抽象工厂模式实验

**【实验目的及要求】**

1. 熟悉抽象工厂模式的定义；
2. 掌握抽象工厂模式角色及作用，熟悉抽象工厂模式结构图；
3. 掌握抽象工厂模式编程。

**【实验内容】**

**在当前疫情期间，所有课程都要在做好线上教学和现场教学两种形式，显然两种教学存在一些差别，例如线上侧重视频，而现场侧重PPT课件。现考虑有两门课程：Java和设计模式。使用抽象工厂模式实例化课程教学。**

**【实验步骤】**

1. **类图**
2. **实现代码**
3. **抽象产品----**

**线上教学（OnlineCourse）**

**package** designpattern;

**public** **interface** OnlineCourse {

**public** **void** teaching();

}

**现场教学（OfflineCourse）**

**package** designpattern;

**public interface** OfflineCourse {

p**ublic void** teaching();

}

1. **具体产品类**

**Java线上教学（JavaOnlineCourse）**

**public class JavaOnlineCourse implements OnlineCourse{  
 @Override  
 public void teaching() {  
 System.*out*.println("java线上教学");  
 }  
}**

**Java现场教学（JavaOfflineCourse）**

**public class JavaOfflineCourse implements OfflineCourse{  
  
 @Override  
 public void teaching() {  
 System.*out*.println("java线下教学");  
 }  
}**

**设计模式线上教学（PatternOnlineCourse）**

**public class PatternOnlineCourse implements OnlineCourse{  
 @Override  
 public void teaching() {  
 System.*out*.println("设计模式线上课");  
 }  
}**

**设计模式现场教学（PatternOfflineCourse）**

**public class PatternOfflineCourse implements OfflineCourse {  
  
 @Override  
 public void teaching() {  
 System.*out*.println("设计模式线下课");  
 }  
}**

1. **抽象工厂类（**CourseFactory**）**

**package** designpattern;

**public** **interface** CourseFactory {

**public** OnlineCourse produceOnlineCourse();

**public** OfflineCourse produceOfflineCourse();

}

1. **具体工厂类**

**Java工厂类（JavaCourseFactory）**

**public class JavaCourseFactory implements CourseFactory{  
 @Override  
 public OnlineCourse produceOnlineCourse() {  
 return new JavaOnlineCourse();  
 }  
  
 @Override  
 public OfflineCourse produceOfflineCourse() {  
 return new JavaOfflineCourse();  
 }  
}**

**设计模式工厂类（PatternCourseFactory）**

**public class PatternCourseFactory implements CourseFactory{  
 @Override  
 public OnlineCourse produceOnlineCourse() {  
 return new PatternOnlineCourse();  
 }  
  
 @Override  
 public OfflineCourse produceOfflineCourse() {  
 return new PatternOfflineCourse();  
 }  
}**

1. **工具类**

**XML工具类（XMLUtil）**

**public class XMLUtil {  
 public static CourseFactory getBean() {  
 DocumentBuilderFactory factory=DocumentBuilderFactory.*newInstance*();  
 try {  
 DocumentBuilder builder = factory.newDocumentBuilder();  
 Document document = builder.parse(new File("src/com/fancy/experiment4/config.xml"));  
 NodeList courseFactoryList = document.getElementsByTagName("course");  
// 获取一个第一个course对象JavaCourseFactory  
 String onlineCourseItem = courseFactoryList.item(0).getTextContent();  
 if (onlineCourseItem.equals("JavaCourseFactory")) {  
 return new JavaCourseFactory();  
 } else if (onlineCourseItem.equals("PatternCourseFactory")) {  
 return new PatternCourseFactory();  
 } else {  
 return null;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return null;  
 }  
}**

1. **配置文件**

**XML配置文件（config.xml）**

***<?*xml version="1.0" encoding="utf-8" *?>*<courseFactory>  
 <course>JavaCourseFactory</course>  
 <course>PatternCourseFactory</course>  
</courseFactory>**

1. **测试类**

**package** designpattern;

**public** **class** Test {

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

CourseFactory factory = (CourseFactory)XMLUtil.*getBean*();

OnlineCourse online = factory.produceOnlineCourse();

OfflineCourse offline = factory.produceOfflineCourse();

online.teaching();

offline.teaching();

}

}

**【思考问题】**

（1）如果增加一门课程，例如数据结构，是否可行？如何可行请完成相应代码。

（2）如果增加一种教学方式，例如混合教学，是否可行？

（3）具体描述抽象工厂模式的优缺点。

**（1）可行**

**public class DataCourseFactory implements CourseFactory{  
  
 @Override  
 public OnlineCourse produceOnlineCourse() {  
 return new DataOnlineCourse() ;  
 }  
  
 @Override  
 public OfflineCourse produceOfflineCourse() {  
 return new DataOfflineCourse();  
 }  
}**

**public class DataOfflineCourse implements OfflineCourse{  
 @Override  
 public void teaching() {  
 System.*out*.println("数据结构线下课");  
 }  
}**

**public class DataOnlineCourse implements OnlineCourse{  
 @Override  
 public void teaching() {  
 System.*out*.println("数据结构线上课");  
 }  
}**

**<course>DataCourseFactory</course>**

**else if (onlineCourseItem.equals("DataCourseFactory")) {  
 return new DataCourseFactory();  
}**

**（2）可行**

**（3）**

**抽象工厂模式通过提供接口创建相关联的对象家族，优点是分离具体类、保证产品族一致性、扩展性强，符合开闭原则和单一职责原则。但它增加了系统复杂性，难以支持新产品类型，且对设计理解要求较高，适合大型或多产品族系统。**

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | 通过本次实验，我加深了对抽象工厂模式的理解，掌握了其定义及适用场景。实验过程中，我明确了抽象工厂模式的角色（抽象工厂、具体工厂、抽象产品、具体产品）及它们的作用，并通过结构图理清了它们的关系。在编程实践中，我成功运用抽象工厂模式实现了不同产品族的创建，体验到该模式在保证产品族一致性和灵活扩展方面的优势，但也意识到其增加的系统复杂性。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |