设计模式实验报告

1. **单例模式**
2. 相关代码

public class Consumer

{

private string ConsumerName;

private string StudentID;

private int Age;

private Gender gender;

private string Password;

private static Consumer Singleton = null;

private static object singletonLock = new object();//该私有对象用于lock，使之可适用于多线程

private Consumer(string ConsumerName, string StudentID, int Age, Gender gender, string Password)

{

this.ConsumerName = ConsumerName;

this.StudentID = StudentID;

this.Age = Age;

this.gender = gender;

this.Password = Password;

}

public static Consumer getConsumerInstance()

{

if (Singleton == null)

{

lock (singletonLock)

{

if (Singleton == null)

{

Singleton = new Consumer("测试", "2018300000000", 20, Gender.MALE, "Test123.");

}

}

}

return Singleton;

}

public bool LogIn(string id, string password)

{

if (id == StudentID && password == Password)

return true;

else

return false;

}

public string getConsumerName()

{

return ConsumerName;

}

public void setConsumerName(string name)

{

ConsumerName = name;

}

public int getAge()

{

return Age;

}

public void setAge(int age)

{

Age = age;

}

public string getPassword()

{

return Password;

}

public void setPassword(string password)

{

Password = password;

}

public string getStudentID()

{

return StudentID;

}

public Gender getGender()

{

return gender;

}

public bool SubmitOrders(Order order)

{

//添加数据操作，即将order送给后台管理系统

return true;

}

}

1. 设计思路分析

本次实验实现的功能是珞珈自强超市商品信息服务平台中购买操作模块的实现，使用单例模式是因为在这一单一功能的实现中，这个类可以只需要一个实例，只需要自行创建后再自行向整个系统提供这一实例。

1. 优势

（1）实例控制：单例模式能够阻止其他对象实例化其自己的单例对象的副本，从而确保所有对象都访问唯一的实例。

（2）灵活性：因为类控制了实例化过程，所以类可以灵活更改实例化过程。

1. **组合模式**
2. 相关代码

（1）叶子

public class Product

{

private string ProductName;

private string Manufacturer;

private string ProductionDate;

private int Price;

public Product(string ProductName, string Manufacturer, string ProductionDate, int Price)

{

this.ProductName = ProductName;

this.Manufacturer = Manufacturer;

this.ProductionDate = ProductionDate;

this.Price = Price;

}

public string getProductName()

{

return ProductName;

}

public string getManufacturer()

{

return Manufacturer;

}

public string getProductDate()

{

return ProductionDate;

}

public int getPrice()

{

return Price;

}

}

1. 容器

public class Order

{

private string OrderID;

private Consumer consumer;

private List<Product> Commodity;

private string ConsumptionTime;

public Order(string OrderID, Consumer consumer, List<Product> Commodity, string ConsumptionTime)

{

this.OrderID = OrderID;

this.consumer = consumer;

this.Commodity = Commodity;

this.ConsumptionTime = ConsumptionTime;

}

public void AddingGoods(Product product)

{

Commodity.Add(product);

}

public bool DeleteGoods(Product product)

{

if (!Commodity.Contains(product))

return false;

else

{

Commodity.Remove(product);

return true;

}

}

}

1. 设计思路分析

本次实验实现的功能是珞珈自强超市商品信息服务平台中购买操作模块的实现，商品和商品列表通过组合模式将容器对象和叶子对象进行递归组合，能够去使得用户在使用时无须对它们进行区分。

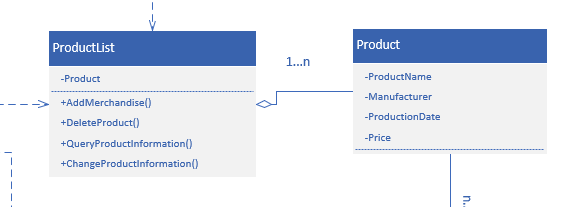


图 1 目前实现功能在类图中对应部分

1. 优势

（1）可以清楚地定义分层次的复杂对象，表示对象的全部或部分层次，让客户端忽略了层次的差异，方便对整个层次结构进行控制。

（2）客户端可以一致地使用一个组合结构或其中单个对象，不必关心处理的是单个对象还是整个组合结构，简化了客户端代码。

（3）增加新的容器构件和叶子构件都很方便，符合开闭原则。

（4）为树形结构的面向对象实现提供了一种灵活的解决方案。