学号 E21514033 专业 E21514033 姓名 何铭春

实验日期 **4.12**  教师签字 成绩

实验报告

【实验名称】 抽象工厂模式与单件模式编程实现

【实验目的】

1.模拟公司数据库创建过程，结合抽象工厂模式和单件模式实现。

2.实现三张表的抽象工厂。

【实验原理】

1. 抽象工厂用于创建一系列产品集团。
2. 单件模式用于工厂类，使得工厂类对象仅存在一个。

【实验内容】

数据表相关定义

//Table父类,用于指向子类对象

public abstract class Table{

abstract void query(String in);

abstract void add(String in);

}

//用户表Mysql

class UserTableForMysql extends Table{

public void query(String in){

System.out.println("user和mysql的查询函数");

}

public void add(String in){

System.out.println("user和mysql的添加函数");

}

}

//用户表sql

class UserTableForSql extends Table{

public void query(String in){

System.out.println("user和sql的查询函数");

}

public void add(String in){

System.out.println("user和sql的添加函数");

}

}

//部门表mysql

class DepartmentTableForMysql extends Table{

public void query(String in){

System.out.println("department和mysql的查询函数");

}

public void add(String in){

System.out.println("department和mysql的添加函数");

}

}

//部门表sql

class DepartmentTableForSql extends Table{

public void query(String in){

System.out.println("department和sql的查询函数");

}

public void add(String in){

System.out.println("departmentt和sql的添加函数");

}

}

//项目表mysql

class ProjectTableForMysql extends Table{

public void query(String in){

System.out.println("project和mysql的查询函数");

}

public void add(String in){

System.out.println("project和mysql的添加函数");

}

}

//项目表sql

class ProjectTableForSql extends Table{

public void query(String in){

System.out.println("project和sql的查询函数");

}

public void add(String in){

System.out.println("projectt和sql的添加函数");

}

}

工厂类相关定义

//工厂类定义，实现了单件模式

class Factory{

public static Factory spring = null;

protected Factory(){

}

public static Factory getFactory(){

if (spring == null) {

spring = new Factory();

}

return spring;

}

public Table createMysqlTable(){

return null;

}

public Table createSqlTable(){

return null;

}

}

//用户表工厂

class UserFactory extends Factory{

public Table createSqlTable(){

return new UserTableForSql();

}

public Table createMysqlTable(){

return new UserTableForMysql();

}

}

//部门表工厂

class DepartmentFactory extends Factory{

public Table createSqlTable(){

return new DepartmentTableForSql();

}

public Table createMysqlTable(){

return new DepartmentTableForMysql();

}

}

//项目表工厂

class ProjectFactory extends Factory{

public Table createSqlTable(){

return new ProjectTableForSql();

}

public Table createMysqlTable(){

return new ProjectTableForMysql();

}

}

测试类：

class Test {

public static void main(String[] args) {

//获取工厂对象

Factory hmc = Factory.getFactory();

//创建用户表的mysql对象

hmc = new UserFactory();

Table test = hmc.createMysqlTable();

test.query("hello");

//创建项目表的sql对象

hmc = new ProjectFactory();

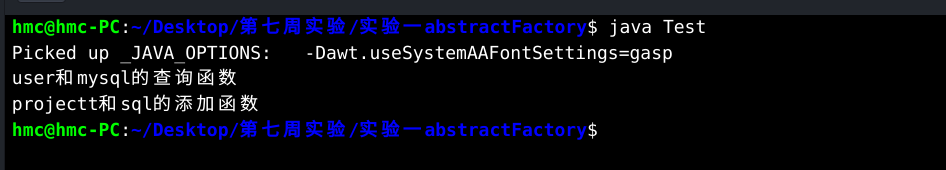
test = hmc.createSqlTable();

test.add("hi");

}

}

实验结果：



【小结或讨论】

1. 利用单件模式控制工厂对象的生成。
2. 利用抽象类工厂来简化复杂的创建过程。
3. 抽象类方法方便扩展。