# 实验四：工厂模式

### (1) 简单工厂方法（50分）

#### 1.1 UML图



#### 1.2 代码

（1） Car接口

**package** cn.edu.scau.cmi.zhouxudong.domain;

**public** **interface** Car {

**public** **void** printCarName();

}

（2）JeepCar类

**package** cn.edu.scau.cmi.zhouxudong.domain;

**public** **class** JeepCar **implements** Car {

@Override

**public** **void** printCarName() {

// **TODO** Auto-generated method stub

System.***out***.println("这是吉普车");

}

}

（3）SportCar类

**package** cn.edu.scau.cmi.zhouxudong.domain;

**public** **class** SportCar **implements** Car {

@Override

**public** **void** printCarName() {

// **TODO** Auto-generated method stub

System.***out***.println("这是运动车");

}

}

（4）CarFactory类

package cn.edu.scau.cmi.zhouxudong.simpleFactoryMethod;

import cn.edu.scau.cmi.zhouxudong.domain.Car;

import cn.edu.scau.cmi.zhouxudong.domain.JeepCar;

import cn.edu.scau.cmi.zhouxudong.domain.SportCar;

public class CarFactory {

public static final String JeepCar="JeepCar";

public static final String SportCar="SportCar";

public static Car getInstance(String carTypeString){

switch (carTypeString) {

case JeepCar: return new JeepCar();

case SportCar: return new SportCar();

}

return null;

}

}

（5）SimpleCarFactoryMethodClient类

**package** cn.edu.scau.cmi.zhouxudong.client;

**import** cn.edu.scau.cmi.zhouxudong.domain.Car;

**import** cn.edu.scau.cmi.zhouxudong.simpleFactoryMethod.CarFactory;

**public** **class** SimpleCarFactoryMethodClient {

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

Car car = CarFactory.*getInstance*("JeepCar"); //单一

car.printCarName();

}

}

### (3) 工厂方法模式（30分）

利用工厂方法实现上面的要求。

#### 2.1 UML图



#### 2.2 代码

（1）CarFactory接口

**package** cn.edu.scau.cmi.zhouxudong.factoryMethod;

**import** cn.edu.scau.cmi.zhouxudong.domain.Car;

**public** **interface** CarFactory {

**public** **static** Car getInstance() {

**return** **null**;

}

}

（2）JeepCarFactory类

**package** cn.edu.scau.cmi.zhouxudong.factoryMethod;

**import** cn.edu.scau.cmi.zhouxudong.domain.Car;

**import** cn.edu.scau.cmi.zhouxudong.domain.JeepCar;

**public** **class** JeepCarFactory **implements** CarFactory{

**public** **static** Car getInstance(){

**return** **new** JeepCar();

}

}

（3）JeepCarFactory类

**package** cn.edu.scau.cmi.zhouxudong.factoryMethod;

**import** cn.edu.scau.cmi.zhouxudong.domain.Car;

**import** cn.edu.scau.cmi.zhouxudong.domain.JeepCar;

**public** **class** JeepCarFactory **implements** CarFactory{

**public** **static** Car getInstance(){

**return** **new** JeepCar();

}

}

（4）FactoryMethodClient类

**package** cn.edu.scau.cmi.zhouxudong.client;

**import** cn.edu.scau.cmi.zhouxudong.domain.Car;

**import** cn.edu.scau.cmi.zhouxudong.factoryMethod.JeepCarFactory;

**public** **class** FactoryMethodClient {

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

// **TODO** Auto-generated method stub

Car car1 = JeepCarFactory.*getInstance*();

car1.printCarName();

}

}

### (3) 抽象工厂模式（20分）

现有两大汽车公司：奔驰(Benz)和宝马(BMW)，每家汽车公司都可以生产吉普车JeepCar和运动车SportCar请利用抽象工厂模式方法设计并实现创建不同保险公司的不同保险理赔对象。界面根据实际情况，在工程方法模式的界面基础上进行适当修改。

#### 3.1 UML图



#### 3.2 代码

(1)domainInterface包

**package** cn.edu.scau.cmi.zhouxudong.abstractFactory.domainInterface;

**public** **interface** Benz {

**final** String ***CarCompany*** = "奔驰";

}

**package** cn.edu.scau.cmi.zhouxudong.abstractFactory.domainInterface;

**public** **interface** BMW {

**final** String ***CarCompany*** = "宝马";

}

（2）domain包

①package cn.edu.scau.cmi.zhouxudong.abstractFactory.domain;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.JeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainInterface.Benz;

public class BenzJeepCar extends JeepCar implements Benz {

@Override

public void show() {

System.out.println(Benz.CarCompany+super.name);

}

}

②

package cn.edu.scau.cmi.zhouxudong.abstractFactory.domain;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.SportCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainInterface.Benz;

public class BenzSportCar extends SportCar implements Benz {

@Override

public void show() {

System.out.println(Benz.CarCompany+super.name);

}

}

③

package cn.edu.scau.cmi.zhouxudong.abstractFactory.domain;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.JeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainInterface.BMW;

public class BMWJeepCar extends JeepCar implements BMW {

@Override

public void show() {

System.out.println(BMW.CarCompany+super.name);

}

}

④

package cn.edu.scau.cmi.zhouxudong.abstractFactory.domain;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.SportCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainInterface.BMW;

public class BMWSportCar extends SportCar implements BMW {

@Override

public void show() {

//System.out.println("这是宝马吉普车");

System.out.println(BMW.CarCompany+super.name);

}

}

1. domainAbstractClass包

**public** **abstract** **class** Car {

**public** **abstract** **void** show();

}

**public** **abstract** **class** JeepCar **extends** Car{

**public** **final** String name="吉普车";

}

**package** cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass;

**public** **abstract** **class** SportCar **extends** Car{

**public** **final** String name="运动车";

}

1. 抽象工厂

①

package cn.edu.scau.cmi.zhouxudong.abstractFactory.factory;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.JeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.SportCar;

public abstract class AbstractFactory {

public static AbstractFactory getFactory(String carCompany){

switch (carCompany) {

case "Benz": return new BenzFactory();

case "BMW": return new BMWFactory();

}

return null;

}

public abstract JeepCar createJeepCar();

public abstract SportCar createSportCar();

}

②

package cn.edu.scau.cmi.zhouxudong.abstractFactory.factory;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domain.BenzJeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domain.BenzSportCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.JeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.SportCar;

public class BenzFactory extends AbstractFactory{

@Override

public JeepCar createJeepCar(){

return new BenzJeepCar();

}

public SportCar createSportCar(){

return new BenzSportCar();

}

}

③

package cn.edu.scau.cmi.zhouxudong.abstractFactory.factory;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domain.BMWJeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domain.BMWSportCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.JeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.SportCar;

public class BMWFactory extends AbstractFactory{

@Override

public JeepCar createJeepCar(){

return new BMWJeepCar();

}

@Override

public SportCar createSportCar(){

return new BMWSportCar();

}

}

1. client类

package cn.edu.scau.cmi.zhouxudong.client;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.domainAbstractClass.JeepCar;

import cn.edu.scau.cmi.zhouxudong.abstractFactory.factory.AbstractFactory;

public class AbstractFactoryClient {

public static void main(String[] args) {

AbstractFactory carFactory = AbstractFactory.getFactory("Benz");

JeepCar jeepCar = carFactory.createJeepCar();

jeepCar.show();

}

}