例题1

5.2 试题精解

例题1（2005年11月试题30）

下列关于静态多态实现重载的描述中正确的是_____.

A.重载的方法（函数）的方法名（函数名）可以不同

B.重载的方法（函数）必须在返回值类型以及参数类型上有所不同

C.重载的方法（函数）必须在参数顺序或参数类型上有所不同

D.重载的方法（函数）只需在返回值类型上有所不同

试题分析

"overload"翻译过来就是：超载，过载，重载，超出标准负荷。先来说说重载的含义，在日常生活中我们经常要清洗一些东西，比如洗车、洗衣服。尽管我们说话的时候并没有明确地说用洗车的方式来洗车，或者用洗衣服的方式来洗一件衣服，但是谁也不会用洗衣服的方式来洗一辆车，否则等洗完时车早就散架了。我们并不要那么明确地指出来就心知肚明，这就有重载的意思了。在同一可访问区内被声名的几个具有不同参数列的（参数的类型、个数、顺序不同）同名函数，程序会根据不同的参数列来确定具体调用哪个函数，这种机制叫做重载，重载不关心函数的返回值类型。这里，"重载"的"重"的意思不同于"轻重"的"重",它是"重复"、"重叠"的意思。例如，在同一可访问区内有：

① double calculate（double）；

② double calculate（double,double）；

③ doublecalculate（double, int）；

④ doublecalculate（int, double）；

⑤ doublecalculate（int）；

⑥ floatcalculate（float）；

⑦ float calculate（double）；

6个同名函数calculate,①②③④⑤⑥中任两个均构成重载，⑥和⑦也能构成重载，而①和⑦却不能构成重载，因为①和⑦的参数相同。

试题答案

C

例题2

例题2（2005年11月试题31）

程序设计语言提供了基本类型及其相关的操作，而_____则允许开发者自定义一种新的类型及其相关的操作。

A.对象 B.实例 C.类 D.引用

试题分析

类是对某个对象的定义。它包含有关对象动作方式的信息，包括它的名称、方法、属性和事件。实际上它本身并不是对象，因为它不存在于内存中。当引用类的代码运行时，类的一个新的实例，即对象，就在内存中创建了。虽然只有一个类，但能通过这个类在内存中创建多个相同类型的对象。

可以把类看作"理论上"的对象，也就是说，它为对象提供蓝图，但在内存中并不存在。从这个蓝图可以创建任何数量的对象。从类创建的所有对象都有相同的成员：属性、方法和事件。但是，每个对象都像一个独立的实体一样动作。例如，一个对象的属性可以设置成与同类型的其他对象不同的值。

用户在设计系统的过程中，可以根据需要开发自定义的类，将属性、操作方法集中到一个类中。

在C++语言中，要在一源文件中引用一个类，只要将定义这个类的头文件包含到该文件中即可。在这里，引用可理解为使用。

试题答案

C

上一节 本书简介 下一节

第 5 章：面向对象方法学 作者：希赛教育软考学院 来源：希赛网 2014年02月08日

例题3

例题3（2005年11月试题32）

表示了对象间"is-a"的关系。

A.组合 B.引用 C.聚合 D.继承

试题分析

继承关系表示了对象间"is-a"的关系，即子类是父类的一种，比如，圆是几何图形的一种，圆是子类，几何图形是基类。

类以另一个类对象作为数据成员，称为组合。比如，车轮类、方向盘类、车架类等组合成汽车类。

在UML中，在描述类之间的关联时，用到组合（Composition）、聚合（Aggregation）的概念，它们都用于描述类的整体-部分关系。聚合只是概念性的，只是区分整体和部分；而组合则具有很强的归属性，而且整体和部分的对象生命周期是一致的，如果部分类没有了则整体类也消失了，如果整体类销毁了部分类也跟着销毁，而且在任何时候，部分类只能是它所在的整体类的组成部分

不能同时又是其他整体类的组成部分。比如，一个团队由很多队员组成，一个人可以是多个团队的队员，"队员和团队"符合聚合的概念，但不符合组合的概念。又比如，一双鞋由鞋底和鞋面组成，当鞋毁了鞋底和鞋面自然也就不存在了，当鞋底或鞋面没了鞋自然也就不能穿了，不能成为鞋了，这符合聚合、组合的概念。可以认为组合是一种要求更严格的聚合形式。

更通俗地说，在组合关联中，部分类不可以独立于整体类而存在；在聚合关联中，部分类可以独立于整体类而存在。

试题答案

D

上一节　本书简介　下一节

例题4

例题4（2005年11月试题48）

若对象A可以给对象B发送消息，那么_____.

A.对象B可以看见对象A　B.对象A可以看见对象B

C.对象A、B相互不可见　D.对象A、B相互可见

试题分析

对象A可以给对象B发送消息，那么表明对象A可以调用对象B的方法，对象A可以看见对象B.

试题答案

B

上一节　本书简介　下一节

例题5

例题5（2005年11月试题49）

类描述了一组对象共同的特性，下列叙述中正确的是_____.

A.类本身不能具有变量

B.对象具有类定义的所有变量的一份拷贝

C.对象间不能共享类定义的变量

D.可通过类名访问静态变量（类变量）

试题分析

此题考查的是面向对象的基础概念--静态变量。首先看选项A:"类本身不能具有变量",此句显然

错误，因为类是可以有自己的变量的，类自己的变量我们称之为静态变量；接着看选项B:"对象具有类定义的所有变量的一份拷贝",此句看似正确，其实是错的，对于普通变量，若有类A,A中定义了属性stuName,则在类A的实例ObjA1中也必然有属性stuName,但有一特例，即静态变量，静态变量是类的变量，它不会被拷贝到所有对象，所有对象都只有它的一个引用而已；接着看C选项："对象间不能共享类定义的变量",这句是错的，类可以定义静态变量，静态变量是属于类的变量，而不是对象的变量，也就是说，若有类A,A中定义了mystatic属性，则A生成的所有对象中mystatic属性是指向同一内存地址的，即同一变量。这也就达到了对象间共享类定义的变量的效果；最后看D选项："可通过类名访问静态变量（类变量）",此句正确，以选项C的举例来说，访问属性mystatic,可直接用A.mystatic.

试题答案

D

例题6

例题6（2005年11月试题50, 51）

（50）反映了类间的一种层次关系，而（51）反映了一种整体与部分的关系。

（50）A.继承B.组合C.封装D.多态

（51）A.继承B.组合C.封装D.多态

试题分析

此题考的是面向对象基本概念，继承反映了类间的一种层次关系，而组合反映了一种整体与部分的关系。

试题答案

（50）A　（51）B

例题8

例题8（2006年5月试题34~36）

已知3个类O、P和Q,类O中定义了一个私有方法Fl、一个公有方法F2和一个受保护的方法F3;类P和类Q是类O的派生类，其继承方式如下所示。

class P:protected O{};

class Q:public O{…};

关于方法Fl的描述中正确的是（34）;关于方法F2的描述中正确的是（35）;关于方法F3的描述中正确的是（36）.

（34）A.方法Fl无法被访问

B.只有在类O内才能访问方法Fl

C.只有在类P内才能访问方法Fl

D.只有在类Q内才能访问方法Fl

（35）A.类O、P和Q的对象都可以访问方法F2

B.类P和Q的对象都可以访问方法F2

C.类0和Q的对象都可以访问方法F2

D.只有在类P内才能访问方法F2

（36）A.类O、P和Q的对象都可以访问方法F3

B.类O、P和Q的对象都不可以访问方法F3

C.类O和Q的对象都可以访问方法F3

D.类P和Q的对象都可以访问方法F3

试题分析

本题考查的是面向对象程序设计中的继承机制。解题的关键在于理解类继承访问控制的规则。下面是对规则的总结（见表5-2）。

基类成员访问控制	继承访问控制	在派生类中的访问控制
public	public	public
protected		protected
private		不可访问
public	protected	protected
protected		protected
private		不可访问
public	private	private
protected		private
private		不可访问

表5-2 继承访问控制总结表

从表5-2中可以看出，当派生类以public方式继承基类时，派生类能继承到基类的public成员和protected成员。当派生类以protected方式继承基类时，派生类同样能继承到基类的public成员和protected成员，但是基类中的public成员在派生类中将变为protected成员。当派生类以private方式继承基类时，派生类同样能继承到基类的public成员和protected成员，但是基类中的public成员在派生类中将变为private成员。

有了这个规则，解题也就容易了，我们可以将题目要求转成表5-3和表5-4.

类 O	继承访问控制	类 P
public: F2()	protected	public:
protected: F3()		protected: F2(),F3()
private: F1()		

表5-3 类P继承类O分析表

类O	继承访问控制	类Q
public: F2()	public	public: F2()
protected: F3()		protected: F3()
private: F1()		

表5-4类Q继承类O分析表

接下来我们可以根据表格来解题。第（34）空是关于函数F1（）的描述，从表中可以看出，只有类O拥有私有成员F1（），所以只有类O可以访问F1（），所以第（34）空的答案为B.第（35）空是关于函数F2（）的描述，这里有一个细节需要注意，即第（34）空备选答案的描述是类**是否能访问F1（），而第（35）空出现了对象**是否能访问F2（）。这其中是有区别的，某个类的对象，只能访问此类的public成员。F2（）在类O和类Q中是public成员，所以类0和Q的对象都可以访问方法F2（），答案为C.第（36）空与第（35）空类似，由于F3（）在类O、类P和类Q中都是protected成员，所以不能被它们的对象访问，答案为B.

试题答案：

（34）B　（35）C　（36）B

上一节　本书简介　下一节

例题7

例题7（2005年11月试题52）

下列叙述中正确的是_____.

A.面向对象程序设计语言都不支持过程化的程序设计

B.面向对象系统只可采用面向对象程序设计语言实现

C.某些过程化程序设计语言也可实现面向对象系统

D.面向对象程序设计语言不支持对成员变量的直接访问

试题分析

某些面向对象的程序设计语言支持过程化的程序设计，过程化的程序设计语言也可实现面向对象系统。

试题答案

C

上一节　本书简介　下一节

例题9

例题9（2006年5月试题37）

在面向对象软件开发过程中，采用设计模式_____.

A.允许在非面向对象程序设计语言中使用面向对象的概念

B.以复用成功的设计和体系结构

C.以减少设计过程创建的类的个数

D.以保证程序的运行速度达到最优值

试题分析

此题考的是设计模式基本概念，要求考生清楚设计模式的优缺点。设计模式是对被用来在特写场景下解决一般设计问题的类和相互通信的对象的描述。一般而言，一个设计模式有4个基本要素：模式名称、问题（模式的使用场合）、解决方案和效果。

每一个设计模式系统地命名、解释和评价了面向对象系统中一个重要的和重复出现的设计。设计模式使人们可以更加简单、方便地复用成功的设计和体系结构；将已证实的技术变成设计模式，也会使新系统的开发者更加容易理解其设计思路。设计模式可以帮助开发者做出有利于复用的选择，避免设计时损害系统复用性。因此正确的答案为B.

试题答案

B

上一节　本书简介　下一节

例题10

例题10（2006年5月试题38）

以下关于单身模式（Singleton）的描述中，正确的是_____.

A.它描述了只有一个方法的类的集合

B.它能够保证一个类只产生一个唯一的实例

C.它描述了只有一个属性的类的集合

D.它能够保证一个类的方法只能被一个唯一的类调用

试题分析

此题属于纯概念题，单身模式的提出，其意图就是保证一个类仅有一个实例，并提供一个访问它的全局访问点。所以本题选B.

试题答案

B

上一节　本书简介　下一节

例题11

例题11（2006年5月试题39）

下列关于一个类的静态成员的描述中，不正确的是_____.

A.该类的对象共享其静态成员变量的值

B.静态成员变量可被该类的所有方法访问

C.该类的静态方法只能访问该类的静态成员变量

D.该类的静态数据成员变量的值不可修改

试题分析

类的静态成员与一般的类成员不同：静态成员与对象的实例无关，只与类本身有关。它们用来实现类要封装的功能和数据，但不包括特定对象的功能和数据。静态成员包括静态方法和静态属性。

静态属性包含在类中要封装的数据，可以由所有类的实例共享。实际上，除了属于一个固定的类并限制访问方式外，类的静态属性非常类似于函数的全局变量。

题目选项D所述的"该类的静态数据成员变量的值不可修改"这种说法是错误的，静态数据成员变量的值是可以修改的。

试题答案

D

上一节 本书简介 下一节

例题12

例题12（2006年5月试题40）

在MVC（模型、视图、控制器）模式中，视图部分描述的是_____.

A.将应用问题域中包含的抽象领域知识呈现给用户的方式

B.应用问题域中所包含的抽象类

C.用户界面对用户输入的响应方式

D.应用问题域中包含的抽象领域知识

试题分析

在解题之前，先来看看MVC.

模型-视图-控制器（Model-View-Controller,MVC）模式是一种拆分方法，它将应用程序拆分成三个部分：模型、视图和控制器。其中模型表示企业数据管理对该数据的访问和更新的业务规则。通常，模型充当现实世界中的过程的软件模拟，这样，在定义模型的时候即可以应用真实世界的建模技术。视图处理模型的内容，它通过模型访问企业数据，并指定应该如何表示该数据。在模

型发生改变时，视图将负责在它的表示中保持一致性，这可以通过使用推（Push）模型（视图向该模型注册，以获取它的改变通知）来实现，也可以使用拉（Pull）模型（此时视图负责在需要检索最新数据时调用模型）来实现。控制器将和视图之间的交互转换为由模型执行的操作。在独立的GUI客户机中，用户交互可能是按钮单击或菜单选择；然而在Web应用程序中，它们则可能是GET和POST HTTP请求。由模型执行的操作包括激活业务流程或改变模型状态。控制器根据用户交互和模型操作的结果选择合适的视图，从而做出响应。采用MVC体系结构有以下优势。

（1）多个视图使用同一个模型。模型和视图的分开使多个视图可以使用相同的企业模型。因此，企业应用程序的模型组件就更容易实现、测试和维护，因为所有对模型的访问都要经过这些组件。

（2）对客户机新类型更容易支持。要支持客户机的新类型，只需为其编写一个视图和控制器，然后在已有的企业模型中将它们进行连接即可。

模型、视图和控制器之间的关系见图5-1.

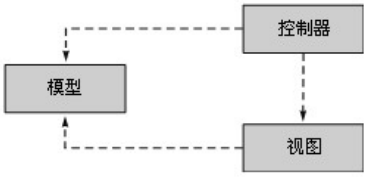


图5-1 模型、视图和控制器之间的关系示意图

视图和控制器都依赖于模型，但是，模型既不依赖于视图，也不依赖于控制器。这是分离的主要优点之一。这样的分离允许模型在独立于可视表示功能的情况下建立和测试。

模型：模型用于管理应用程序域的行为和数据，并响应为获取其状态信息（通常来自视图）而发出的请求，还会响应更改状态的指令（通常来自控制器）。

视图：视图用于管理信息的显示。

控制器：控制器用于解释用户的鼠标和键盘输入，以通知模型和/或视图进行相应的更改。

通过对概念的学习，可以得知此题正确答案为：A.

试题答案

A

上一节　本书简介　下一节

例题13

例题13（2006年5月试题41）

对于如图5-2所示的UML类图，正确的描述是_____.

图5-2 UML类图

A.类B的实例中包含了对类C的实例的引用

B.类A的实例中包含了对类B的实例的引用

C.类A的实例中包含了对类C的实例的引用

D.类B的实例中包含了对类A的实例的引用

试题分析

本题考查的是UML中的关联关系。

在UML中，关联（Association）是一种结构关系，它指明一个事物的对象与另一个事物的对象之间的联系。给定一个连接两个类的关联，可以从一个类的对象导航到另一个类的对象，反之亦然。在图形上，把关联画成一条连接相同类或不同类的实线。

通过一个指定走向的单向箭头修饰关联，可以显示地描述导航的方向。在题目中所示的UML类图中，导航方向为A->B,说明可以从类A的实例导航到类B的实例，因此在类A中必然包含一个对类B的实例引用。图中"C"表示的是关联一端的角色名称。

试题答案

B

上一节　本书简介　下一节

例题14

例题14（2006年5月试题42）

UML中关联的多重度是指_____。

A.一个类有多少个方法被另一个类调用

B.一个类的实例能够与另一个类的多少个实例相关联

C.一个类的某个方法被另一个类调用的次数

D.两个类所具有的相同的方法和属性

试题分析

此题属于概念题。在UML中，重复度（Multiplicity）即多重度，定义了某个类的一个实例可以与另一个类的多少个实例相关联。通常把它写成一个表示取值范围的表达式或者一个具体的值。所以此题应选B.

试题答案

B

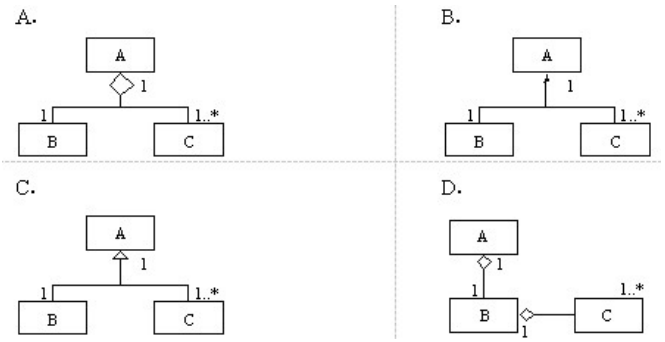
上一节　本书简介　下一节

例题15

例题15（2006年5月试题43）

已知3个类A、B和C,其中类A由类B的一个实例和类C的1个或多个实例构成。能够正确表示类A、B和C之间关系的UML类图是　　.

供选择的答案



试题分析

此题涉及UML图中类之间的关系图示法，所以我们得先弄清楚备选答案中出现的几个符号所代表的含义。

依赖关系：

有两个元素X、Y,如果修改元素X的定义可能会引起对另一个元素Y的定义的修改，则称元素Y依赖（Dependency）于元素X.在UML中，使用带箭头的虚线表示依赖关系，如图5-3所示。

图5-3 依赖关系的图示

在类中，依赖由各种原因引起，例如，一个类向另一个类发消息；一个类是另一个类的数据成员；一个类是另一个类的某个操作参数。如果一个类的接口改变，它发出的任何消息可能不再合法。

泛化关系：

泛化关系描述了一般事物与该事物中的特殊种类之间的关系，也就是父类与子类之间的关系。继承关系是泛化关系的反关系，也就是说，子类是从父类继承的，而父类则是子类的泛化。在UML中，使用带空心箭头的实线表示泛化关系，箭头指向父类，如图5-4所示。

图5-4 泛化关系的图示

在UML中，对泛化关系有3个要求。

子类应与父类完全一致，父类所具有的关联、属性和操作，子类都应具有；

子类中除了有与父类一致的信息外，还包括额外的信息；

可以使用父类实例的地方，也可以使用子类实例。

聚合关系：

聚合（Aggregation）是一种特殊形式的关联，它是传递和反对称的。聚合表示类之间的关系是整体与部分的关系。例如，一辆轿车包含四个车轮、一个方向盘、一个发动机和一个底盘，就是聚合的一个例子。在UML中，使用一个带空心菱形的实线表示聚合关系，空心菱形指向的是代表"整体"的类，如图5-5所示。

图5-5 聚合关系的图示

组合关系：

如果聚合关系中表示"部分"的类的存在与否，与表示"整体"的类有着紧密的关系，例如"公司"与"部门"之间的关系，那么就应该使用"组合"关系来表示这种关系。在UML中，使用带有实心菱形的实线表示组合关系（见图5-6）。

图5-6 组合关系的图示

由于题目指出"类A由类B的一个实例和类C的1个或多个实例构成",所以类A与类B、类C的关系应为聚合或者组合关系，其示意图如图5-7所示。

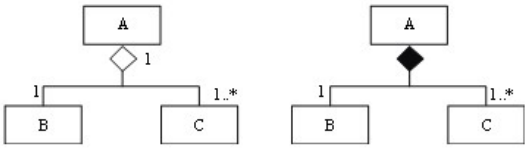


图5-7 聚合、组合示意图

所以答案为：A.

试题答案

A

上一节　本书简介　下一节

例题16

例题16（2006年11月试题33）

面向对象分析的第一步是_____.

A.定义服务　B.确定附加的系统约束

C.确定问题域　D.定义类和对象

试题分析

面向对象分析的步骤如下。

第一步，确定问题域。

第二步，确定对象和类。这里所说的对象是对数据及其处理方式的抽象，它反映了系统保存和处理现实世界中某些事物的信息的能力。类是多个对象的共同属性和方法集合的描述，它包括如何在一个类中建立一个新对象的描述。

第三步，确定结构（structure）。结构是指问题域的复杂性和连接关系。类成员结构反映了泛化-特化关系，整体-部分结构反映整体和局部之间的关系。

第四步，确定主题（subject）。主题是指事物的总体概貌和总体分析模型。

第五步，确定属性（attribute）。属性就是数据元素，可用来描述对象或分类结构的实例，可在图中给出，并在对象的存储中指定。

第六步，确定方法（method）。方法是在收到消息后必须进行的一些处理方法，方法要在图中定义，并在对象的存储中指定。对于每个对象和结构来说，那些用来增加、修改、删除和选择一个方法本身都是隐含的，而有些则是显示的。

所以此题的答案为：C.

试题答案

C

例题17

例题17（2006年11月试题34）

面向对象程序设计语言为_____提供支持。

A.面向对象用例设计阶段　B.面向对象分析阶段

C.面向对象需求分析阶段　D.面向对象实现阶段

试题分析

此题非常容易，程序设计语言是实施阶段用到的工具，故是为面向对象实现阶段提供技术。

试题答案

D

例题18

例题18（2006年11月试题35）

下面关于面向对象的描述正确的是_____.

A.针对接口编程，而不是针对实现编程

B.针对实现编程，而不是针对接口编程

C.接口与实现不可分割

D.优先使用继承而非组合

试题分析

"针对接口编程，而不是针对实现编程",这是面向对象程序设计的一条重要原则，遵循此原则有以下好处。

使用者不必知道其使用对象的具体所属类；

使用者无需知道特定类，只需知道他们所期望的接口；

一个对象可以很容易地被（实现了相同接口的）另一个对象所替换；

对象间的连接不必硬绑定到一个具体类的对象上，因此增加了灵活性。

试题答案

A

例题19

例题19（2006年11月试题37~39）

UML的设计视图包含了类、接口和协作，其中，设计视图的静态方面由（37）和（38）表现；动态方面由交互图、（39）表现。

（37）A.类图　B.状态图　C.活动图　D.用例图

（38）A.状态图　B.顺序图　C.对象图　D.活动图

（39）A.状态图和类图　B.类图和活动图

C.对象图和状态图　D.状态图和活动图

试题分析

UML的9种图可分为表示系统静态结构的静态模型（包括对象图、类图、构件图、部署图），以及表示系统动态结构的动态模型（包括顺序图、协作图、状态图、活动图），其中顺序图和协作图都属于交互图。此外，用例图的分类目前尚有争议，有书上将其分类为动态模型，也有书上将其分类为静态模型，软设的官方标准教程上将其分类为静态模型，而此题中明显将其分类为动态模型，所以为了避免出错，建议大家在解题时先不要考虑用例图，以其他选项来推答案。

在第（37）空的备选答案中，类图是明显的静态图，而（38）中对象图是明显的静态图，（39）中将静态图：类图和对象图排除，可以得到状态图和活动图为动态图。由此可见，本题答案为：A,C,D.

试题答案：

（37）A　（38）C　（39）D

例题20

例题20（2006年11月试题41）

设计模式具有_____的优点。

A.适应需求变化　B.程序易于理解

C.减少开发过程中的代码开发工作量　D.简化软件系统的设计

试题分析

设计模式是用一种固定的解决方案来解决某一类问题，这种方式第一大优点是方案出错的可能性很小，因为这些方案都是经过很多人实践总结出来的；第二是适应需求变化，如例32中所描述的职责链模式，当系统业务流程有变化时，只需要很少的调整即可达到目的。

试题答案

A

　作者：希赛教育软考学院　来源：希赛网　2014年02月08日

例题21

例题21（2006年11月试题42）

下面的_____模式将对象组合成树形结构以表示"部分-整体"的层次结构，并使得用户对单个对象和组合对象的使用具有一致性。

A.组合（Composite）B.桥接（Bridge）

C.修饰（Decorator）D.外观（Facade）

试题分析

下面是常用的23种模式简介。

（1）Abstract Factory模式：提供一个创建一系列相关或相互依赖对象的接口，而无须指定它们具体的类。

（2）Adapter模式：将一个类的接口转换成客户希望的另一个接口。Adapter模式使得原来由于接口不兼容而不能一起工作的那些类可以一起工作。

（3）Bridge模式：将抽象部分与它的实现部分分离，使它们都可以独立地变化。

（4）Builder模式：将一个复杂对象的构建与它的表现分离，使得同样的构建过程可以创建不同的表示。

（5）Chain of Responsibility模式：为解除请求的发送者与接收者之间耦合，而使多个对象都有机会处理这个请求。将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它。

（6）Command模式：将一个请求封装为一个对象，从而使你可用不同请求对客户进行参数化、对请求排队或记录请求日志，以及支持可取消的操作。

（7）Composite模式：将对象组合成树形结构以表示"部分-整体"的层次结构。Composite使得客户对单个对象和复合对象的使用具有一致性。

（8）Decorator模式：动态地给一个对象添加一个额外的职责。就扩展功能而言，Decorator模式比生成子类方式更为灵活。

（9）Facade模式：为子系统中的一组接口提供一个一致的界面，Facade模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。

（10）Factory Method模式：定义一个用于创建对象的接口，让子类决定将哪个类实例化。

Factory Method使一个类的实例化延迟到其子类。

（11）Flyweight模式：运用共享技术有效地支持大量细粒度的对象。

（12）Interpreter模式：给定一个语言，定义它的文法的一种表示，并定义一个解释器，该解释器使用该表示来解释语言中的句子。

（13）Iterator模式：提供一种方法顺序访问一个聚合对象中各个元素，而又不需要暴露该对象的内部表示。

（14）Mediator模式：用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显示地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。

（15）Memento模式：在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。这样以后就可将该对象恢复到保存状态。

（16）Observer模式：定义对象间的一种一对多的依赖关系，以便当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动刷新。

（17）Prototype模式：用原型实例指定创建对象的种类，并且通过拷贝这个原型来创建新的对象。

（18）Proxy模式：为其他对象提供一个代理，以控制对这个对象的访问。

（19）Singleton模式：保证一个类仅有一个实例，并提供一个访问它的全局访问点。

（20）State模式：允许一个对象在其内部状态改变时改变它的行为。对象看起来似乎修改了它所属的类。

（21）Strategy模式：定义一系列的算法，把它们一个个封装起来，并且使它们可相互替换。本模式使得算法的变化可独立于使用它的客户。

（22）Template Method模式：定义一个操作中的算法的骨架，而将一些步骤延迟到子类中。Template Method使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。

（23）Visitor模式：表示一个作用于某对象结构中的各元素的操作。它使你可以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。

从中可以看出题目的正确答案应为：A.

试题答案

A

上一节　本书简介　下一节

例题22

例题22（2006年11月试题43）

下图描述了一种设计模式，该设计模式不可以_____.

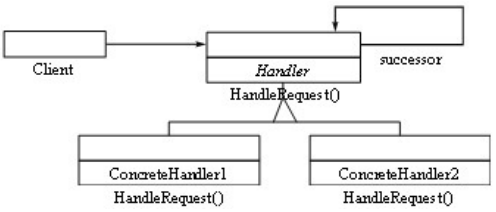


图5-8 类图

A.动态决定由一组对象中某个对象处理该请求

B.动态指定处理一个请求的对象集合，并高效率地处理一个请求

C.使多个对象都有机会处理请求，避免请求的发送者和接收者间的耦合关系

D.将对象连成一条链，并沿着该链传递请求

试题分析

本题考的是设计模式中职责链设计模式的基本概念。图中的各元素含义如下。

Handler

--定义一个处理请求的接口。

--（可选）实现后继链。

ConcreteHandler

--具体处理者，处理它所负责的请求。

--可访问它的后继者。

--如果可处理该请求，就处理之，否则将请求转发给它的后继者。

Client

--向链上的具体处理者对象提交请求。

职责链设计模式的意图就是使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系。将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它为止。这样有多个对象可以处理一个请求，具体是哪个对象处理请求要在运行时确定。所以答案A、C、D都是正确的。B选项中提到的"高效率地处理一个请求"这种讲法是不对的，职责链的处理流程有一个寻找处理对象的过程，这肯定不及指定处理对象的效率高。另外，职责链的优势在于其适应性强，能应对流程的变化，而不是效率高。

试题答案

B

上一节　本书简介　下一节

例题23

例题23（2006年11月试题44）

在面向对象程序设计中，常常将接口的定义与接口的实现相分离，可定义不同的类实现相同的接口。在程序运行过程中，对该接口的调用可根据实际的对象类型调用其相应的实现。为达到上述目的，面向对象语言需提供_____机制。

A.继承和过载（overloading） B.抽象类

C.继承和重置（overriding） D.对象自身引用

试题分析

"overload"翻译过来就是：超载、过载、重载、超出标准负荷；"override"翻译过来就是：重置、覆盖，使原来的失去效果。

在同一可访问区内（如同一个类内）被声明的几个具有不同参数列的（参数的类型、个数、顺序不同）同名函数，程序会根据不同的参数列来确定具体调用哪个函数，这种机制叫做重载，重载不关心函数的返回值类型。这里，"重载"的"重"的意思不同于"轻重"的"重",它是"重复"、"重叠"的意思。

覆盖是指派生类中存在重新定义的函数，其函数名、参数列、返回值类型必须同父类中的相对应被覆盖的函数严格一致，覆盖函数和被覆盖函数只有函数体（花括号中的部分）不同，当派生类对象调用子类中该同名函数时会自动调用子类中的覆盖版本，而不是父类中的被覆盖版本，这种机制叫做覆盖。

覆盖的本质就是面向对象，即根据实际的对象类型来调用其相应的实现函数以实现多态。所以答案为C.

试题答案

C

上一节 本书简介 下一节

作者：希赛教育软考学院 来源：希赛网 2014年02月08日

例题24

例题24（2007年5月试题21）

在面向对象的语言中，（21）.

A. 类的实例化是指对类的实例分配存储空间

B. 每个类都必须创建一个实例

C. 每个类只能创建一个实例

D. 类的实例化是指对类进行初始化

试题分析

类是对某个对象的定义。它包含有关对象动作方式的信息，包括它的名称、方法、属性和事件。实际上它本身并不是对象，因为它不存在于内存中。当引用类的代码运行时，类的一个新的实例，即对象，就在内存中创建了。虽然只有一个类，但能从这个类在内存中创建多个相同类型的对象。

可以把类看作"理论上"的对象，也就是说，它为对象提供蓝图，但在内存中并不存在。从这个蓝图可以创建任何数量的对象。从类创建的所有对象都有相同的成员：属性、方法和事件。但是，每个对象都象一个独立的实体一样动作。例如，一个对象的属性可以设置成与同类型的其他对象不同的值。

不一定每个类都必须创建一个实例，如接口类，是不能进行实例化的。

试题答案

A

例题25

例题25（2007年5月试题22）

在统一建模语言（UML）中，（22）用于描述系统与外部系统及用户之间的交互。

A. 类图

B. 用例图

C. 对象图

D. 协作图

试题分析

本题考查UML中各种图的功能。UML中需要掌握的有9种图：

类图（class diagram）：展现了一组对象、接口、协作和它们之间的关系。在面向对象系统的建模中所建立的最常见的图就是类图。类图给出系统的静态设计视图。包含主动类的类图给出了系统的静态进程视图。

对象图（object diagram）：展现了一组对象以及它们之间的关系。对象图描述了在类图中所建立的事物实例的静态快照。和类图相同，这些图给出系统的静态设计视图或静态进程视图，但它们是从真实的或原型案例的角度建立的。

用例图（use case diagram）：展现了一组用例、参与者（actor）以及它们之间的关系。用例图给出系统的静态用例视图。这些图对系统的行为进行组织和建模是非常重要的。

序列图（sequence diagram）：是场景（scenano）的图形化表示，描述了以时间顺序组织的对象之间的交互活动。

协作图（collaboration diagram）：强调收发消息的对象的结构组织。序列图和协作图都是交互图（interaction diagram）。交互图展现了一种交互，它由一组对象和它们之间的关系组成，包括它们之间可能发送的消息。交互图关注系统的动态视图。序列图和协作图是同构的，它们之间可以相互转换。

状态图（statechart diagram）：展现了一个状态机，它由状态、转换、事件和活度组成。状态图关注系统的动态视图，它对于接口、类和协作的行为建模尤为重要，它强调对象行为的事件顺序。

活动图（activity diagram）：是一种特殊的状态图，它展现了在系统内从一个活动到另一个活动的流程。活动图专注于系统的动态视图。它对于系统的功能建模特别重要，并强调对象间的控制流程。

构件图（component diagram）：展现了一组构件之间的组织和依赖。构件图专注于系统的静

态实现视图。它与类图相关，通常把构件映射为一个或多个类、接口或协作。

部署图（deployment diagram）：展现了运行处理节点以及其中的构件的配置。部署图给出了体系结构的静态实施视图。它与构件图相关，通常一个节点包含一个或多个构件。

试题答案

B

上一节　本书简介　下一节

例题26

例题26（2007年5月试题36）

面向对象分析与设计是面向对象软件开发过程中的两个重要阶段，下列活动中，（36）_____不属于面向对象分析阶段。

A. 构建分析模型

B. 识别分析类

C. 确定接口规格

D. 评估分析模型

试题分析

面向对象分析的目的是为了获得对应用问题的理解。理解的目的是确定系统的功能、性能要求。面向对象分析阶段包含5个活动：认定对象、组织对象、描述对象间的相互作用、定义对象的操作、定义对象的内部信息。分析阶段最重要的是理解问题域的概念，其结果将影响整个工作。A、B、D答案都属于对象分析阶段，而C答案确定接口规格是面向对象设计。

试题答案

C

上一节　本书简介　下一节

例题27

例题27（2007年5月试题38）

面向对象分析需要找出软件需求中客观存在的所有实体对象（概念），然后归纳、 抽象出实体类。_____ 是寻找实体对象的有效方法之一。

供选择的答案：

A. 会议调查

B. 问卷调查

C. 电话调查

D. 名词分析

试题分析

面向对象分析的过程包括：

1、从用例中提取实体对象和实体类。

提取实体对象的方法，依据用例描述中出现的名词和名词短语来提取实体对象，必须对原始的名词和名词短语进行筛选。得到实体对象后，对实体对象进行归纳、抽象出实体类。

2、提取属性

3、提取关系

4、添加边界类

5、添加控制类

6、绘制类图

7、绘制顺序图

8、编制术语表

所以"名词分析"是寻找实体对象的有效方法之一。

试题答案：

D

上一节　本书简介　下一节

例题28

例题28（2007年5月试题39-40）

在"模型-视图-控制器"（MVC）模式中，（39）主要表现用户界面，（40）用来描述核心业务逻辑。

（39）A. 视图

B. 模型

C. 控制器

D. 视图和控制器

（40）A. 视图

B. 模型

C. 控制器

D. 视图和控制器

试题分析

MVC模式是一个复杂的架构模式，其实现也显得非常复杂。

视图（View）代表用户交互界面，对于Web应用来说，可以概括为HTML界面，但有可能为

XHTML、XML和Applet.

模型（Model）：就是业务流程/状态的处理以及业务规则的制定。业务模型的设计可以说是MVC最主要的核心。

控制（Controller）可以理解为从用户接收请求， 将模型与视图匹配在一起，共同完成用户的请求。

试题答案

（39）A （40）B

上一节 本书简介 下一节

例题29

例题29（2007年5月试题41）

在进行面向对象设计时，采用设计模式能够_____.

A. 复用相似问题的相同解决方案

B. 改善代码的平台可移植性

C. 改善代码的可理解性

D. 增强软件的易安装性

试题分析

因为模式是一种指导，在一个良好的指导下，有助于你完成任务，有助于你作出一个优良的设计方案，达到事半功倍的效果，而且会得到解决问题的最佳办法。采用设计模式能够复用相似问题的相同解决方案，加快设计的速度，提高了一致性。

试题答案

A

上一节 本书简介 下一节

例题30

例题30（2007年5月试题42-44）

下面给出了四种设计模式的作用：

外观（Facade）：为子系统中的一组功能调用提供一个一致的接口，这个接口使得这一子系统更加容易使用；

装饰（Decorate）： 当不能采用生成子类的方法进行扩充时，动态地给一个对象添加一些额

外的功能；

单件（Singleton）：保证一个类仅有一个实例，并提供一个访问它的全局访问点；

模板方法（Template Method）：在方法中定义算法的框架，而将算法中的一些操作步骤延迟到子类中实现。

请根据下面叙述的场景选用适当的设计模式。若某面向对象系统中的某些类有且只有一个实例，那么采用（42）设计模式能够有效达到该目的；该系统中的某子模块需要为其它模块提供访问不同数据库系统（Oracle、SQL Server、DB2 UDB 等）的功能，这些数据库系统提供的访问接口有一定的差异，但访问过程却都是相同的，例如，先连接数据库，再打开数据库，最后对数据进行查询，（43）设计模式可抽象出相同的数据库访问过程；系统中的文本显示类（TextView）和图片显示类（PictureView）都继承了组件类（Component），分别显示文本和图片内容，现需要构造带有滚动条、或者带有黑色边框、或者既有滚动条又有黑色边框的文本显示控件和图片显示控件，但希望最多只增加三个类，（44）设计模式可以实现该目的。

（42）A. 外观 B. 装饰 C. 单件 D. 模板方法

（43）A. 外观 B. 装饰 C. 单件 D. 模板方法

（44）A. 外观 B. 装饰 C. 单件 D. 模板方法

试题分析

单件模式保证一个类仅有一个实例，并提供一个访问它的全局访问点。第（42）空的题目中讲"某些类有且仅有一个实例",所以选择C答案。模板方法指在方法中定义算法的框架，而将算法中的一些操作步骤延迟到子类中实现。第（43）空的数据库连接、打开、查询这是所有数据库系统操作的步骤，只是访问方式有所不同，在算法框架里没必要事先摆出来，可以放到各个子类中讨论，所以选择D答案。装饰是当不能采用生成子类的方法进行扩充时，动态地给一个对象添加一些额外的功能。第（44）问是"构造带有……"动态的给一些对象添加额外的功能。所以选择B答案。

试题答案

（42）C （43）D （44） B

上一节 本书简介 下一节

第 5 章：面向对象方法学 作者：希赛教育软考学院 来源：希赛网 2014年02月08日

例题31

例题31（2007年5月试题45）

在采用标准 UML 构建的用例模型（Use-Case Model）中，参与者（Actor）与用例（Use Case）是模型中的主要元素，其中参与者与用例之间可以具有_____关系。

A. 包含（include）

B. 递归（Recursive）

C. 关联（Association）

D. 组合（Composite）

试题分析

参与者用于表示使用系统的对象，可以是一个物体或另一个系统。用例是用户期望系统具备的动作。参与者可以与多个用例相关，而用例也可以与多个参与者相关。所以参与者与用例之间可以具有关联关系。

试题答案

C

例题32

例题32（2007年5月试题46-47）

当采用标准 UML 构建系统类模型（Class Model）时，若类 B 除具有类 A 的全部特性外，类 B 还可定义新的特性以及置换类 A 的部分特性，那么类 B 与类 A 具有 （46） 关系； 若类 A 的对象维持类 B 对象的引用或指针，并可与类 C 的对象共享相同的类 B 的对象， 那么类 A 与类 B 具有（47） 关系。

（46）A. 聚合 B. 泛化 C. 传递 D. 迭代

（47）A. 聚合 B. 泛化 C. 传递 D. 迭代

试题分析

泛化表示类与类之间的继承关系，接口与接口之间的继承关系，或类对接口的实现关系。一般泛化的关系是从子类指向父类的，与继承或实现的方法相反。

聚合当对象A被加入到对象B中，成为对象B的组成部分时，对象B和对象A之间为聚集关系。聚合是关联关系的一种，是较强的关联关系，强调的是整体与部分之间的关系。

关联对于两个相对独立的对象，当一个对象的实例与另一个对象的一些特定实例存在固定的对应关系时，这两个对象之间为关联关系。

依赖对于两个相对独立的对象，当一个对象负责构造另一个对象的实例，或者依赖另一个对象的服务时，这两个对象之间主要体现为依赖关系。

迭代当对象A维持B对象的引用或指针，并与对象C共享相同的对象B,则A与B具有聚合关系。

试题答案

B　A

例题33

例题33（2007年11月试题19）

采用UML进行软件建模过程中，类图是系统的一种静态视图，用　　可明确表示两类事物之间存在的整体/部分形式的关联关系。

A. 依赖关系B. 聚合关系C. 泛化关系D. 实现关系

试题分析

请参看例题15分析。

试题答案

B

上一节　本书简介　下一节

例题34

例题34（2007年11月试题37）

_____表示了系统与参与者之间的接口。在每一个用例中，该对象从参与者处收集信息，并将之转换为一种被实体对象和控制对象使用的形式。

A. 边界对象　B. 可视化对象　C. 抽象对象　D. 实体对象

试题分析

边界类描述的是系统外部环境和系统内部运作之间的交互，他工作在actor和系统之间；则边界对象表示的为一个交互接口。

实体类的主要职责是存储和管理系统内部的信息，它也可以有行为，甚至很复杂的行为，但这些行为必须与它所代表的实体对象密切相关；实体类独立于系统外部环境（Actor）。

控制类描述的是特定UseCase的控制行为，与特定的UseCase实现密切相关，可以说他就是在执行UseCase实现；控制类可以有效地降低边界类和实体类之间的耦合，使系统对于外部环境的变化具有良好的适应性。

试题答案

A

上一节　本书简介　下一节

例题35

例题35（2007年11月试题38）

在UML语言中，下图中的a、b、c三种图形符号按照顺序分别表示_____.

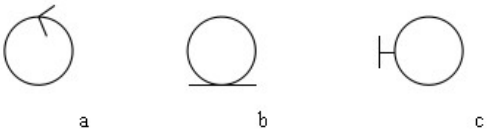


图5-9 图形符号

A. 边界对象、实体对象、控制对象B. 实体对象、边界对象、控制对象

C. 控制对象、实体对象、边界对象D. 边界对象、控制对象、实体对象

试题分析

本题考查的是采用UML进行面向对象设计过程中，各种不同对象的图形表示，a、b、c三种图形符号按照顺序分别表示控制对象、实体对象、边界对象。

试题答案

C

上一节　本书简介　下一节

第 5 章：面向对象方法学　作者：希赛教育软考学院　来源：希赛网　2014年02月08日

例题36

例题36（2007年11月试题39-42）

在下面的用例图（UseCase Diagram）中，X1、X2和X3表示（39），已知UC3是抽象用例，那么X1可通过（40）用例与系统进行交互。并且，用例（41）是UC4的可选部分，用例（42）是UC4的必须部分。

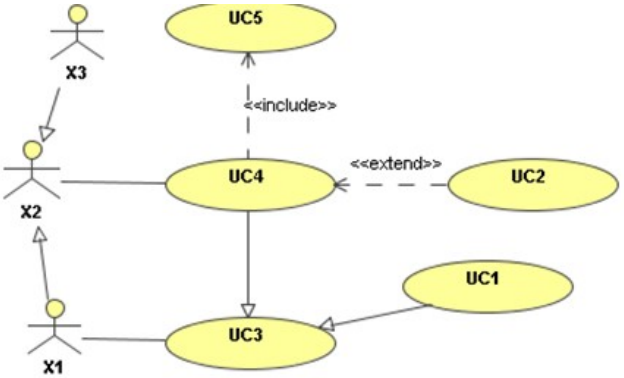


图5-10 用例图

（39）A. 人B. 系统 C. 参与者D. 外部软件

（40）A. UC4、UC1B. UC5、UC1C. UC5、UC2D. UC1、UC2

（41）A. UC1B. UC2C. UC3D. UC5

（42）A. UC1B. UC2 C. UC3D. UC5

试题分析

用例图由参与者（Actor）、用例（Use Case）、系统边界、箭头组成，用画图的方法来完成。用例描述用来详细描述用例图中每个用例，用文本文档来完成。用例建模的最主要功能就是用来表达系统的功能性需求或行为。此题当中的x1、x2、x3表示参与者（人）。一个椭圆表示用例，一个小人表示参与者。

在用例图中，用例间的关系有：

1、泛化关系（Generalization）

用例间的泛化关系和类间的泛化关系类似，即在用例泛化中，子用例表示父用例的特殊形式。子用例从父用例处继承了行为和属性， 还可以添加行为和属性，改变已继承的行为。

2、包含关系（Include）

包含关系把几个用例的公共步骤分离成一个单独的被包含用例。用例间的包含关系允许包含提供者用例的行为到用户用例的事件中， 把包含用例称为客户用例，被包含用例称为提供者用例，包含用例提供功能给客户用例。

3、扩展关系（Extend）

扩展关系是把新行为插入到已有的用例的方法。基础用例提供了一组扩展点（Extension points），在这些扩展点中可以添加新的行为 ,而扩展用例提供了一组插入片段，这些片段能够被插入到基础用例的扩展点。

试题答案

（39）C （40）A （41）B （42）D

 作者：希赛教育软考学院 来源：希赛网 2014年02月08日

例题37

例题37（2007年11月试题43）

_____设计模式定义了对象间的一种一对多的依赖关系，以便当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动刷新。

A. Adapter（适配器）B. Iterator（迭代器）

C. Prototype（原型）D. Observer（观察者）

试题分析

设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

1、Observer（观察者）设计模式，定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时， 所有依赖于它的对象都得到通知并被自动更新。

适用性：当一个抽象模型有两个方面， 其中一个方面依赖于另一方面。将这二者封装在独立的对象中以使它们可以各自独立地改变和复用。（1）当对一个对象的改变需要同时改变其它对象， 而不知道具体有多少对象有待改变。（2）当一个对象必须通知其它对象，而它又不能假定其它对象是谁。换言之， 你不希望这些对象是紧密耦合的。

2、Adapter（适配器）设计模式的意图是将一个类的接口转换成客户希望的另外一个接口。

Adapter模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

适用性： 你想使用一个已经存在的类，而它的接口不符合你的需求。

你想创建一个可以复用的类，该类可以与其他不相关的类或不可预见的类（即那些接口可能不

一定兼容的类）协同工作。（仅适用于对象Adapter）你想使用一些已经存在的子类，但是不可能对每一个都进行子类化以匹配它们的接口。对象适配器可以适配它的父类接口。

3、Prototype（原型）设计模式的意图是用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。

适用性： 当要实例化的类是在运行时刻指定时，例如，通过动态装载；或者为了避免创建一个与产品类层次平行的工厂类层次时；或者当一个类的实例只能有几个不同状态组合中的一种时。建立相应数目的原型并克隆它们可能比每次用合适的状态手工实例化该类更方便一些。

4、Iterator（迭代器）设计模式的意图是提供一种方法顺序访问一个聚合对象中各个元素， 而又不需暴露该对象的内部表示。

适用性： 访问一个聚合对象的内容而无需暴露它的内部表示。

（1）支持对聚合对象的多种遍历。

（2）为遍历不同的聚合结构提供一个统一的接口（即， 支持多态迭代）。

试题答案

D

上一节　本书简介　下一节

例题38

例题38（2007年11月试题44-47）

UML中有多种类型的图，其中，（44）对系统的使用方式进行分类，（45）显示了类及其相互关系，（46）显示人或对象的活动，其方式类似于流程图，通信图显示在某种情况下对象之间发送的消息，（47）与通信图类似，但强调的是顺序而不是连接。

（44）A. 用例图　B. 顺序图 C. 类图 D. 活动图

（45）A. 用例图　B. 顺序图 C. 类图　D. 活动图

（46）A. 用例图　B. 顺序图 C. 类图　D. 活动图

（47）A. 用例图　B. 顺序图　C. 类图　D. 活动图

试题分析

用例图：描述了系统提供的一个功能单元。用例图的主要目的是帮助开发团队以一种可视化的方式理解系统的功能需求，包括基于基本流程的"角色"（actors,也就是与系统交互的其他实体）关系，以及系统内用例之间的关系。用例图一般表示出用例的组织关系--要么是整个系统的全部用例，要么是完成具有功能（例如，所有安全管理相关的用例）的一组用例。

类图：表示不同的实体（人、事物和数据）如何彼此相关；换句话说，它显示了系统的静态结构。类图可用于表示逻辑类，逻辑类通常就是业务人员所谈及的事物种类。类图还可用于表示实现类，实现类就是程序员处理的实体。实现类图或许会与逻辑类图显示一些相同的类。

序列图：显示具体用例（或者是用例的一部分）的详细流程。它几乎是自描述的，并且显示了流程中中不同对象之间的调用关系，同时还可以很详细地显示对不同对象的不同调用。序列图有两

个维度：垂直维度以发生的时间顺序显示消息/调用的序列；水平维度显示消息被发送到的对象实例。

活动图：表示在处理某个活动时，两个或者更多类对象之间的过程控制流。活动图可用于在业务单元的级别上对更高级别的业务过程进行建模，或者对低级别的内部类操作进行建模。

试题答案

（44）A （45）C （46）D （47）B

上一节 本书简介 下一节

例题39

例题39（2008年5月试题16）

采用 UML 进行软件设计时，可用（16）关系表示两类事物之间存在的特殊/一般关系，用聚集关系表示事物之间存在的整体/部分关系。

A. 依赖 B. 聚集 C. 泛化 D. 实现

试题分析

请参看例题15分析。

试题答案

C

上一节 本书简介 下一节

例题40

例题40（2008年5月试题37-38）

（37）是指把数据以及操作数据的相关方法组合在同一个单元中，使我们可以把类作为软件中的基本复用单元，提高其内聚度，降低其耦合度。面向对象中的（38）机制是对现实世界中遗传现象的模拟，通过该机制，基类的属性和方法被遗传给派生类。

（37）A. 封装 B. 多态 C. 继承 D. 变异

（38）A. 封装 B. 多态 C. 继承 D. 变异

试题分析

本题考查面向对象的方法学知识，是常考的知识点。

◆封装性

封装性是一种信息隐蔽技术，它使系统分析员能够清晰地标明他们所提供的服务界面，用户和

应用程序员则只看得见对象提供的操作功能（即封装面上的信息），看不到其中的数据或操作代码细节。封装就是将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体，也就是将数据与操作数据的源代码进行有机的结合，形成"类",其中数据和函数都是类的成员。

◆多态性

多态性是指同一个操作作用于不同的对象可以有不同的解释，产生不同的执行结果。

◆继承性

继承是指在某个类的层次关联中，不同的类共享属性和操作的一种机制。一个父类可以有多个子类。父类描述了这些子类的公共属性和操作，子类中还可以定义其自己的属性和操作。如果一个子类只有唯一的一个父类，这种继承称为单一继承。如果一个子类有多个父类，可以从多个父类中继承特性，这种继承称为多重继承。

根据上面的定义，第（37）空是A选项，第（38）空是C选项。

试题答案

A C

 作者：希赛教育软考学院 来源：希赛网 2014年02月08日

例题41

例题41（2008年5月试题39-40）

（39）以静态或动态的连接方式，为应用程序提供一组可使用的类。（40）除了提供可被应用程序调用的类以外，还基本实现了一个可执行的架构。

（39）A. 函数库 B. 类库 C. 框架 D. 类属

（40）A. 函数库 B. 类库 C. 框架 D. 类属

试题分析

本题考查面向对象方法学的类及框架知识。

1.什么是类库

大量的程序员都尽可能多地利用现有的代码。程序员经常购买那些包装成库的代码。当c语言流行时，代码库就是函数库。例如，可以购买一个数学库，该库含有完成微积分和代数运算的函数。通常，可以在程序代码中使用一个包含文件来指定一个函数库，可以静态或动态地链接这个函数库。静态链接意味者库代码直接集成到程序中。在这种情况下，程序不依赖于其他文件，但文件的大小可能很大。动态链接意味着程序有库的版本信息，其代码存放于一个单独的文件中，这个文件称为动态链接库（dll）。只有程序运行期间调用dll中的函数时，才加载dll到内存。dll作为一个独立的实体存在于内存中，可以同时被多个程序访问。

出现c++后，函数库转变为类库。两者的区别在于函数库只包含一系列函数，而类库是用面向对象编程的原理设计的，例如，可以为数据结构做一个类库，该库包括一个链接列表的类。如果使用一个函数库代替，那么链接列表会独立于操作它的函数。另一方面，使用类库时，链接列表和操作它的函数存在于同一个文件的同一个类中。然而，正如使用函数库一样，使用类库涉及到包含文

件和链接一个静态库。最近，已经从类库发展到模板库。其原因是c++编译器现在已经能够处理c++模板。模板库提供了一系列优于类库的优点。要使用模板库，只要在程序中加入包含文件即可，不用链接到库，因为所有的库代码已经包含在该包含文件中了。vc有三个供开发者使用的库：一个类库（mfc），二个模板库（atl和stl）。

2.什么是框架

框架是对于那些试图在他们所关心的领域构造一个复杂软件系统的用户而言的。因为它是处于特定领域中，所以应用系统的体系结构在许多不同的方面具有一定的相似性。框架利用一系列的对象和它们之间的接口来对应静态和恒定结构的端口，并保留友好界面使用户能够很容易完成变化的、不稳定的剩余部分而得到一个新应用程序。任何框架都是特定领域的框架，一个框架可以包含一个或多个模式。

框架是一种软件重用技术，它是一个应用软件系统的部分或整体的可重用设计。框架的具体表现为一组抽象类以及其实例（对象）之间的相互作用方式。它是对于一个软件系统的全部或部分的可复用设计。

在一个专用领域内构造框架时，把握框架的共同点是关键因素。框架一旦被建立，其适应性和可扩展性就被提到了首要地位。框架的直接目的是被复用，以减少建立一个新应用系统的工作量。只有当理解一个框架的代价少于建立一个新系统的代价时，框架的复用才成为可能。构造框架是一个逐渐积累的过程，就是说不断有新发现的共同点和新的构件被加入，同时那些老的、不必要的部件被删除，以此来保证整个应用系统的性能，这样新旧构件之间的接口直接影响框架的适应性和可扩展性。任何框架都不能被直接复用，除非要建立的应用系统非常简单或者与框架极为相似，除此之外，都要对框架进行或多或少的修改和扩展才能被复用。

3.框架和类库的不同

我们首先要给类库一个明确的定义，对类库的广义描述为：类库以库文件的形式存在，库文件中包含了事先定义好的类。从广义的角度看，面向对象框架的存在形式可以看作一个类库，它是建立在对多态性和动态绑定的系统化的广泛使用基础之上，这些说起来很简单，但现在已建立的开发方法很少有支持对框架设计的。不过一些新的方法原理，例如模式和抽取，表现了对面向对象方法有意义的支持。但在传统意义上，框架是不同于类库的，框架包含更多的内容。框架是对协作完成一系列相关责任的类集合的抽象设计。框架和类库的一些主要区别与联系如下：

①在类库中是由用户实例化抽象类，类库构件被单独使用；而在框架中要有抽象类的一些具体子类，应用软件开发者通过修改现存的类或定义新的子类以扩展现存的类来使用框架。

②在类库中是用户在需要的时候调用函数，不预定义控制流；而框架使用了与用户之间的反向控制流，使用框架时，通常只是实现一些回调函数或者使一些类适用于上下文，然后调用一个单一的方法或过程，框架将完成其余的工作，在适当的时候和地点调用必要的函数或方法。这就是框架的Hollywood法则（"Don't call us, we'll call you"）。

③使用类库只需要了解类的外部接口，但必须要定义应用软件的整体结构；而框架的使用者要了解框架的抽象设计和类的内部结构来进行修改和扩展，因此学习框架要比类库更加困难，同时有被错误使用的危险，但框架被复用的潜力大大超过了类库。

④在类库中没有定义缺省行为，而在框架中提供缺省行为。

试题答案

B　C

例题42

例题42（2008年5月试题41-42）

已知某子系统为外界提供功能服务，但该子系统中存在很多粒度十分小的类，不便被外界系统直接使用，采用（41）设计模式可以定义一个高层接口，这个接口使得这一子系统更加容易使用；当不能采用生成子类的方法进行扩充时，可采用（42）设计模式动态地给一个对象添加一些额外的职责。

（41）A. Facade（外观）　B. Singleton（单件）

C. Participant（参与者）　D. Decorator（装饰）

（42）A. Facade（外观）_____ B. Singleton（单件）

C. Participant（参与者）　D. Decorator（装饰）

试题分析

本题考查设计模式，是2007年5月的第42~44空的翻版。

四种设计模式的作用：

外观（Facade）：为子系统中的一组功能调用提供一个一致的接口，这个接口使得这一子系统更加容易使用；

装饰（Decorate）：当不能采用生成子类的方法进行扩充时，动态地给一个对象添加一些额外的功能；

单件（Singleton）：保证一个类仅有一个实例，并提供一个访问它的全局访问点；

模板方法（Template Method）：在方法中定义算法的框架，而将算法中的一些操作步骤延迟到子类中实现。

所以第（41）空选择A答案。第（42）空选择D答案。

试题答案

A　D

例题43

例题43（2008年5月试题43-44）

（43）设计模式将抽象部分与它的实现部分相分离，使它们都可以独立地变化。下图为该设计模式的类图，其中，（44）用于定义实现部分的接口。

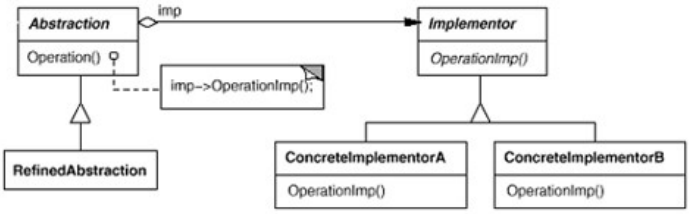


图5-11 类图

（43）A. Singleton（单件） B. Bridge（桥接）

C. Composite（组合） D. Facade（外观）

（44）A. Abstraction_____B. ConcreteImplementorA

C. ConcreteImplementorB D. Implementor

试题分析

本题考查面向对象方法学的设计模式。

设计模式/软件设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

基本的设计模式：

Abstract Factory:提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

Adapter:将一个类的接口转换成客户希望的另外一个接口。adapter模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

Bridge:将抽象部分与它的实现部分分离，使它们都可以独立地变化。

Builder:将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

Chain of Responsibility:为解除请求的发送者和接收者之间耦合，而使多个对象都有机会处理这个请求。将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它。

Command:将一个请求封装为一个对象，从而使你可用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可取消的操作。

Composite:将对象组合成树形结构以表示"部分-整体"的层次结构。它使得客户对单个对象和复合对象的使用具有一致性。

Decorator:动态地给一个对象添加一些额外的职责。就扩展功能而言， 它比生成子类方式更为灵活。

Facade:为子系统中的一组接口提供一个一致的界面， Facade模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。

Factory Method:定义一个用于创建对象的接口，让子类决定将哪一个类实例化。Factory Method使一个类的实例化延迟到其子类。

Flyweight:运用共享技术有效地支持大量细粒度的对象。

Interpreter:给定一个语言， 定义它的文法的一种表示，并定义一个解释器， 该解释器使用该表示来解释语言中的句子。

Iterator:提供一种方法顺序访问一个聚合对象中各个元素， 而又不需暴露该对象的内部表示。

Mediator:用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。

Memento:在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。这样以后就可将该对象恢复到保存的状态。

Observer:定义对象间的一种一对多的依赖关系，以便当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动刷新。

Prototype:用原型实例指定创建对象的种类，并且通过拷贝这个原型来创建新的对象。

Proxy:为其他对象提供一个代理以控制对这个对象的访问。

Singleton:保证一个类仅有一个实例，并提供一个访问它的全局访问点。

State:允许一个对象在其内部状态改变时改变它的行为。对象看起来似乎修改了它所属的类。

Strategy:定义一系列的算法，把它们一个个封装起来， 并且使它们可相互替换。本模式使得算法的变化可独立于使用它的客户。

Template Method:定义一个操作中的算法的骨架，而将一些步骤延迟到子类中。Template Method使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。

Visitor:表示一个作用于某对象结构中的各元素的操作。它使你可以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。

本题考查了Bridge（桥接）设计模式，所以第（43）空选择B选项。

从上面的类图中可以看出，类Abstraction和类Implementor之间存在聚集关系（整体和部分的关系），而ConcreteImplementorA和ConcreteImplementorB都是整体类Abstraction类的部分类，所以中间的Implementor用于定义实现部分的接口，故第（44）空选择D答案。

试题答案

B D

例题44

例题44（2008年5月试题45-47）

在UML类图中，类与类之间存在依赖（Dependency）、关联（Association）、聚合（Aggregation）、组合（Composition）和继承（Inheritance）五种关系，其中，（45）关系表明类之间的相互联系最弱，（46）关系表明类之间的相互联系最强，聚合（Aggregation）的标准UML图形表示是（47）.

（45）A. 依赖　B. 聚合　C. 组合　D. 继承

（46）A. 依赖　B. 聚合　C. 组合　D. 继承

（47）A.  　B.  　C.  　D.

试题分析

本题考查面向对象方法学的UML统一建模语言。

请参看例题15分析。

试题答案

A D B

例题45

例题45（2008年12月试题15）

在面向对象系统中，用___ _关系表示一个较大的"整体"类包含一个或多个较小的"部分"类。

A. 泛化　B. 聚合　C. 概化　D. 合成

试题分析

本题是面向象系统的基础概念题。泛化关系也就是父子关系，子类能继承父类的所有特征。聚合即整体与多个较小部分之间的关系。没有"概化"与"合成"这种说法。

试题答案

B

例题46

例题46（2008年12月试题37-38）

面向对象分析与设计中的（37）是指一个模块在扩展性方面应该是开放的，而在更改性方面应该是封闭的；而（38）是指子类应当可以替换父类并出现在父类能够出现的任何地方。

（37）A. 开闭原则　B. 替换原则　C. 依赖原则　D. 单一职责原则

（38）A. 开闭原则　B. 替换原则　C. 依赖原则　D. 单一职责原则

试题分析

面向对象设计有五大原则，他们分别是：单一职责原则（SRP）、开放封闭原则OCP、Liskov替换原则（LSP）、依赖倒置原则（DIP）、接口隔离原则（ISP）。

1、单一职责原则（Single Responsibility Principle SRP）：就一个类来说，应该仅有一个引起它变化的原因。也就是说，一个类应该只有一个职责。如果有多个职责，那么就相当于把这些指责耦合在一起，一个职责的变化就可能削弱或者抑制了这个类完成其他职责的能力，引起类的变化的原因就会有多个。所以在构造一个类时，将类的不同职责分离至两个或多个类中（或者接口中），确保引起该类变化的原因只有一个。

2、开放封闭法则（Open Closed Principle OCP）：软件组成实体应该是可扩展的，但是不可修改的。 开放-封闭法认为我们应该试图去设计出永远也不需要改变的模块。我们可以添加新代码来

扩展系统的行为。我们不能对已有的代码进行修改。

符合OCP的模块需满足两个标准：

可扩展，即"对扩展是开放的"（Open For Extension）-模块的行为可以被扩展，以需要满足新的需求。

不可更改，即"对更改是封闭的"（Closed for Modification）-模块的源代码是不允许进行改动的。

3、替换原则（Liskov Substitution Principle LSP）：子类应当可以替换父类并出现在父类能够出现的任何地方。这个原则是Liskov于1987年提出的设计原则。它同样可以从Bertrand Meyer 的DBC（Design by Contract）的概念推出。我们以圆和椭圆为例，圆是椭圆的一个特殊子类。因此任何出现椭圆的地方，圆均可以出现。但反过来就可能行不通。

运用替换原则时，我们尽量把类B设计为抽象类或者接口，让C类继承类B（接口B）并实现操作A和操作B,运行时，类C实例替换B,这样我们即可进行新类的扩展（继承类B或接口B），同时无须对类A进行修改。

4、依赖原则（Dependency Inversion Principle DIP）：在进行业务设计时，与特定业务有关的依赖关系应该尽量依赖接口和抽象类，而不是依赖于具体类。具体类只负责相关业务的实现，修改具体类不影响与特定业务有关的依赖关系。在结构化设计中，我们可以看到底层的模块是对高层抽象模块的实现（高层抽象模块通过调用底层模块），这说明，抽象的模块要依赖具体实现相关的模块，底层模块的具体实现发生变动时将会严重影响高层抽象的模块，显然这是结构化方法的一个"硬伤".面向对象方法的依赖关系刚好相反，具体实现类依赖于抽象类和接口。为此，我们在进行业务设计时，应尽量在接口或抽象类中定义业务方法的原型，并通过具体的实现类（子类）来实现该业务方法，业务方法内容的修改将不会影响到运行时业务方法的调用。

5、接口分离原则（the Interface Segregation Principle ISP）：采用多个与特定客户类有关的接口比采用一个通用的涵盖多个业务方法的接口要好。ISP原则是另外一个支持诸如COM等组件化的使能技术。缺少ISP,组件、类的可用性和移植性将大打折扣。这个原则的本质相当简单。如果你拥有一个针对多个客户的类，为每一个客户创建特定业务接口，然后使该客户类继承多个特定业务接口将比直接加载客户所需所有方法有效。

试题答案

A C

上一节　本书简介　下一节

例题47

例题47（2008年12月试题39）

在选择某种面向对象语言进行软件开发时，不需要着重考虑的因素是，该语言_____.

A. 将来是否能够占据市场主导地位

B. 类库是否丰富

C. 开发环境是否成熟

D. 是否支持全局变量和全局函数的定义

试题分析

答案选项中的"是否支持全局变量和全局函数的定义"是不需要考虑的。任何一种面向对象的语言都会有相应的处理机制，所以不需要考虑。

试题答案

D

例题48

例题48（2008年12月试题40-41）

（40）限制了创建类的实例数量，而（41）将一个类的接口转换成客户希望的另外一个接口，使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

（40）A. 命令模式（Command）　B. 适配器模式（Adapter）

C. 策略模式（Strategy）　D. 单例模式（Singleton）

（41）A. 命令模式（Command）　B. 适配器模式（Adapter）

C. 策略模式（Strategy）　D. 单例模式（Singleton）

第 5 章：面向对象方法学　作者：希赛教育软考学院　来源：希赛网　2014年02月08日

例题48

本题考查常见设计模式的特点。涉及设计模式有：命令模式（Command）、适配器模式（Adapter）、策略模式（Strategy）、单例模式（Singleton）。

命令模式（Command）：将一个请求封装为一个对象，从而达到用不同的请求对客户进行参数化的目标。

适配器模式（Adapter）：适配器模式的用意是将接口不同而功能相同或者相近的两个接口加以转换。

适配器模式中存在三个角色：一是目标Target,它是目标接口，最近得具有它的接口形式；二是被适配的源类Adaptee,它的功能与Target所描述的接口有相似的是它无法成为Target的实现类，所提供的方法不全，或者不完全符合要求；三是适配器类,它的功能是把Adaptee转化为目标接口Target.

单例模式（Singleton）：单例模式的意图是保证一个类仅有一个实例，并提供一个访问它的全局访问点。

策略模式（Strategy）：策略模式可以将一系列的算法一个个封装起来，并且使它们可相互替换。

试题答案

D B

例题49

例题49（2008年12月试题45-47）

在UML的各种视图中，（45）显示外部参与者观察到的系统功能；（46）从系统的静态结构和动态行为角度显示系统内部如何实现系统的功能；（47）显示的是源代码以及实际执行代码的组织结构。

（45）A. 用例视图 B. 进程视图　C. 实现视图 D. 逻辑视图

（46）A. 用例视图 B. 进程视图　C. 实现视图 D. 逻辑视图

（47）A. 用例视图 B. 进程视图　C. 实现视图 D. 逻辑视图

试题分析

UML的五个系统视图：

（1）逻辑视图：用来显示系统内部的功能是怎样设计的，它利用系统的静态结构和动态行为来刻画系统功能。静态结构描述类、对象和它们之间的关系等。动态行为主要描述对象之间的动态协作。

（2）进程视图（并发视图）：用来显示系统的并发工作状况。并发视图将系统划分为进程和处理机方式，通过划分引入并发机制，利用并发高效地使用资源、并行执行和处理异步事件。它是逻辑视图的一次执行实例。

（3）实现视图（组件视图）：用来显示代码组件的组织方式。它描述了实现模块和它们之间的依赖关系。

（4）部署视图：用来显示系统的物理架构，即系统的物理展开。比如，计算机和设备以及它们之间的联接方式。其中计算机和设备称为结点。

（5）用例视图：最基本的需求分析模型，以外部参与者的角度来看待系统，他主要说明了谁要使用系统以及他们使用了该系统可以做些什么。

由此知本题的答案为：A、D、C

试题答案

A D C

例题1

6.2 试题精解