

目录

第 12 章 面向对象程序设计 1

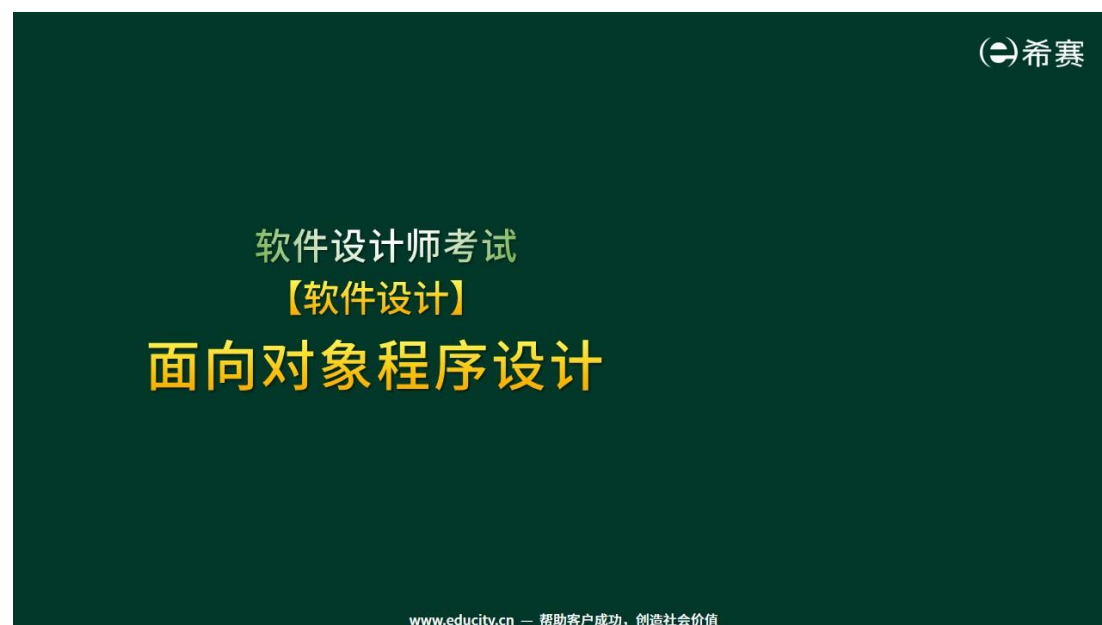
 12.1 内容提要1

 12.2 C++语法要点2

 12.3 Java 语法要点2

 12.4 设计模式程序实现5

第 12 章 面向对象程序设计



12.1 内容提要



12.2 C++语法要点

12.3 Java 语法要点

 课程内容提要

希赛

➤ Java语法要点

www.educity.cn — 帮助客户成功，创造社会价值

 **JAVA – 类的定义**

希赛

类的定义格式如下：
[import包]
[类修饰符] class xxxclass [extends超类][implements 接口] {
 public:
 公有数据成员或公有函数成员的定义；
 protected:
 保护数据成员或保护函数成员的定义；
 private:
 私有数据成员或私有函数成员的定义；
}

说明：

- import包：引入包中的类。
- 类修饰符：主要有四个修饰符，public、abstract、final、private。
- class为关键字，xxxclass为类名，命名遵循Java标识符的命名规则。
- extends为继承关键字，implements为接口关键字。

www.educity.cn — 帮助客户成功，创造社会价值



JAVA – 类的定义

例：

```
class CashDiscount implements CashSuper {
    private double moneyDiscount;    // 折扣率
    public CashDiscount(double moneyDiscount) {
        this.moneyDiscount = moneyDiscount;
    }
}

class Whip extends CondimentDecorator { //奶泡
    private final int WHIP_PRICE = 8;
    public Whip (Beverage beverage) { this.beverage = beverage; }
    public String getDescription () {
        return beverage.getDescription () + ", Whip";
    }
    public int cost() { return WHIP_PRICE + beverage.cost(); }
}
```

www.educity.cn — 帮助客户成功，创造社会价值



JAVA – 类的定义

```
import java. util.*;
(1) class Beverage {    //饮料
    String description = "Unknown Beverage";
    public String getDescription () {return description;}
    public abstract int cost () ;
}
```

(1) **abstract**

www.educity.cn — 帮助客户成功，创造社会价值



JAVA – 类的定义

```
class Department{/*代码省略*/}

class SqlserverDepartment _ (1) _{/*代码省略*/}
```

```
(2) class Shape{
    abstract public void draw()
}

class Rectangle extends Shape{
    (3) _{/*代码省略*/}
}
```

www.educity.cn — 帮助客户成功，创造社会价值



JAVA – 接口的定义


接口的定义格式如下：

```
[修饰符] interface 接口名 [extends 父接口名列表]{
    [public] [static] [final] 常量;
    [public] [abstract] 方法;
}
```

```
interface IFactory{}

class SqlServerFactory implements IFactory
```

www.educity.cn — 帮助客户成功，创造社会价值

 **JAVA – 接口的定义**

希赛

(1) **Drawing**{

(2) ; (1) **interface**

(3) ; (2) **void drawLine(double x1, double y1, double x2 ,double y2)**

 (3) **void drawCircle (double x, double y, double r)**

}

class V1Drawing **implements** **Drawing**{

 public void drawLine(double x1, double y1, double

 x2 ,double y2){/*代码省略*/}

 public void drawCircle (double x, double y, double r){ (4) ;}

}

www.educity.cn — 帮助客户成功，创造社会价值

12.4 设计模式程序实现

 **课程内容提要**

希赛

➤ **注意事项及代码填空技巧**

www.educity.cn — 帮助客户成功，创造社会价值

注意事项

一、大小写问题

Java语言大小写敏感，如：定义接口关键字为interface，写成Interface，则算错。C++也如此。

二、什么时候要加this

1、set方法中，引用对象的变量

```
class EnvironmentData implements (1) {
    private float temperature, humidity, cleanness;
    public void setMeasurements(float temperature, float humidity, float
cleanness) {
        this.temperature = temperature; 总不能写成: temperature = temperature 吧。
        ...}
    ...}
```

2、带参数的构造函数

```
class CashDiscount implements CashSuper {
    private double moneyDiscount; // 折扣率
    public CashDiscount(double moneyDiscount) { //构造函数
        this.moneyDiscount = moneyDiscount;
    }
}
```

www.educity.cn — 帮助客户成功，创造社会价值

注意事项

三、父类与子类（接口与实现类）之间方法的统一

- 1、对于方法填空，主要依据父类与子类（接口与实现类）之间方法的统一；
- 2、注意对于接口抽象方法，JAVA没有方法体（无花括号），C++纯虚函数写法。

四、注意函数调用

- 1、代码中可能出现方法体的缺失，这一类填空可以根据函数传参，从而判断该参数类型、该类型对象能够调用的方法，从而填空；
- 2、代码中可能出现方法形参的缺失，这一类填空需要根据方法体判断该方法需要使用的参数名，根据调用的过程判断该参数的类型。

```
class Book implements LibraryItemInterfac{
    ...
    public void accept(LibraryVisitor visitor){
        (5) ; //根据LibraryVisitor类型对象visitor，能够调用的方法只有visit(Book p_book)
        //和(Article p_article)，根据当前类为Book，这里visitor.visit(this)。
    }
}
```

www.educity.cn — 帮助客户成功，创造社会价值



代码填空技巧

希赛

一、关键字填空

经常出现关键字填空，比如abstract、extends、implements等，注意不要出现拼写错误。

二、缺失方法填空

- 1、注意父类与子类、接口与实现类之间的统一，经常出现对应方法的缺失，可以根据上下文进行填写，注意接口方法无方法体，其他返回值类型、参数列表应该保持一致；
- 2、注意构造函数的写法，get和set方法的应用；
- 3、如果JAVA自带的一些类型的方法，需要靠平常的积累，建议多做真题进行积累。

三、实例化填写

注意实例化的参数选择。

四、注意函数调用（参数列表、参数类型、对应参数能够调用的方法）

五、在软件设计师考试中，程序设计题最难的地方在于与设计模式的结合，建议多做真题进行积累。



软件设计-访问者模式

希赛

阅读下列说明和Java代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

【说明】

某图书管理系统中管理着两种类型的文献：图书和论文。现在要求统计所有馆藏文献的总页码（假设图书馆中有一本540页的图书和两篇各25页的论文，那么馆藏文献的总页码就是590页）。采用Visitor（访问者）模式实现该要求，得到如图6-1所示的类图。

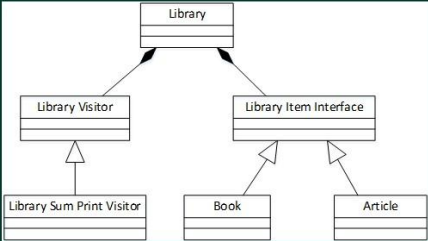
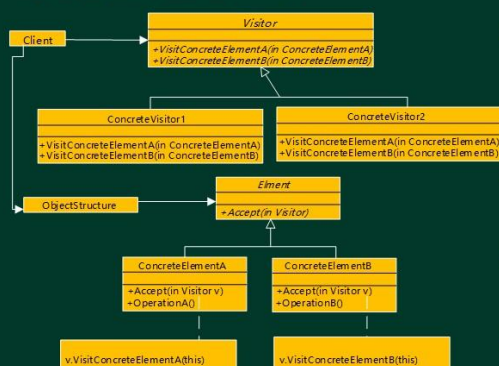


图6-1

行为性模式 - 访问者模式 (Visitor)

访问者模式的意图是：表示一个作用于某对象结构中的各元素的操作，使得在不改变各元素的类的前提下定义作用于这些元素的新操作。



- Visitor: 抽象访问者，为对象结构类中每一个ConcreteElement的类声明一个Visit操作。
- ConcreteVisitor: 具体访问者，实现每个由Visitor声明的操作。
- Element: 元素，定义一个Accept操作，它一个访问者为参数。
- ConcreteElement: 抽象具体元素，实现Accept操作，该操作以一个访问者为参数。
- ObjectStructure: 对象结构类。

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

```

【Java 代码】
import java.util.*;
interface LibraryVisitor {
    (1) ;
    (2) ;
    void printSum();
}
class LibrarySumPrintVisitor implements LibraryVisitor { //打印总页数
    private int sum = 0;
    public void visit(Book p_book) {
        sum = sum + p_book.getNumberOfPages();
    }
    public void visit(Article p_article) {
        sum = sum + p_article.getNumberOfPages();
    }
    public void printSum(){
        System.out.println("SUM = " + sum);
    }
}
interface LibraryItemInterface {
    (3) ;
}
    
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(=) 希赛

```
class Article implements LibraryItemInterface{
    private String m_title; //论文名
    private String m_author; //论文作者
    private int m_start_page;
    private int m_end_page;
    public Article(String p_author, String p_title,int p_start_page,int p_end_page){
        m_title=p_title;
        m_author= p_author;
        m_end_page=p_end_page;
    }
    public int getNumberOfPages(){
        return m_end_page - m_start_page;
    }
    public void accept(LibraryVisitor visitor){
        (4) ;
    }
}
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(=) 希赛

```
class Book implements LibraryItemInterface{
    private String m_title; //书名
    private String m_author; //书作者
    private int m_pages; //页数
    public Book(String p_author, String p_title,int p_pages){
        m_title= p_title;
        m_author= p_author;
        m_pages= p_pages;
    }
    public int getNumberOfPages(){
        return m_pages;
    }
    public void accept(LibraryVisitor visitor){
        (5) ;
    }
}
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

【参考答案】

- (1) void visit(Book p_book)
- (2) void visit(Article p_article)
- (3) void accept(LibraryVisitor visitor)
- (4) visitor.visit(this)
- (5) visitor.visit(this)



软件设计

阅读下列说明和JAVA代码，将应填入空(n)处的字句写在答题纸的对应栏内。

【说明】

某快餐店主要制作并出售儿童套餐，一般包括主餐(各类比萨)、饮料和玩具，其餐品种类可能不同，但其制作过程相同。前台服务员 (Waiter) 调度厨师制作套餐。现采用生成器 (Builder) 模式实现制作过程，得到如图 6-1 所示的类图。

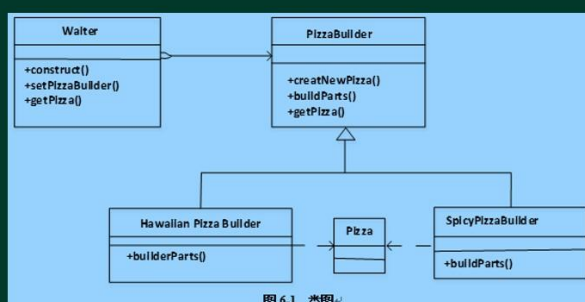


图 6-1 类图



软件设计

(希赛)

```

【Java代码】
class Pizza {
    private String parts;
    public void setParts(String parts) { this.parts = parts; }
    public String toString() { return this.parts; }
}

abstract class PizzaBuilder {
    protected Pizza pizza;
    public Pizza getPizza() { return pizza; }
    public void createNewPizza() { pizza = new Pizza(); }
    public (1) ;
}

class HawaiianPizzaBuilder extends PizzaBuilder {
    public void buildParts() { pizza.setParts("cross + mild + ham&pineapple"); }
}

class SpicyPizzaBuilder extends PizzaBuilder {
    public void buildParts() {
        pizza.setParts("pan baked + hot +pepperoni&salami");
    }
}

```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(希赛)

```

class Waiter {
    private PizzaBuilder pizzaBuilder;
    public void setPizzaBuilder(PizzaBuilder pizzaBuilder) { /*设置构建器*/
        (2) ;
    }
    public Pizza getPizza(){ return pizzaBuilder.getPizza(); }
    public void construct() { /*构建*/
        pizzaBuilder.createNewPizza();
        (3) ;
    }
}

Class FastFoodOrdering {
    public static void mainSting[]args) {
        Waiter waiter = new Waiter();
        PizzaBuilder hawaiian_pizzabuilder = new HawaiianPizzaBuilder();
        (4) ;
        (5) ;
        System.out.println("pizza: " + waiter.getPizza());
    }
}

程序的输出结果为：
Pizza:cross + mild + ham&pineapple

```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(=) 希赛

【参考答案】

- (1) `abstract void buildParts();`
- (2) `this.pizzaBuilder=pizzaBuilder`
- (3) `pizzaBuilder.buildParts()`
- (4) `waiter.setPizzaBuilder(hawaiian_pizzabuilder)`
- (5) `waiter.construct()`

www.educity.cn — 帮助客户成功，创造社会价值



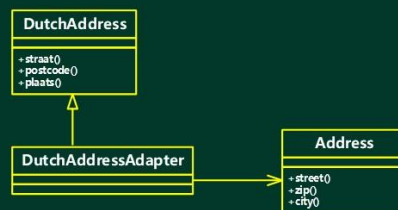
软件设计

(=) 希赛

阅读下列说明和Java代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

【说明】

某软件系统中，已设计并实现了用于显示地址信息的类Address（如图6-1所示），现要求提供基于Dutch语言的地址信息显示接口。为了实现该要求并考虑到以后可能还会出现新的语言的接口，决定采用适配器（Adapter）模式实现该要求，得到如图6-1所示的类图。



www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(希赛)

```

【Java代码】
import java.util.*;

class Address{
    public void street() { //实现代码省略 }
    public void zip() { //实现代码省略 }
    public void city() { //实现代码省略 }
    //其他成员省略
}
class DutchAddress{
    public void straat() { //实现代码省略 }
    public void postcode() { //实现代码省略 }
    public void plaats() { //实现代码省略 }
    //其他成员省略
}
class DutchAddressAdapter extends DutchAddress {
    private __ (1) __;
    public DutchAddressAdapter (Address addr){
        address= addr;
    }
    public void straat() {
        __ (2) __;
    }
}
    
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(希赛)

```

    public void postcode() {
        __ (3) __;
    }
    public void plaats(){
        __ (4) __;
    }
    //其他成员省略
}

class Test {
    public static void main(String[] args) {
        Address addr= new Address();
        __ (5) __;
        System.out.println("\n The DutchAddress\n");
        testDutch(addrAdapter);
    }

    Static void testDutch(DutchAddress addr){
        addr.straat();
        addr.postcode();
        addr.plaats();
    }
}
    
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

希赛

```
(1) Address address;  
(2) address.street();  
(3) address.zip();  
(4) address.city();  
(5) DutchAddress addrAdapter=new DutchAddressAdapter(addr);
```



软件设计

希赛

阅读下列说明和JAVA代码，将应填入空(n)处的字句写在答题纸的对应栏内。
【说明】
欲开发一个绘图软件，要求使用不同的绘图程序绘制不同的图形。以绘制直线和圆形为例，对应的绘图程序如表6-1所示。

	DP1	DP2
绘制直线	draw_a_line(x1,y1,x2,y2)	drawline(x1,x2,y1,y2)
绘制圆	draw_a_circle(x,y,r)	drawcircle(x,y,r)

表6-1 不同的绘图程序



软件设计

(希赛)

该绘图软件的扩展性要求，将不断扩充新的图形和新的绘图程序。为了避免出现类爆炸的情况，现采用桥接（Bridge）模式来实现上述要求，得到如图6-1所示的类图。

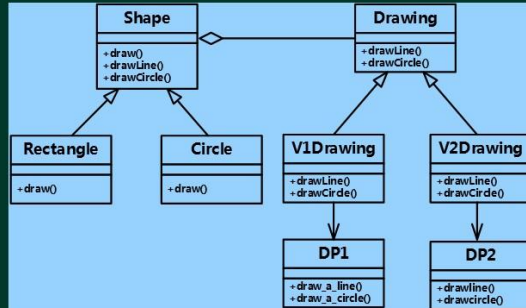


图6-1 类图

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(希赛)

【Java代码】

```

(1) Drawing{
(2) ;
(3) ;
}
class DP1{
    static public void draw_a_line(double x1, double y1, double x2, double y2) { /*代码省略*/ }
    static public void draw_a_circle(double x, double y, double r) { /*代码省略*/ }
}
class DP2{
    static public void drawline(double x1, double y1, double x2, double y2) { /*代码省略*/ }
    static public void drawcircle (double x, double y, double r) { /*代码省略*/ }
}
class V1Drawing implements Drawing{
    public void drawLine(double x1, double y1, double x2, double y2) { /*代码省略*/ }
    public void drawCircle(double x, double y, double r) {
(4) ;
    }
}
  
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(=) 希赛

```
class V2Drawing implements Drawing{
    public void drawLine(double x1, double y1, double x2 ,double y2){/*代码省略*/}
    public void drawCircle (double x, double y, double r){ (5) ;}
}
abstract class Shape{
    private Drawing _dp;
    (6) ;
    Shape(Drawing dp) {_dp=dp;}
    public void drawLine(double x1, double y1, double x2 ,double
y2){_dp.drawLine(x1,y1,x2,y2);}
    public void drawCircle (double x, double y, double r){ _dp.drawCircle(x,y,r);}
}
class Rectangle extends Shape{
    private double _x1, _x2, _y1, _y2;
    public Rectangle(Drawing dp,double x1, double y1, double x2 ,double y2) {/*代码省略*/}
    public void draw(){/*代码省略*/}
}
class Circle extends Shape{
    private double _x, _y, _r;
    public Circle(Drawing dp,double x, double y, double r) {/*代码省略*/}
    public void draw(){drawCircle(_x,_y,_r);}
}
```

www.educity.cn — 帮助客户成功，创造社会价值



软件设计

(=) 希赛

【参考答案】

- (1) public interface 或 interface
- (2) public void drawLine(double x1, double y1, double x2 ,double y2)
或 void drawLine(double x1, double y1, double x2 ,double y2)
- (3) public void drawCircle (double x, double y, double r)
或 void drawCircle (double x, double y, double r)
- (4) DP1. draw_a_circle(x,y,r)
- (5) DP2. drawcircle(x,y,r)
- (6) abstract public void draw()

www.educity.cn — 帮助客户成功，创造社会价值