceEyes() {

**return** **new** architectEyes();

}

**public** Hair produceHair() {

**return** **new** architectHair();

}

**public** Height produceHeight() {

**return** **new** architectHeight();

}

**public** Complexion produceComplexion() {

**return** **new** architectComplexion();

}

}

chefFactory.java

**package** abstractfactory;

**public** **class** chefFactory **extends** professionFactory {

**public** Clothes produceClothes() {

**return** **new** chefClothes();

}

**public** Eyes produceEyes() {

**return** **new** chefEyes();

}

**public** Hair produceHair() {

**return** **new** chefHair();

}

**public** Height produceHeight() {

**return** **new** chefHeight();

}

**public** Complexion produceComplexion() {

**return** **new** chefComplexion();

}

}

magicianFactory.java

**package** abstractfactory;

**public** **class** magicianFactory **extends** professionFactory {

**public** Clothes produceClothes() {

**return** **new** magicianClothes();

}

**public** Eyes produceEyes() {

**return** **new** magicianEyes();

}

**public** Hair produceHair() {

**return** **new** magicianHair();

}

**public** Height produceHeight() {

**return** **new** magicianHeight();

}

**public** Complexion produceComplexion() {

**return** **new** magicianComplexion();

}

}

metaltechnicianFactory.java

**package** abstractfactory;

**public** **class** metaltechnicianFactory **extends** professionFactory {

**public** Clothes produceClothes() {

**return** **new** metaltechnicianClothes();

}

**public** Eyes produceEyes() {

**return** **new** metaltechnicianEyes();

}

**public** Hair produceHair() {

**return** **new** metaltechnicianHair();

}

**public** Height produceHeight() {

**return** **new** metaltechnicianHeight();

}

**public** Complexion produceComplexion() {

**return** **new** metaltechnicianComplexion();

}

}

physicianFactory.java

**package** abstractfactory;

**public** **class** physicianFactory **extends** professionFactory {

**public** Clothes produceClothes() {

**return** **new** physicianClothes();

}

**public** Eyes produceEyes() {

**return** **new** physicianEyes();

}

**public** Hair produceHair() {

**return** **new** physicianHair();

}

**public** Height produceHeight() {

**return** **new** physicianHeight();

}

**public** Complexion produceComplexion() {

**return** **new** physicianComplexion();

}

}

Client.java

**package** abstractfactory;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

professionFactory factory = **new** chefFactory();

Clothes clothes = factory.produceClothes();

clothes.displayClothes();

}

}

运行结果：

文本

中度可信度描述已自动生成

|  |  |
| --- | --- |
| **学生实验心德** | 通过这次实验我明白了创建型模式的应用。  学生（签名）：徐睿航  2022年5月30日 |
| **指导教师评语** | 成绩评定：  指导教师（签名）：  年 月 日 |