·实验教学·

面向设计模式融合的教学案例设计与实施

欧阳宏基，吴粉侠，葛　萌，张　伟

（咸阳师范学院 计算机学院，咸阳 712000）

摘要：为了使学生充分理解设计模式的重要性，做到在需求分析中识别设计模式并灵活应用，通过 2 次显性迭代和 3 次隐性迭代设计了一个融合建造者、抽象工厂、策略和外观设计模式的教学案例。首先阐述了案例的设计思路，然后详细描述了案例的教学迭代过程，最后根据案例分解了实验教学任务，并对案例教学效果进行了评价。实验结果表明，该案例能够激发学生学习设计模式的兴趣，提高了学生利用设计模式进行面向对象问题求解的能力。

关　键　词：设计模式；教学案例；建造者设计模式；抽象工厂设计模式；外观设计模式

中图分类号：G642.0　　**文献标志码：**A　　DOI: 10.12179/1672-4550.20230356

Design and Implementation of Teaching Cases for Design Pattern Integration

OUYANG Hongji, WU Fenxia, GE Meng, ZHANG Wei

（School of Computer, Xianyang Normal University, Xianyang 712000, China）

Abstract: In order to enable students to fully understand the importance of design patterns, identify design patterns in demand analysis, and apply them flexibly, a teaching case integrating builder, abstract factory, strategy and facade design patterns is designed through two explicit iterations and three invisible iterations. Firstly, the design concept of the case is elaborated. Then the iterative process of the case teaching is described in detail. Finally, practical teaching tasks are decomposed based on the case, and the effectiveness of case teaching is evaluated. The practical results indicate that this case can stimulate students' interest in learning design patterns and improve their ability to use design patterns for object-oriented problem solving.

Key words: design pattern; teaching cases; builder pattern; abstract factory pattern; facade pattern

我们目前处在一个“软件定义一切”的时代，云计算、大数据、物联网、区块链、人工智能等技术正在引领新一轮信息技术革命和产业革命，新型信息技术助力传统产业升级和改造需要依靠软件这个载体来落地实施[1]。软件产业逐渐成为国家基础性、战略性产业，在促进国民经济的发展和社会文明进步中具有重要的地位和作用[2]，培养良好的软件人才迫在眉睫。目前，面向对象是主流的软件设计与开发技术，如何提高学生面向对象的分析、设计和编码能力是软件工程专业人才培养的一个核心问题[3-4]。

设计模式是面向对象技术在软件开发中所积累的经验总结，是七大面向对象设计原则的具体实现[5]。将设计模式应用到软件的分析、设计和开发中，从微观层面利于书写结构清晰和易于阅读的代码，提高业务功能的扩展性；从宏观层面可创建良好的软件架构，便于系统的扩展和维护。因此，灵活应用设计模式有利于提高面向对象技术的使用水平。

根据软件项目经验得知，一个具体的软件业务功能往往需要多个设计模式相互协作来完成。如何在学生掌握单一设计模式应用的基础上，通

收稿日期：2023-07-19；**修回日期：**2024-04-01

基金项目：陕西省教育科学“十四五”规划项目 2021 年度课题(SGH21Y0201)；2023 年陕西省高等教育教学改革研究项目 (23BY143)；咸阳师范学院教学改革研究项目(2021Z008)；2022 年第二批教育部产学合作协同育人项目(220803631270021)。

作者简介：欧阳宏基(1982-)，男，硕士，副教授，主要从事软件工程、软件体系结构方面的研究。E-mail: oyhj_nicholas@163.com

过分析评价当前版本设计存在的问题，通过引入新的设计模式或者面向对象技术来解决当前问题，经过多次的迭代，达到多个设计模式相互融合应用的目的，使得设计结果具有良好的扩展性、维护性，对于任课教师来说是一项挑战。

本文从多个设计模式融合的角度出发，经过 5 次迭代，将建造者、抽象工厂、策略和外观等模式进行了融合应用，从而用来表示一个复杂对象的创建过程。在每一次的迭代过程中，通过提出问题—分析问题—引出解决方案—编码实现等步骤，逐步引导学生将设计模式思想应用到实际问题的分析、设计和编码过程中，从而不断培养和提高学生面向对象技术应用的能力和水平。

1　教学现状分析

设计模式是软件设计与体系结构这门课的核心内容之一，在大三第一学期开设。前驱课程是 Java 程序设计，后续课程是 Java EE 应用开发。主要学习软件设计模式基础知识，面向对象设计原则， 常用的创建型、结构型和行为型设计模式[6]。针对设计模式部分，基本教学过程是：首先讲解某一个设计模式应用动机和概念，分析模式的组成元素；然后结合 UML 类图分析各元素之间的关系和承担的职责，展示核心元素代码编写规范；最后结合一个需求场景给出模式完整的编码过程，再对模式的优缺点进行分析和总结[7]。

设计模式具有概念学习起来简单、理解起来抽象、应用起来困难等特点。深入分析原因，主要存在以下两个问题：

1）需求分析的文字描述中不能直接提供所要使用的设计模式，需要开发人员根据应用动机来分析判断使用什么设计模式；

2）需求分析中还隐含了无法从字面含义体现的扩展点，如当前所使用设计模式固有的缺点、业务功能可能的变化等，需要从复用性、封装性以及健壮性等方面来评估当前的设计。如果无法满足，往往还需要进一步利用其他设计模式或面向对象技术进行再次设计。

以往的教学设计和实验主要是关注第一个问题，基本围绕单一设计模式进行教学活动的开展，学生对某一个模式的理解和掌握相对较好。如何解决第二个问题，是本教学案例设计与实施的目的。

2　案例设计

2.1　设计思路

本案例以创建电脑对象为目的，综合应用了建造者、抽象工厂、策略和外观设计模式，结合配置文件和反射机制等面向对象技术，经过 5 次迭代综合设计而成，每一次迭代过程如图 1 所示。每次迭代所需要的知识点、迭代依据和迭代性质如表 1 所示。

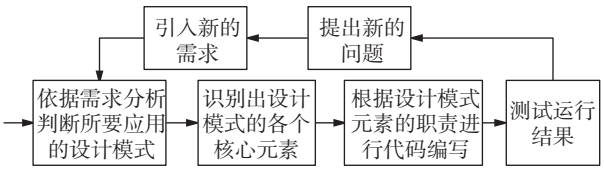

图 1　案例迭代过程图

表 1　迭代数据表

迭代次数/次	需要的知识点	迭代依据	迭代性质
1	建造者设计模式	需求分析直接描述	显性
2	抽象工厂设计模式	模式的缺点	显性
3	配置文件+反射机制	复用性	隐性
4	策略设计模式	健壮性	隐性
5	外观设计模式	封装性	隐性

其中第一次迭代所要使用的设计模式可从需求分析的显示文字描述中分析确定；第二次迭代所引入的设计模式是根据需求的显性意义推断得知，并解决了第一次所使用的设计模式的部分缺点；第三次迭代解决了代码冗余度问题，使定义的类更具复用性；第四次迭代从隐性需求角度提高了代码的健壮性；第五次迭代从隐性需求角度提高了代码的封装性。案例最终设计完成后，部分类图关系如图 2 所示。

从图 2 可看出，ComputerDirector、AbstractComputerDirector、ComputerBuilder、Computer 和 ComputerConfig 是建造者设计模式的组成元素；ComputerFactory、LenovoComputerFactory 和 AsusComputerFactory 是抽象工厂模式的组成元素；StrategyContext、ComputerBuilderStrategy、BuildStrategyWithConfig 和 BuildStrategyWithFactory 是策略模式的组成元素；ComputerFacade 是外观模式的外观类。客户端只需依赖 ComputerFacade 类就可以得到电脑对象，而无须了解电脑对象来自

抽象工厂设计模式还是建造者设计模式，以及如何创建等细节问题。

在每次迭代过程的教学中应用项目式学习（project-based learing，PBL）教学法[8]，首先分析上一次设计完后存在的问题，提出问题解决的方案，如果解决方案是引入新的设计模式，那么对该模式进行复习，针对新模式的组成元素引导学生通过新增接口、类等方式进行扩展；如果解决方案需要用到其他面向对象技术，引导学生将这部分技术编码到相关设计模式的角色中。同时将每次迭代后的编码过程录制成视频，上传到学习通平台，供学生完成实验任务时参考。

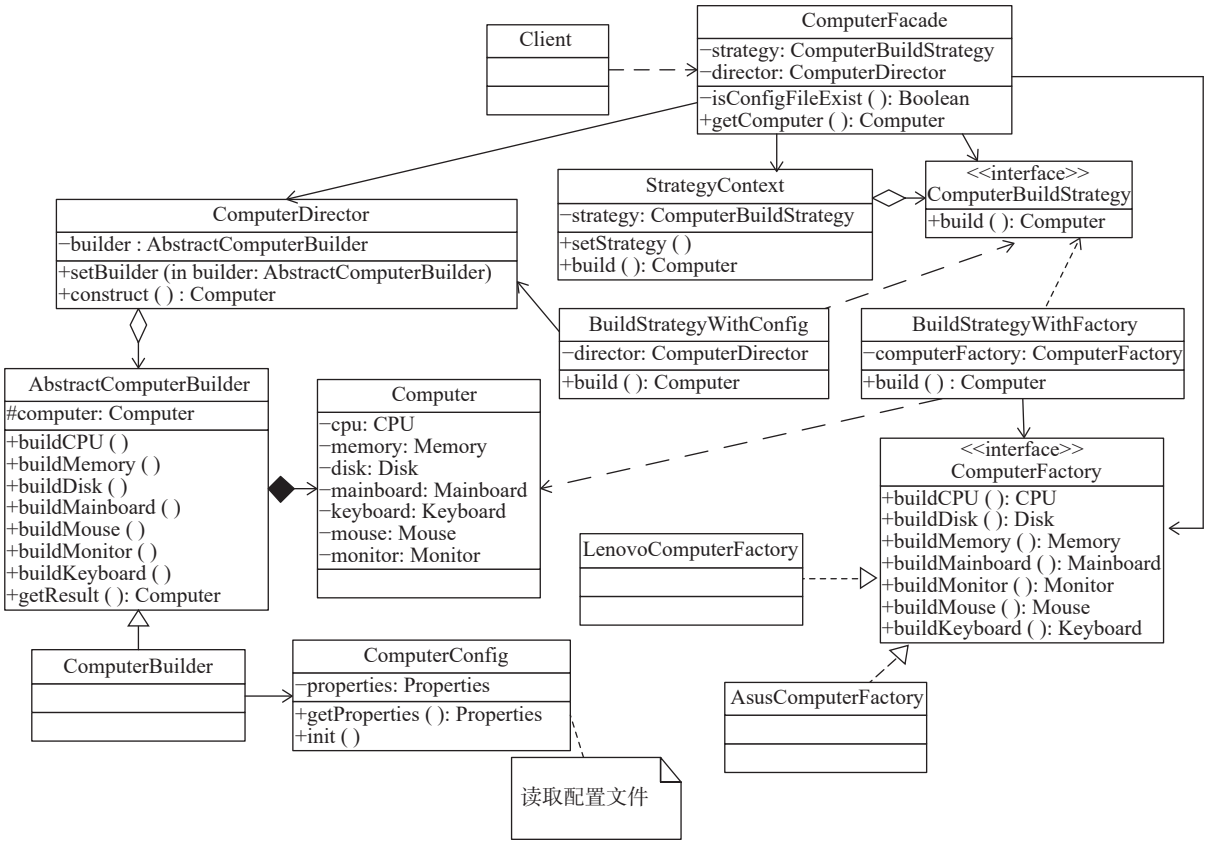

图 2　案例最终实现后的部分类图关系

2.2　案例教学过程

2.2.1　问题导入

以创建电脑为例，需求分析描述如下：假定组成电脑的核心部件包括 CPU、主板、内存、硬盘、显示器、键盘和鼠标等。每一种部件有多个不同的品牌，如内存有 GLOWAY、Netac、Unis 等多个品牌。一个电脑的所有组成部件可来自同一品牌，也可来自不同品牌。通过设计模式对上述需求进行设计，客户端通过设计得到电脑对象。

2.2.2　引入建造者设计模式

首先引导学生从需求分析的直接文字描述部分找出其中的名词：电脑、CPU、主板、内存、硬盘、显示器、键盘、鼠标和品牌等。通过分析名词之间的关系得知：电脑和组成部件之间属于关联关系；部件和品牌之间属于泛化关系；电脑属于复杂对象。客户端目的是得到电脑对象，因此需要创建出电脑，这符合创建型模式应用动机。进一步分析得知：无论每个部件采用哪个品牌，部件的属性是相同的，而且电脑组装的过程也是相同的，因此符合建造者设计模式的应用动机。

建造者设计模式属于典型的创建型模式，它的动机是将一个复杂对象的创建与表示分离，使得同样的创建过程可以得到不同的表示。

在确定使用建造者设计模式后，根据该模式组成的各角色职责开始设计、编码。建造者设计模式包含 4 个角色，分别为：复杂产品、抽象建造者、具体建造者和指挥者。首先引导学生编写名词对应的类，将 CPU、主板、内存、硬盘、显示器、键盘和鼠标等部件定义成接口，根据不同

品牌定义接口对应的实现类；再定义电脑类的各组成部件；然后定义抽象建造者类 AbstractComputerBuilder，组合电脑对象，定义创建各组成部件和返回电脑对象的方法；再根据某个电脑组成部件使用的品牌创建具体的建造者；最后再创建指挥者类 ComputerDirector，并在客户端创建出具体建造者和指挥者对象，通过指挥者对象得到电脑对象。至此第一次迭代结束，学生按照上述过程进行设计编码即可。

2.2.3　引入抽象工厂设计模式

从建造者设计模式的缺点出发，引导学生思考问题：电脑由多个部件构成，每个部件的具体表示又有多个品牌，每一个具体的建造者只能表示某一种部件使用某一种品牌的电脑；由于不同部件使用不同品牌会有多种组合情况，如果每一个具体建造者类型对应一种组合情况就会造成具体建造者类的个数膨胀(这本身也是建造者模式的缺点)。

引入新的需求：如何在减少具体建造者类型的情况下，还能表示电脑的创建过程。由于电脑各组成部件存在来自于同一品牌的可能，各部件可以理解为不同的产品等级结构，同一个品牌的各个部件来自一个产品族，符合抽象工厂设计模式应用动机。因此确定第二次迭代引入抽象工厂设计模式。

抽象工厂设计模式是典型的创建型模式，是一种为访问类提供一个创建一组相关或者相互依赖对象的接口，且访问类无须指定所要产品的具体类就能得到同族不同等级的产品[9]。引导学生根据抽象工厂设计模式的各组成角色进行设计与编码。首先新增抽象工厂角色，在这个接口中定义创建各组成部件的方法；然后再定义具体工厂类来创建同一品牌的各个部件。抽象产品和具体产品在第一次迭代时已经定义，所以本次迭代不再定义。

2.2.4　引入配置文件和反射机制

第二次迭代通过引入抽象工厂设计模式解决了电脑组成部件来自同一品牌的情况，减少了具体建造者类型的个数，但没有解决电脑组成部件来自不同品牌的情况。引入新的需求：如何减少不同部件来自不同品牌的具体建造者类型个数。

引导学生仔细阅读不同具体建造者类中关于部件创建方法的逻辑代码后，可发现不同具体建造者中的上述方法逻辑是相同的，只是创建的部件类型不相同，代码具有很高的冗余度。因此减少具体建造者类型的解决方法就是把电脑部件的类型写入配置文件，通过读取配置文件中定义的类型名称再结合反射机制将具体的部件对象创建出来。

配置文件是目前软件系统普遍采用的一种简化代码结构、提高扩展性的有效手段。在软件过程的各个阶段都存在利用配置文件对软件进行配置的操作，从而实现对软件特性的设置和个性化定制[10]。

Java 反射机制能够在程序运行时动态创建一个未知类的对象，并且能够操作这个对象的属性和方法[11]。通过反射可以提高 Java 程序的灵活性和动态性，是中间件、框架技术底层的重要支撑技术[12]。

配置文件中存放的只是对系统运行参数取值的设置，具体使用这些设置还需要对文件进行解析、读取设置并根据取值动态创建出对应的对象。目前，Java 软件系统中常用的配置文件有文本文件和 XML 文件两种形式。文本类型配置文件可以通过 Properties 工具类进行解析；XML 文件可以通过 DOM、SAX、JDOM 和 DOM4J 这 4 种常用方式进行解析[13]。

根据上述解决方案对原有设计进行变更，以文本类型配置文件为例，某个电脑组成部件的配置信息写入配置文件，代码如下所示：

```
computer.memory=cn.edu.xync.entity.GlowayMemory
computer.cpu=cn.edu.xync.entity.IntelCPU
computer.disk=cn.edu.xync.entity.AigoDisk
computer.mainboard=cn.edu.xync.entity.AsusMainboard
computer.monitor=cn.edu.xync.entity.LenovoMonitor
computer.keyboard=cn.edu.xync.entity.DeLUXKeyboard
computer.mouse=cn.edu.xync.entity.A4TechMouse
```

具体建造者类只需保留一个，同时修改其中创建各组成部件的代码。以创建 CPU 部件为例，ComputerBuilder 类中的 buildCPU()方法的代码如下所示：

```
public void buildCPU() {
CPU cpu=null;
try {
```

```
Object obj=config.getProperties( ) .get( "computer.cpu");
    if(obj!=null)
    cpu = (CPU)Class.forName((String)obj).newInstance();
      else
       throw new ComputerConfigException("请检查属性:computer.cpu 是否在配置文件中正确定义");
      } catch (InstantiationException e) {
        e.printStackTrace();
      } catch (IllegalAccessException e) {
        e.printStackTrace();
      } catch (ClassNotFoundException e) {
         throw new ComputerConfigException("配置文件中 computer.cpu 对应的字节码没有找到,请检查对应的实现类是否定义");
      }
      computer.setCpu(cpu);
      System.out.println("CPU 部件组装完成");
    }
```

2.2.5 引入策略设计模式

经过第三次迭代，利用配置文件+反射机制解决了减少具体建造者个数的问题。此时从健壮性角度出发，引导学生思考一个问题：某种特殊情况下，如果配置文件不存在，现有的程序就无法创建出电脑对象(出现异常)。提出新的需求：在缺少配置文件的情况下，程序依然能够创建出电脑对象，在缺少配置文件的情况下可以利用某个工厂对象创建电脑对象。

在新的需求问题下引导学生进行分析：如果配置文件存在就从配置文件中通过建造者设计模式创建对象；否则就通过抽象工厂设计模式创建对象。这符合策略设计模式应用动机，所以本次迭代引入策略设计模式。策略设计模式属于典型的对象行为型模式，核心思想是将算法的不同实现通过具体类独立封装，使它们可以相互替换[14–15]。使用策略设计模式后，引导学生按如下过程进行设计编码：新增策略接口 ComputerBuildStrategy，定义 build()方法，返回类型为 Computer。新增两个策略接口实现类 BuildStrategyWithConfig 和 BuildStrategyWithFactory，前者通过 ComputerDirector 读取配置文件来创建电脑；后者通过 ComputerFactory 来创建电脑。但是后者 build()方法的逻辑是：首先创建 Computer 对象，然后通过抽象工厂设计模式创建各等级结构再赋值给对应的 set()方法。新增 StrategyContext 作为策略使用的上下文环境类，定义 build()方法通过具体策略对象的 build()方法返回 Computer 对象。

通过上述变更后，客户端创建电脑对象的代码如下所示：

```
InputStream stream=Test.class.geClassLoader( ) .getResourceAsStream("computerconfig.properties");
ComputerBuildStrategy strategy=null;
    if(stream!=null)
{
    ComputerDirector director=new ComputerDirector(new ComputerBuilder());
    strategy=new BuildStrategyWithConfig(director);
}
else{
    ComputerFactory factory=new LenovoComputerFactory();
    strategy=new BuildStrategyWithFactory(factory);
}
    Computer computer=strategy.build();
```

2.2.6 引入外观设计模式

经过第四次迭代，利用策略设计模式解决了无论配置文件是否存在都能正确创建出电脑对象的问题。至此从封装性角度引导学生思考问题：在客户端创建电脑对象的代码量较多，需要判断配置文件是否存在，如果存在要创建 ComputerBuilder 对象、ComputerDirector 对象和 BuildStrategyWithConfig 对象，才能得到 Computer 对象；如果不存在要创建一个具体的电脑工厂对象和 BuildStrategyWithFactory 对象，才能得到 Computer 对象。这样的逻辑无疑将电脑创建细节暴露给了客户端。提出新的需求：向客户端屏蔽电脑创建的细节并减少额外创建逻辑。

在新的需求问题下引导学生进行分析：从配置文件创建 Computer 对象使用建造者设计模式，缺少配置文件使用抽象工厂设计模式，这两种不同的创建方式可以看作是两个不同的子系统，通过提供一个类把两个子系统封装起来，这样可以向客户端屏蔽子系统创建对象的细节[16]。这符合外观设计模式应用动机，所以本次迭代需要引入

外观设计模式。

使用外观设计模式后，引导学生按如下过程进行设计编码：新增外观类 ComputerFacade 聚合 StrategyContext、ComputerDirector 和 ComputerFactory 这 3 个对象。ComputerDirector 和 ComputerFactory 可以看作是两个子系统，分别提供两种不同方式的 Computer 对象的创建过程。ComputerFacade 通过提供 isConfigFileExist()方法来判断当前配置文件是否存在；通过 initStrategy()方法依据 isConfigFileExist()方法的结果来实例化具体的策略类；通过 getComputer()方法依赖策略上下文对象的 build()方法提供创建的对象。

通过上述变更后，客户端创建电脑对象的代码如下所示：

```
ComputerFacade facade=new ComputerFacade();
Computer computer=facade.getComputer();
```

与上一次迭代后客户端创建电脑对象的代码相比较，引入外观设计模式后，客户端得到电脑对象只需两行代码即可，达到了屏蔽电脑对象创建细节和降低代码量的目的。依据某一次具体的配置文件，程序运行结果如图 3 所示。

```
本次Computer对象部件信息从配置文件中进行读取并创建
++++++++++++++++++++++++++++++++++++++++++++++++
CPU组装完成
Memory组装完成
Disk组装完成
Mainboard组装完成
Keyboard组装完成
Monitor组装完成
Mouse组装完成
++++++++++++++++++++++++++++++++++++++++++++++++
Intel CPU is working
Gloway Memory is working
Aigo Disk is working
Asus Mainboard is working
Lenovo Monitor is working
DeLUX Keyboard is working
A4Tech Mouse is working
```

图 3　案例运行结果

3　实验任务实施与效果分析

3.1　实验任务分解

依据上述教学设计过程，按照迭代次序将教学案例分解后作为实验教学任务，如表 2 所示。

上述实验教学任务的分析和实施都是在学生能力范围内可以完成的，实验内容都与设计模式有关。每 3~4 名学生组成一个小组，以小组为单位进行实验任务的分析、设计、编码和测试运行。小组成员之间充分沟通和交流，确保组内每一位学生都能完成实验任务。按照迭代次序把任务结合起来就是一个有机整体，最终形成的设计能够满足开闭原则。如电脑组成部件的品牌有新增，只需创建具体品牌的部件类并在配置文件中进行配置即可，其余代码均无须修改；如果要更换电脑组成部件的品牌只需修改配置文件即可；如果电脑整体有新增品牌，只需创建对应品牌的部件和具体工厂即可。

表 2　实验任务分解表

序号	任务描述	知识点	难度
①	通过分析建立电脑及其各组成部件的对应类，每个部件可以有多个品牌，无论每个部件采用什么品牌，电脑的创建过程相同	建造者设计模式	难
②	存在电脑各部件来自同一品牌的情况，为了减少具体建造者类的个数，引入抽象工厂设计模式	抽象工厂设计模式	中等
③	减少不同品牌部件组成电脑的具体建造者类的个数，引入配置文件，通过反射机制实例化电脑各组成部件	配置文件+反射机制	难
④	无论配置文件是否存在，客户端都能得到电脑对象	策略设计模式	难
⑤	向客户端屏蔽电脑创建细节，减少客户端创建电脑对象的代码量	外观设计模式	中等

3.2　效果分析

本案例应用于计算机学院 2021 级软件工程专业的软件设计与体系结构课程教学中，共有 44 名学生参与了课程教学与实验，教学效果表现良好。在课堂教学中，学生能够提高注意力，跟随教师分析思路，认真做好笔记，并积极沟通交流，表现出刻苦钻研的精神。在实验任务中，各小组最终都能按照案例分析过程编码实现并且结果运行正确，小组内部体现了团结合作精神。少部分学生在反射应用和模式融合过程中的思路还不是很清晰，通过反复观看教学视频以及在小组成员、老师的帮助下最终也能完成实验任务。选取单一实验案例和本文综合实验案例的教学效果进行对比，如表 3 所示。其中兴趣度和能力提升度通过问卷调查得到，平均成绩为任课教师根据学生提交实验任务的评阅结果得到。其中单一实验案例的平均成绩为 9 次实验作业得分的平均值，综合实验案例的平均成绩为 5 个任务得分的平均值。通过表 3 数据得知，本综合实验案例能够相对提高学生的学习兴趣和效果。

表 3 教学效果对比表

案例类型	兴趣度	能力提升度	平均成绩
单一实验	79	72	77
综合实验	82	76	81

少数学生在第 5 次迭代后的基础上，提出了新的需求：如何在不重复创建 ComputerFacade 对象的前提下使客户端得到多个不同的 Computer 对象。经分析在引入原型设计模式后解决了问题，实现了第 6 次迭代，对本教学案例进行了扩展，达到了融会贯通的效果。

4 结束语

小到业务功能、大到软件架构，合理应用设计模式能够提高软件的扩展性和维护性。本文设计了一个融合多种设计模式的教学案例，在学生充分理解和掌握当前设计和编码的基础上，从显性和隐性需求两个方面不断提出新的问题，逐步引导学生思考，通过融合新的设计模式和相关技术为新的问题提供解决方案，不断加深学生对设计模式的理解和面向对象技术的综合应用能力，并为后续学习理解框架原理和阅读框架源码奠定了基础。对于实际软件项目中有类似本文电脑这类复杂对象的表示和创建的场景，图 2 的设计类图有一定的借鉴作用。下一步将继续按照本文这种迭代过程，在依托实际软件项目背景下，设计融合其他设计模式的教学案例，进一步丰富课程综合案例库。

参 考 文 献

[1] 欧阳宏基, 葛萌, 李红. 面向工程认证的地方本科院校软件工程专业教学改革研究[J]. 产业与科技论坛, 2024, 23(5): 214−218.

[2] 张建, 刘博, 朱青, 等. 软件设计与体系结构课程内容建设及创新探索[J]. 计算机教育, 2022(7): 62−66.

[3] 段喜龙, 邬志红. 基于 BOPPPS 的面向对象程序设计课程线上线下混合教学[J]. 高教学刊, 2023, 9(23): 104−107.

[4] 司慧琳, 李素. 面向对象程序设计课程建设与改革研究[J]. 软件导刊, 2023, 22(6): 67−69.

[5] 林欣欣, 郭元术, 运杰伦, 等. 基于模板方法与抽象工厂的复合模式[J]. 计算机系统应用, 2020, 29(6): 218−223.

[6] 尹梓名, 周雷, 郑建立. 以“软件设计模式” 促进“面向对象程序设计” 课程教学方法研究[J]. 软件, 2019, 40(8): 216−219.

[7] 刘战东, 李勇. 基于雨课堂的设计模式课程教学改革[J]. 计算机教育, 2020(9): 174−177.

[8] 薛镇镇, 潘杰, 王国明. PBL 教学法在结构化学休克尔分子轨道理论教学中的应用[J]. 大学化学, 2023, 38(12): 274−278.

[9] 杨萍, 文信峰. 基于抽象工厂模式的机载总线数据通用化处理方法研究[J]. 计测技术, 2022, 42(4): 35−41.

[10] 陈艳, 叶宏杰, 陈伟. 软件系统配置研究综述[J]. 计算机系统应用, 2021, 30(7): 1−12.

[11] 张胜楠. 基于 Java 反射和 Fel 计算引擎动态导出 Excel 的实现[J]. 现代计算机, 2022, 28(12): 102−106.

[12] 丁春玲, 路志强, 彭伟. Java 反射机制在数据持久层轻量级 ORM 框架中的应用研究[J]. 西安文理学院学报(自然科学版), 2017, 20(1): 39−42.

[13] 刘宁. 如何利用 Java 语言进行 XML 编程[J]. 数字技术与应用, 2016(8): 243.

[14] 熊风光, 张元, 况立群. 面向对象程序设计课程教学改革[J]. 计算机教育, 2021(9): 86−90.

[15] 虞泉. 策略设计模式在实验耗材 Web 系统中的应用[J]. 软件导刊, 2020, 19(4): 189−193.

[16] 刘伟. 设计模式[M]. 2 版. 北京: 清华大学出版社, 2023.

编辑 葛晋