

宜宾学院

软件设计模式与体系结构设计报告书

学 院: 人工智能与大数据学部 班 级: 2018 级 9 班

学生姓名: 杨雪 学 号: 200109327

设计地点 (单位) 6305

设计题目: 软件设计模式与软件设计-实验 6

完成日期: 2021 年 6 月 1 日

1 迭代器

1.1 题干要求

现有的商家菜单系统的菜单由 Menu 类表示，主要属性为：处理价格，菜单名和描述。其主要由 4 种数据结构实现：传统数组，arrayList, List, Map。现假设要实现一个美团，该系统要对接现有的商家菜单系统，但又不能更改现有的商家菜单系统，请使用迭代器模式，完成商家系统的对接。需要编写代码的 UML 如图 1-1 所示。

1.2 设计思路

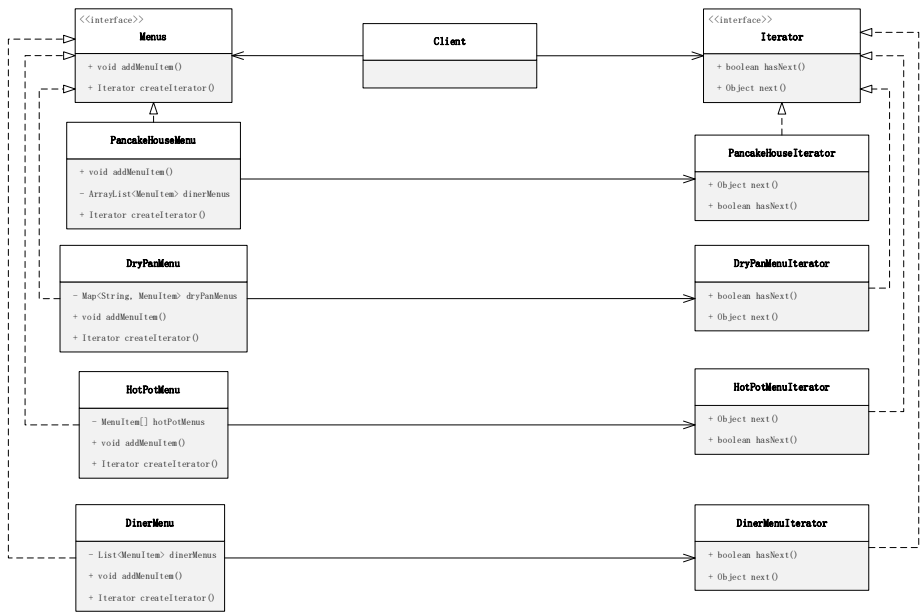


图 1-0-1 迭代器 UML 图

1.3 编码与测试

编码：

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.UUID;

/**
```

* 现有的商家菜单系统的菜单由 **Menu** 类表示，主要属性为：处理价格，菜单名和描述。其主要由 4 种数据结构实现：传统数组，**arrayList**, **List**, **Map**。

* 现假设要实现一个美团，该系统要对接现有的商家菜单系统，但又不能更改现有的商家菜单系统，请使用迭代器模式，完成商家系统的对接。

```
*/  
/*
```

```
/*
```

```
* 菜单基类
```

```
*/
```

```
class Menu {  
    private String name; // 菜单名  
    private double price; // 处理价格  
    private String description; // 描述  
  
    public Menu(String name, double price, String description) {  
        super();  
        this.name = name;  
        this.price = price;  
        this.description = description;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public double getPrice() {  
        return price;  
    }  
}
```

```

    public void setPrice(double price) {
        this.price = price;
    }

    public String getDescription() {
        return description;
    }

    public void setdescription(String description) {
        this.description = description;
    }
}

/*
 * 菜单系统迭代器
 */

interface Iterator {
    // 判断当前元素是否存在
    boolean hasNext();

    // 获取当前存在的元素，将元素指针指向下一个
    Object next();
}

/*
 * 干锅迭代器
 */

class DrypanIterator implements Iterator {
    List<Menu> dryPan;
    int position = 0;
    public DrypanIterator(List<Menu> dryPan) {
        this.dryPan = dryPan;
    }
}

```

```

@Override

public boolean hasNext() {
    if (dryPan.size() != 0 && position < dryPan.size() && dryPan.get(position) != null)
    {
        return true;
    }
    return false;
}

@Override

public Object next() {
    return dryPan.get(position++);
}
}

/*
 * 火锅迭代器
 */

class HotpotIterator implements Iterator {
    ArrayList<Menu> hotPot;

    int position = 0;

    public HotpotIterator(ArrayList<Menu> hotPot) {
        this.hotPot = hotPot;
    }

    @Override

    public boolean hasNext() {
        return false;
    }
}

```

```

        @Override
        public Object next() {
            return hotPot;
        }
    }

}

/*
 * 烤肉迭代器
 */

class BarbecueIterator implements Iterator {
    Menu[] barbecue;

    int position = 0;

    public BarbecueIterator(Menu[] barbecue) {
        this.barbecue = barbecue;
    }

    @Override
    public boolean hasNext() {
        if (position < barbecue.length && barbecue[position] != null ){
            return true;
        }

        return false;
    }

    @Override
    public Object next() {
        return barbecue[position++];
    }
}

```

```

/*
 * 甜点饮品迭代器
 */

class DessertDrinksIterator implements Iterator {
    Map<String, Menu> dessertDrinks;
    java.util.Iterator<Map.Entry<String, Menu>> it ;
    @SuppressWarnings("rawtypes")
    Map.Entry entry;

    public DessertDrinksIterator(Map<String, Menu> dessertDrinks) {
        this.dessertDrinks = dessertDrinks;
        it = dessertDrinks.entrySet().iterator();
    }

    @Override
    public boolean hasNext() {
        return it.hasNext();
    }

    @Override
    public Object next() {
        Map.Entry<String, Menu> entry = it.next();
        return entry.getValue(); }
}

/*
 * 干锅
 */

class DryPan {

```

```

List<Menu> dryPan = new LinkedList<Menu>();

    public void addMenu(String name, String describe, double price) {
        dryPan.add(new Menu(name, price, describe));
    }

    public Iterator createIterator() {
        return new DrypanIterator(dryPan);
    }
}

/*
 * 火锅
 */

class HotPot {
    ArrayList<Menu> hotPot = new ArrayList<Menu>();

    public void addMenu(String name, String describe, double price) {
        hotPot.add(new Menu(name, price, describe));
    }

    public Iterator createIterator() {
        return new HotpotIterator(hotPot);
    }
}

/*
 * 烤肉
 */

class Barbecue{
    // 烤肉菜单菜品最大数量
    private static final Integer MENU_ITEM_MAX = 10;

    int numberOfItems = 0;

```



```

// 烤肉菜单
Menu[] barbecue = new Menu[MENU_ITEM_MAX];

public void addMenu(String name, String description, double price) {
    if(numberOfItems < MENU_ITEM_MAX) {
        barbecue[numberOfItems++] = new Menu(name, price, description);
    }
}

public Iterator createIterator() {
    return new BarbecueIterator(barbecue);
}
}

/*
 * 甜点饮品
 */

class DessertDrinks{
    Map<String, Menu> dessertDrinks = new HashMap<String, Menu>();

    public void addMenu(String name, String description, double price) {
        dessertDrinks.put(UUID.randomUUID().toString(), new Menu(name, price,
description));
    }

    public Iterator createIterator() {
        return new DessertDrinksIterator(dessertDrinks);
    }
}

class Meituan {
    DryPan dryPan;
    HotPot hotPot;
    Barbecue barbecue;
}

```

```

DessertDrinks dessertDrinks;

public Meituan(DryPan dryPan, HotPot hotPot, Barbecue barbecue, DessertDrinks
dessertDrinks){
    this.dryPan = dryPan;
    this.hotPot = hotPot;
    this.barbecue = barbecue;
    this.dessertDrinks = dessertDrinks;
}

public Meituan(DryPan dryPan) {
    this.dryPan = dryPan;
}

public Meituan(HotPot hotPot) {
    this.hotPot = hotPot;
}

public Meituan(Barbecue barbecue) {
    this.barbecue = barbecue;
}

public Meituan(DessertDrinks siChuanRestaurant) {
    this.dessertDrinks = siChuanRestaurant;
}

public void printMenu() {
    Iterator dryPanIterator = dryPan.createIterator();
    Iterator hotPotIterator = hotPot.createIterator();
    Iterator barbecueIterator = barbecue.createIterator();
    Iterator dessertDrinksIterator = dessertDrinks.createIterator();

    printMenuItem(dryPanIterator);
    printMenuItem(hotPotIterator);
    printMenuItem(barbecueIterator);
    printMenuItem(dessertDrinksIterator);
}

```

```

    }

    private void printMenuItem(Iterator menuIterator) {
        while (menuIterator.hasNext()) {
            Menu next = (Menu) menuIterator.next();
            System.out.print(next.getName() + " " + next.getDescription() + " 价格: "
                + next.getPrice() + "\n");
        }
        System.out.println();
    }
}

public class IteratorTest {
    public static void main(String[] args) {
        // 干锅菜单
        DryPan dryPan = new DryPan();
        dryPan.addMenu("干锅鸭掌", "糯香味道的鸭掌", 56.00);
        dryPan.addMenu("干锅虾", "新鲜的虾", 45.00);
        dryPan.addMenu("干锅排骨", "新鲜排骨", 50.00);

        // 火锅菜单
        HotPot hotPot = new HotPot();
        hotPot.addMenu("99 元双人套餐", "麻辣牛肉、千层肚、肥牛、撒尿牛丸...",
99.00);
        hotPot.addMenu("放心优享 4 人餐", "串串、油炸牛肉、油炸五花肉、土豆...",
143.00);
        hotPot.addMenu("1 元单人餐", "白雪冰汤圆", 0.70);

        // 烤肉菜单
        Barbecue barbecue = new Barbecue();

```

```

        barbecue.addMenu("特色烤肉自助", "有烤肉类, 海鲜类, 寿司类, 还有各种素
菜、饮品哦", 67.00);

        barbecue.addMenu("晚餐 4 大 2 小", "有烤肉类, 海鲜类, 寿司类, 还有各种素
菜、饮品哦", 288.00);

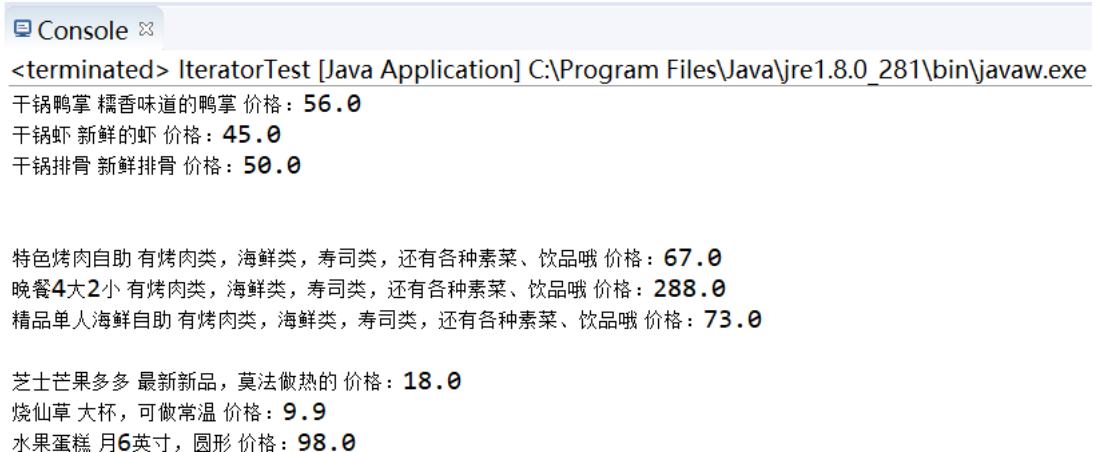
        barbecue.addMenu("精品单人海鲜自助", "有烤肉类, 海鲜类, 寿司类, 还有各种
素菜、饮品哦", 73.00);

// 甜点饮品菜单
DessertDrinks dessertDrinks = new DessertDrinks();
dessertDrinks.addMenu("水果蛋糕", "月 6 英寸, 圆形", 98.00);
dessertDrinks.addMenu("烧仙草", "大杯, 可做常温", 9.90);
dessertDrinks.addMenu("芝士芒果多多", "最新新品, 莫法做热的", 18.00);

Meituan meituan = new Meituan(dryPan, hotPot, barbecue, dessertDrinks);
meituan.printMenu();
    }
}

```

测试:



```

Console
<terminated> IteratorTest [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe
干锅鸭掌 糯香味的鸭掌 价格: 56.0
干锅虾 新鲜的虾 价格: 45.0
干锅排骨 新鲜排骨 价格: 50.0

特色烤肉自助 有烤肉类, 海鲜类, 寿司类, 还有各种素菜、饮品哦 价格: 67.0
晚餐4大2小 有烤肉类, 海鲜类, 寿司类, 还有各种素菜、饮品哦 价格: 288.0
精品单人海鲜自助 有烤肉类, 海鲜类, 寿司类, 还有各种素菜、饮品哦 价格: 73.0

芝士芒果多多 最新新品, 莫法做热的 价格: 18.0
烧仙草 大杯, 可做常温 价格: 9.9
水果蛋糕 月6英寸, 圆形 价格: 98.0

```

图 1-0-2 迭代器测试截图

2 组合模式

2.1 题干要求

现代计算机一般分为具有以下部件：键盘、显示器、机箱、鼠标。而机箱内部包括了主板、硬盘、电源等。主板是主机的核心，其上面一般又插入了 CPU、内存、显卡等设备，请用组合模式描述一台计算机，并尝试根据自己的对游戏的知识，实现一台电脑，能进行“绝地求生”游戏。UML 如图 1-2 所示。

2.2 设计思路

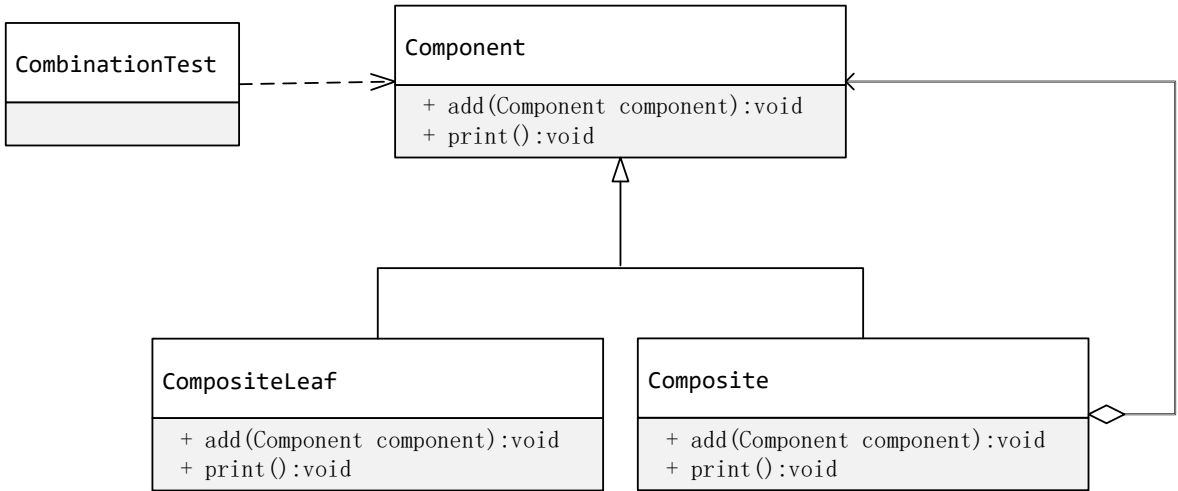


图 2-0-1 组合模式 UML 图

2.3 编码与测试

编码：

/**

* 现代计算机一般分为具有以下部件：键盘、显示器、机箱、鼠标。而机箱内部包括了主板、硬盘、电源等。

* 主板是主机的核心，其上面一般又插入了CPU、内存、显卡等设备，请用组合模式描述一台计算机，并尝试根据自己的对游戏的知识，实现一台电脑，能进行“绝地求生”游戏。

* @author Moppet

*

*/

```

abstract class Component{
    protected String name;
    abstract void add(Component component);
    abstract void print();
}

class Composite extends Component {

    public Composite(String name) {
        this.name = name;
    }

    List<Component> compositeList = new ArrayList<Component>();

    @Override
    void add(Component component) {
        compositeList.add(component);
    }

    @Override
    void print() {
        System.out.println(name);
        for(Component item: compositeList) {
            item.print();
        }
    }
}

class Leaf extends Component {

    public Leaf(String name) {
        this.name = name;
    }
}

```

```

    }

    List<Component> compositeList = new ArrayList<Component>();

    @Override
    void add(Component component) {}

    @Override
    void print() {
        System.out.println(name);
        for(Component item: compositeList) {
            item.print();
        }
    }
}

public class CombinationTest {
    public static void main(String[] args) {
        Composite games = new Composite("玩绝地求生游戏的配置:");
        Leaf keyboard = new Leaf("键盘: 北通K1键盘");
        Leaf monitor = new Leaf("显示器: 飞利浦 (PHILIPS) 27M6FJMB 27英寸 曲面屏2K/144Hz电竞显示器");
        Composite chassis = new Composite("机箱: 航嘉MVP2机箱");
        Leaf mouse = new Leaf("鼠标: 罗技GPW无线鼠标");

        Composite motherboard = new Composite("主板: 华硕Z390-PLUS GAMING主板");
        Leaf harddisk = new Leaf("硬盘: 三星750EVO 250G SATA3 SSD固态硬盘");
        Leaf powersupply = new Leaf("电源: 1200w");

        Leaf cpu = new Leaf("CPU: Intel Core i3-4340 / AMD FX-6300");
        Leaf graphicscard = new Leaf("显卡: 索泰Geforce GTX1060-6GD5 X-GAMING

```

```

OC");

    Leaf networkcard = new Leaf("网卡: 基于PCI的无线网卡");

    games.add(keyboard);
    games.add(monitors);
    games.add(chassis);
    games.add(mouse);

    chassis.add(motherboard);
    chassis.add(harddisk);
    chassis.add(powersupply);
    motherboard.add(cpu);
    motherboard.add(graphicscard);
    motherboard.add(networkcard);

    games.print();

}
}

```

测试:



Console

<terminated> CombinationTest [Java Application] C:\Program Files\Java\jre1.8.0_281\bin\javaw.exe

玩绝地求生游戏的配置:

键盘: 北通K1键盘

显示器: 飞利浦(PHILIPS) 27M6FJMB 27英寸 曲面屏2K/144Hz电竞显示器

机箱: 航嘉MVP2机箱

主板: 华硕Z390-PLUS GAMING主板

CPU: Intel Core i3-4340 / AMD FX-6300

显卡: 索泰Geforce GTX1060-6GD5 X-GAMING OC

网卡: 基于PCI的无线网卡

硬盘: 三星750EVO 250G SATA3 SSD固态硬盘

电源: 1200w

鼠标: 罗技GPW无线鼠标

图 2-2 组合模式测试截图