窗体顶端

# 面向对象

您现在的位置：[希赛网](http://www.educity.cn/" \o "希赛网) > [云阅读](http://www.educity.cn/jiaocheng/) > [软件设计师考试试题分类精解（2018版）](http://www.educity.cn/jiaocheng/zt251.html" \t "http://www.educity.cn/jiaocheng/_blank) > **试题1(2017年下半年试题15-16)**

第 8 章：面向对象作者：[希赛软考学院](http://www.educity.cn/jiaocheng/a%cf%a3%c8%fc%c8%ed%bf%bc%d1%a7%d4%ba.html" \o "希赛软考学院" \t "http://www.educity.cn/jiaocheng/_blank)    来源：希赛软考学院    2017年11月21日

# **试题1(2017年下半年试题15-16)**

某医院预约系统的部分需求为：患者可以查看医院发布的专家特长介绍及其就诊时间；系统记录患者信息，患者预约特定时间就诊。用DFD对其进行功能建模时，患者是（  ）；用ERD对其进行数据建模时，患者是（  ）。  
（15）A．外部实体  
B.加工  
C.数据流  
D.数据存储  
  
（16）A．实体  
B.属性  
C.联系  
D.弱实体

**试题分析**

1、患者不涉及加工，为外部实体。  
2、患者有其信息，所以为实体。

**试题答案**

（15）A（16）A

# **试题2(2017年下半年试题37-38)**

在面向对象方法中，将逻辑上相关的数据以及行为绑定在一起，使信息对使用者隐蔽称为（  ）。当类中的属性或方法被设计为private时，（  ）可以对其进行访问。  
（37）A．抽象  
B.继承  
C.封装  
D.多态  
  
（38）A．应用程序中所有方法  
B.只有此类中定义的方法  
C.只有此类中定义的public方法  
D.同一个包中的类中定义的方法

**试题分析**

1、封装是指利用抽象数据类型将数据和基于数据的操作封装在一起，使其构成一个不可分割的独立实体，数据被保护在抽象数据类型的内部，尽可能地隐藏内部的细节，只保留一些对外接口使之与外部发生联系。  
2、只有此类中定义的方法可以对私有成员进行访问。

**试题答案**

（37）C（38）B

# **试题3(2017年下半年试题39)**

采用继承机制创建子类时，子类中（  ）。  
（39）A．只能有父类中的属性  
B.只能有父类中的行为  
C.只能新增行为  
D.可以有新的属性和行为

**试题分析**

子类相对于父类，要更加特殊。所以会有新的成员来描述其特殊。

**试题答案**

（39）D

# **试题4(2017年下半年试题40)**

面向对象分析过程中，从给定需求描述中选择（  ）来识别对象。  
（40）A．动词短语  
B.名词短语  
C.形容词  
D.副词

**试题分析**

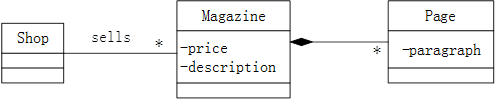
名词短语暗示类及其属性 动词和动词短语暗示对职责或操作。

**试题答案**

（40）B

# **试题5(2017年下半年试题41-43)**

如下所示的UML类图中，Shop和Magazine之间为（  ）关系，Magazine和Page之间为（  ）关系。UML类图通常不用于对（  ）进行建模。



（41）A．关联  
B.依赖  
C.组合  
D.继承  
  
（42）A．关联  
B.依赖  
C.组合  
D.继承  
  
（43）A．系统的词汇  
B.简单的协作  
C.逻辑数据库模式  
D.对象快照

**试题分析**

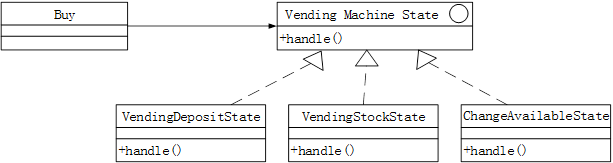
1、关联关系的表示图法。  
2、实心凌形表示组合。  
3、对象快照是对象图的。

**试题答案**

（41）A（42）C（43）D

# **试题6(2017年下半年试题44-47)**

自动售货机根据库存、存放货币量、找零能力、所选项目等不同，在货币存入并进行选择时具有如下行为：交付产品不找零 ；交付产品并找零；存入货币不足而不提供任何产品；库存不足而不提供任何产品。这一业务需求适合采用（  ）模式设计实现，其类图如下图所示，其中（  ）是客户程序使用的主要接口，可用状态来对其进行配置。此模式为（  ），体现的最主要的意图是（  ）。



（44）A．观察者（Observer）  
B.状态（State）  
C.策略（Strategy）  
D.访问者（Visitor）  
  
（45）A．VendingMachineState  
B.Buy  
C.VendingDepositState  
D.VendingStockState  
  
（46）A．创建型对象模式  
B.结构型对象模式  
C.行为型类模式  
D.行为型对象模式  
  
（47）A．当一个对象状态改变时所有依赖它的对象得到通知并自动更新  
B.在不破坏封装性的前提下，捕获对象的内部状态并在对象之外保存  
C.一个对象在其内部状态改变时改变其行为  
D.将请求封装为对象从而可以使用不同的请求对客户进行参数化

**试题分析**

1、有State关键词，状态模式：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的类。在很多情况下，一个对象的行为取决于一个或多个动态变化的属性，这样的属性叫做状态，这样的对象叫做有状态的(stateful)对象，这样的对象状态是从事先定义好的一系列值中取出的。当一个这样的对象与外部事件产生互动时，其内部状态就会改变，从而使得系统的行为也随之发生变化。  
2、是整个状态模式的基类也是对外的接口。

**试题答案**

（44）B（45）A（46）D（47）C

# **试题7(2017年上半年试题37-38)**

在面向对象方法中，两个及以上的类作为一个类的超类时，称为（  ），使用它可能造成子类中存在（  ）的成员。  
  
（37）A．多重继承  
B.多态  
C.封装  
D.层次继承  
  
（38）A．动态  
B.私有  
C.公共  
D.二义性

**试题分析**

多重继承是指一个类有多个父类，正是题目所述的情况。多重继承可能造成混淆的情况，出现二义性的成员。

**试题答案**

（37）A（38）D

# **试题8(2017年上半年试题39)**

采用面向对象方法进行软件开发，在分析阶段，架构师主要关注系统的（  ）。  
  
（39）A．技术  
B.部署  
C.实现  
D.行为

**试题分析**

采用面向对象方法进行软件开发，分析阶段，架构师主要关注系统的行为，即系统应该做什么。

**试题答案**

（39）D

# **试题9(2017年上半年试题40)**

在面向对象方法中,多态指的是（  ）。  
  
（40）A．客户类无需知道所调用方法的特定子类的实现  
B. 对象动态地修改类  
C.一个对象对应多张数据库表  
D. 子类只能够覆盖父类中非抽象的方法

**试题分析**

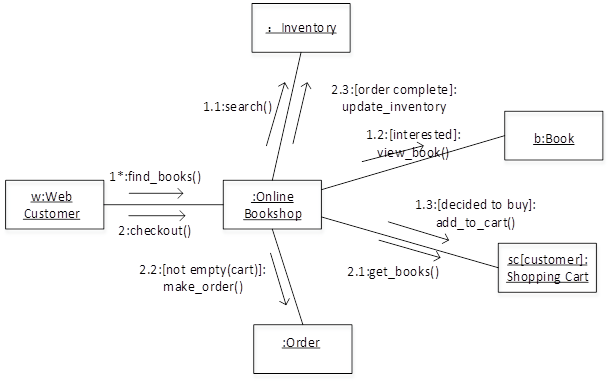
多态：同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果。在运行时，可以通过指向基类的指针，来调用实现派生类中的方法。也就是说客户类其实在调用方法时，并不需要知道特定子类的实现，都会用统一的方式来调用。

**试题答案**

（40）A

# **试题10(2017年上半年试题41-43)**

以下UML图是（  ），图中IMG_269和IMG_270表示（  ），IMG_271和IMG_272表示（  ）。



（41）A．序列图  
B.状态图  
C.通信图  
D.活动图  
  
（42）A．类  
B.对象  
C.流名称  
D.消息  
  
（43）A．类  
B.对象  
C.流名称  
D.消息

**试题分析**

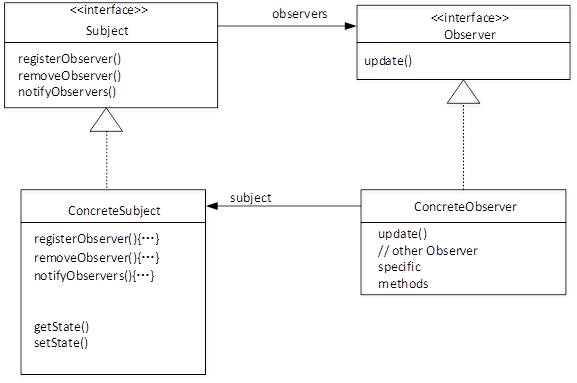
从图示可以了解到，题目中的图是通信图。通信图描述的是对象和对象之间的关系，即一个类操作的实现。简而言之就是，对象和对象之间的调用关系，体现的是一种组织关系。该图明显表达的是对象与对象之间的关系。其中如果一个框中的名称中带有“:”号，说明这表示的是一个对象，“:”号前的部分是对象名，“:”号后面的部分是类名。而对象之间连线上面的箭头所标识的是对象之间通信的消息。

**试题答案**

（41）C（42）B（43）D

# **试题11(2017年上半年试题44-45)**

下图所示为观察者(Obserrver)模式的抽象示意图，其中（  ）知道其观察者，可以有任何多个观察者观察同一个目标;提供住处和删除观察者对象的接口。此模式体现的最主要的特征是（  ）。



（44）A．Subject  
B.Observer  
C.ConcreteSubject  
D.ConcreteObserver  
  
（45）A．类应该对扩展开放，对修改关闭  
B. 使所要交互的对象尽量松耦合  
C. 组合优先于继承使用  
D. 仅与直接关联类交互

**试题分析**

观察者将自己注册到事件，那么具体的事件就知道了自己的观察者。观察者和事件都有自己的抽象，当实现具体的观察者和事件的时候都要实现相应接口，所以对扩展是开放的。

**试题答案**

（44）C（45）A

# **试题12(2017年上半年试题46-47)**

装饰器 (Decorator) 模式用于（  ）;外观 (Facade) 模式用于（  ）。  
①将一个对象加以包装以给客户提供其希望的另外一个接口  
②将一个对象加以包装以提供一些额外的行为  
③将一个对象加以包装以控制对这个对象的访问  
④将一系列对象加以包装以简化其接口  
  
（46）A．①  
B.②  
C.③  
D.④  
  
（47）A．①  
B.②  
C.③  
D.④

**试题分析**

装饰模式是一种对象结构型模式，可动态地给一个对象增加一些额外的职责，就增加对象功能来说，装饰模式比生成子类实现更为灵活。通过装饰模式，可以在不影响其他对象的情况下，以动态、透明的方式给单个对象添加职责；当需要动态地给一个对象增加功能，这些功能可以再动态地被撤销时可使用装饰模式；当不能采用生成子类的方法进行扩充时也可使用装饰模式。  
外观模式是对象的结构模式，要求外部与一个子系统的通信必须通过一个统一的外观对象进行，为子系统中的一组接口提供一个一致的界面，外观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。

**试题答案**

（46）B（47）D

# **试题13(2016年下半年试题37)**

在面向对象方法中，不同对象收到同一消息可以产生完全不同的结果，这一现象称为（  ）。在使用时，用户可以发送一个通用的消息，而实现的细节则由接收对象自行决定。  
（37）A．接口  
B.继承  
C.覆盖  
D.多态

**试题分析**

本题考察面向对象多态的概念  
在收到消息时，对象要予以响应。不同的对象收到同一消息可以产生完全不同的结果，这种现象就叫多态。

**试题答案**

（37）D

# **试题14(2016年下半年试题38)**

在面向对象方法中，支持多态的是（  ）。  
（38）A．静态分配  
B.动态分配  
C.静态类型  
D.动态绑定

**试题分析**

动态绑定是实现多态的基础

**试题答案**

（38）D

# **试题15(2016年下半年试题39)**

面向对象分析的目的是为了获得对应用问题的理解，其主要活动不包括（  ）。  
（39）A．认定并组织对象  
B.描述对象间的相互作用  
C.面向对象程序设计  
D.确定基于对象的操作

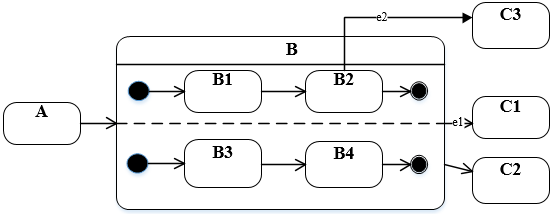
**试题分析**

面向对象分析的任务是了解问题域所涉及的对象、对象间的关系和操作，然后构造问题的对象模型。

**试题答案**

（39）C

# **试题16(2016年下半年试题40)**

如下所示的UML状态图中，（  ）时，不一定会离开状态B  
  
（40）A．状态B中的两个结束状态均达到  
B.在当前状态为B2时，事件e2发生  
C.事件e2发生  
D.事件e1发生

**试题分析**

e2发生时只是从B2出来，还有B4等，不一定离开了B

**试题答案**

（40）C

# **试题17(2016年下半年试题41)**

以下关于UML状态图中转换（transition）的叙述中，不正确的是（  ）。  
  
（41）A．活动可以在转换时执行也可以在状态内执行  
B.监护条件只有在相应的事件发生时才进行检查  
C.一个转换可以有事件触发器、监护条件和一个状态  
D.事件触发转换

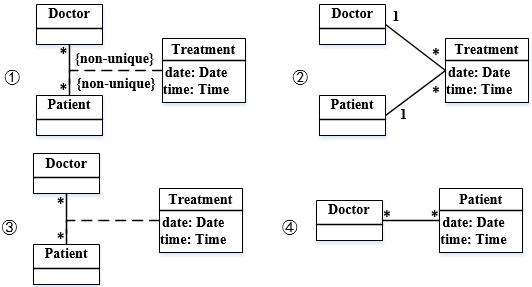
**试题分析**

转换应该有多个状态

**试题答案**

（41）C

# **试题18(2016年下半年试题42-43)**

下图①②③④所示是UML（  ）。现有场景：一名医生（Doctor）可以治疗多位病人（Patient），一位病人可以由多名医生治疗，一名医生可能多次治疗同一位病人。要记录哪名医生治疗哪位病人时，需要存储治疗（Treatment）的日期和时间。以下①②③④图中（  ）。是描述此场景的模型。  
  
（42）A．用例图  
B.对象图  
C.类图  
D.协作图  
  
（43）A．①  
B.②  
C.③  
D.④

**试题分析**

1是标准的类图  
2.根据题中关系描述，应该是3.这时候医生跟病人关系是多对多，且治疗这个处理过程正确

**试题答案**

（42）C（43）C

# **试题19(2016年下半年试题44-45)**

（  ）模式定义一系列的算法，把它们一个个封装起来，并且使它们可以相互替换，使得算法可以独立于使用它们的客户而变化。以下（  ）情况适合选用该模式。  
①一个客户需要使用一组相关对象  
②一个对象的改变需要改变其它对象      
③需要使用一个算法的不同变体  
④许多相关的类仅仅是行为有异  
  
（44）A．命令（Command）  
B.责任链（Chain of Responsibility）  
C.观察者（Observer）  
D.策略（Strategy）  
  
（45）A．①②  
B.②③  
C.③④  
D.①④

**试题分析**

策略模式定义了一系列的算法，并将每一个算法封装起来，而且使它们还可以相互替换。策略模式让算法独立于使用它的客户而独立变化。  
应用场景：  
1、 多个类只区别在表现行为不同，可以使用Strategy模式，在运行时动态选择具体要执行的行为。  
2、 需要在不同情况下使用不同的策略(算法)，或者策略还可能在未来用其它方式来实现。  
3、 对客户隐藏具体策略(算法)的实现细节，彼此完全独立。

**试题答案**

（44）D（45）C

# **试题20(2016年下半年试题46-47)**

（  ）模式将一个复杂对象的构建与其表示分离，使得同样的构建过程可以创 建不同的表示。以下（  ）情况适合选用该模式。  
①抽象复杂对象的构建步骤  
②基于构建过程的具体实现构建复杂对象的不同表示  
③一个类仅有一个实例  
④一个类的实例只能有几个不同状态组合中的一种  
（46）A．生成器（Builder）  
B.工厂方法（Factory Method）  
C.原型（Prototype）  
D.单例（ Singleton）  
  
（47）A．①②  
B.②③  
C.③④  
D.①④

**试题分析**

生成器模式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。  
实用范围  
1 当创建复杂对象的算法应该独立于该对象的组成部分以及它们的装配方式时。  
2 当构造过程必须允许被构造的对象有不同表示时。

**试题答案**

（46）A（47）A

# **试题21(2016年上半年试题37-38)**

在面向对象方法中，（  ）是父类和子类之间共享数据和方法的机制。子类在原有父类接口的基础上，用适合于自己要求的实现去置换父类中的相应实现称为（  ）。   
（37）A．封装   
B.继承   
C.覆盖   
D.多态   
  
（38）A．封装   
B.继承   
C.覆盖   
D.多态

**试题分析**

继承是父类和子类之间共享数据和方法的机制。

覆盖是子类的方法覆盖了基类的方法，以实现不同的功能，或者对父类的功能进行扩充。

**试题答案**

（37）B（38）C

# **试题22(2016年上半年试题39)**

在UML用例图中，参与者表示（  ）。  
（39）A．人、硬件或其他系统可以扮演的角色   
B.可以完成多种动作的相同用户   
C.不管角色的实际物理用户   
D.带接口的物理系统或者硬件设计

**试题分析**

参与者是指存在于系统外部并直接与系统进行交互的人、系统、子系统或类的外部实体的抽象。

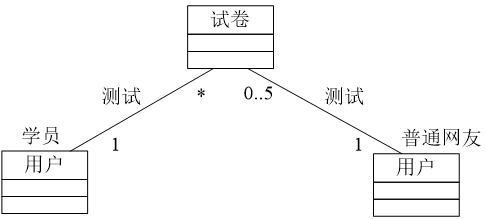
**试题答案**

（39）A

# **试题23(2016年上半年试题40)**

UML中关联是一个结构关系，描述了一组链。两个类之间（  ）关联。  
（40）A．不能有多个   
B.可以有多个由不同角色标识的   
C.可以有任意多个   
D.的多个关联必须聚合成一个

**试题分析**

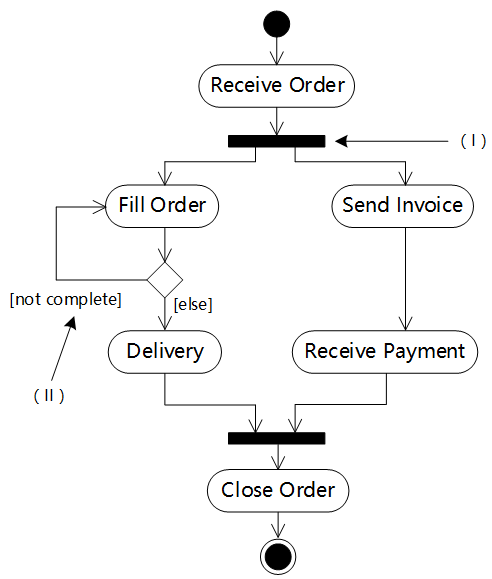
两个类之间可以由不同角色标识存在多个关联，如：  


**试题答案**

（40）B

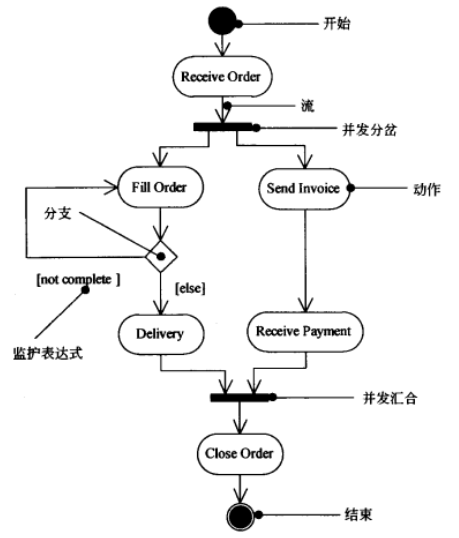
# **试题24(2016年上半年试题41-43)**

如下所示的UML图是（  ），图中（Ⅰ）表示（  ），（Ⅱ）表示（  ）。



（41）A．序列图  
B.状态图  
C.通信图  
D.活动图  
  
（42）A．合并分叉  
B.分支  
C.合并汇合  
D.流  
  
（43）A．分支条件  
B.监护表达式  
C.动作名  
D.流名称

**试题分析**

本题所涉及到的图为活动图，该图容易与状态图混淆，对于初学者，可以把握一个原则来判断，即：状态图中每个结点对应的是状态，而状态与状态之间的变迁涉及到事件触发，所以在状态图中，每条箭线上都会有事件，而活动图则不一定有。  
在活动图中，各个组成部分的标准名称为：  


**试题答案**

（41）D（42）A（43）B

# **试题25(2016年上半年试题44)**

为图形用户界面（GUI）组件定义不同平台的并行类层次结构，适合采用（  ）模式。  
（44）A．享元（Flyweight）   
B.抽象工厂（Abstract Factory）   
C.外观（Facade））   
D.装饰器（Decorator）

**试题分析**

抽象工厂模式的特点

**试题答案**

（44）B

# **试题26(2016年上半年试题45)**

（  ）设计模式将一个请求封装为一个对象，从而使得可以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，以及支持可撤销的操作。  
（45）A．命令（Command）   
B.责任链（Chain of Responsibility）   
C.观察者（Observer）   
D.策略（Strategy）

**试题分析**

命令模式的特点为：将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化，将请求排队或记录请求日志，支持可撤销的操作

**试题答案**

（45）A

# **试题27(2016年上半年试题46)**

（  ）设计模式最适合用于发布/订阅消息模型，即当订阅者注册一个主题后，此主题有新消息到来时订阅者就会收到通知。  
（46）A．适配器（Adapter）   
B.通知（Notifier）   
C.观察者（Observer）   
D.状态（State）

**试题分析**

观察者模式：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自动更新。

**试题答案**

（46）C

# **试题28(2016年上半年试题47)**

因使用大量的对象而造成很大的存储开销时，适合采用（  ）模式进行对象共享，以减少对象数量从而达到较少的内存占用并提升性能。  
（47）A．组合（Composite）   
B.享元（Flyweight）   
C.迭代器（Iterator）   
D.备忘（Memento）

**试题分析**

享元模式提供支持大量细粒度对象共享的有效方法。

**试题答案**

（47）B

# **试题29(2015年下半年试题37-38)**

在面向对象的系统中，对象是运行时实体，其组成部分不包括（  ）；一个类定义了一组大体相似的对象，这些对象共享（  ）。

（37）A．消息   
B.行为（操作）   
C.对象名   
D.状态   
  
（38）A．属性和状态   
B.对象名和状态   
C.行为和多重度   
D.属性和行为

**试题分析**

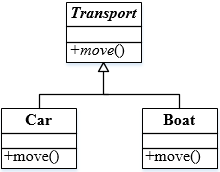
对象的组成部分包括：对象名，状态（属性），行为（操作）。一个类产生的对象共享属性和行为。

**试题答案**

（37）A（38）D

# **试题30(2015年下半年试题39)**

如下所示的UML类图中，Car和Boat类中的move()方法（  ）了Transport类中的move()方法。



（39）A．继承   
B.覆盖（重置）   
C.重载   
D.聚合

**试题分析**

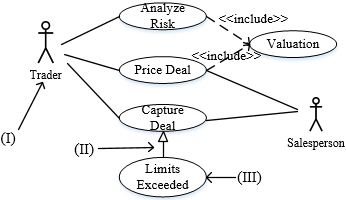
覆盖：子类重写父类的方法。  
重载：一个类可以有多个同名而参数类型不同的方法。

**试题答案**

（39）B

# **试题31(2015年下半年试题40-42)**

如下所示的**UML**图中，(I)是（  ），(Ⅱ)是（  ），(Ⅲ)是（  ）。

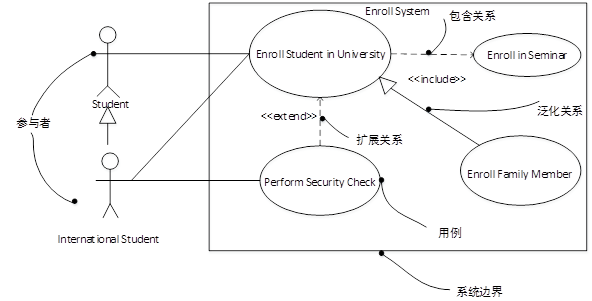


（40）A．参与者  
B.用例  
C.泛化关系  
D.包含关系  
  
（41）A．参与者  
B.用例  
C.泛化关系  
D.包含关系  
  
（42）A．参与者  
B.用例  
C.泛化关系  
D.包含关系

**试题分析**

本题考查统一建模语言（UML）的基本知识。

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。用例图通常包括用例、参与者，以及用例之间的扩展关系（<<extend>>）和包含关系（<<include>>），参与者和用例之间的关联关系，用例与用例以及参与者与参与者之间的泛化关系。如下图所示。



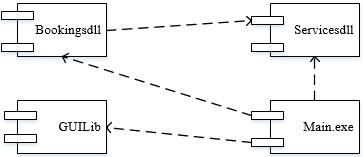
用例图用于对系统的静态用例视图进行建模，主要支持系统的行为，即该系统在它的周边环境的语境中所提供的外部可见服务。

**试题答案**

（40）A（41）C（42）B

# **试题32(2015年下半年试题43)**

下所示为UML（  ）。



（43）A．类图  
B.部署图  
C.组件图  
D.网络图

**试题分析**

本题考查统一建模语言（UML）的基本知识。  
UML中提供了多种建模系统的图，体现系统的静态方面和动态方面。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。部署图（deployment diagram）是用来对面向对象系统的物理方面建模的方法，展现了运行时处理结点以及其中构件（制品）的配置。部署图对系统的静态部署视图进行建模，它与组件图（构件图）相关。组件图或构件图（component diagram）展现了一组构件之间的组织和依赖，如题中的图所示。组件图或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。UML部署图经常被认为是一个网络图。

**试题答案**

（43）C

# **试题33(2015年下半年试题44)**

以下关于Singleton（单例）设计模式的叙述中，不正确的是（  ）。  
（44）A．单例模式是创建型模式   
B.单例模式保证一个类仅有一个实例   
C.单例类提供一个访问唯一实例的全局访问点   
D.单例类提供一个创建一系列相关或相互依赖对象的接口

**试题分析**

抽象工厂模式提供一个创建一系列相关或相互依赖对象的接口，而无需指定他们具体的类，而非单例模式。

**试题答案**

（44）D

# **试题34(2015年下半年试题45-47)**

（  ）设计模式能够动态地给一个对象添加一些额外的职责而无需修改此对象的结构；（  ）设计模式定义一个用于创建对象的接口，让子类决定实例化哪一个类；欲使一个后端数据模型能够被多个前端用户界面连接，采用（  ）模式最适合。

（45）A．组合（Composite）   
B.外观(Facade)   
C.享元（Flyweight）   
D.装饰器(Decorator)   
  
（46）A．工厂方法（Factory Method）   
B.享元（Flyweight)   
C.观察者( Observer)   
D.中介者(Mediator)   
  
（47）A．装饰器(Decorator)   
B.享元（Flyweight)   
C.观察者( Observer)   
D.中介者(Mediator)

**试题分析**

抽象工厂模式（Abstract Factory）:提供一个接口，可以创建一系列相关或相互依赖的对象，而无需指定它们具体的类。  
构建器模式（Builder）：将一个复杂类的表示与其构造相分离，使得相同的构建过程能够得出不同的表示。  
工厂方法模式（Factory Method）：定义一个创建对象的接口，但由子类决定需要实例化哪一个类。工厂方法使得子类实例化的过程推迟。  
原型模式（Prototype）：用原型实例指定创建对象的类型，并且通过拷贝这个原型来创建新的对象。  
单例模式（Singleton）:保证一个类只有一个实例，并提供一个访问它的全局访问点。  
适配器模式（Adapter）：将一个类的接口转换成用户希望得到的另一种接口。它使原本不相容的接口得以协同工作。  
桥接模式（Bridge）：将类的抽象部分和它的实现部分分离开来，使它们可以独立地变化。  
组合模式（Composite）：将对象组合成树型结构以表示“整体-部分”的层次结构，使得用户对单个对象和组合对象的使用具有一致性。  
装饰模式（Decorator）：动态地给一个对象添加一些额外的职责。它提供了用子类扩展功能的一个灵活的替代，比派生一个子类更加灵活。  
外观模式（Facade）:定义一个高层接口，为子系统中的一组接口提供一个一致的外观，从而简化了该子系统的使用。  
享元模式（Flyweight）：提供支持大量细粒度对象共享的有效方法。  
代理模式（Proxy）：为其他对象提供一种代理以控制这个对象的访问。  
职责链模式（Chain of Responsibility）：通过给多个对象处理请求的机会，减少请求的发送者与接收者之间的耦合。将接收对象链接起来，在链中传递请求，直到有一个对象处理这个请求。  
命令模式（Command）：将一个请求封装为一个对象，从而可用不同的请求对客户进行参数化，将请求排队或记录请求日志，支持可撤销的操作。  
解释器模式（Interpreter）：给定一种语言，定义它的文法表示，并定义一个解释器，该解释器用来根据文法表示来解释语言中的句子。  
迭代器模式（Iterator）：提供一种方法来顺序访问一个聚合对象中的各个元素，而不需要暴露该对象的内部表示。  
中介者模式（Mediator）：用一个中介对象来封装一系列的对象交互。它使各对象不需要显式地相互调用，从而达到低耦合，还可以独立地改变对象间的交互。  
备忘录模式（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，从而可以在以后将该对象恢复到原先保存的状态。  
观察者模式（Observer）：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自动更新。  
状态模式（State）：允许一个对象在其内部状态改变时改变它的行为。  
策略模式（Strategy）：定义一系列算法，把它们一个个封装起来，并且使它们之间可互相替换，从而让算法可以独立于使用它的用户而变化。  
模板方法模式（Template Method）：定义一个操作中的算法骨架，而将一些步骤延迟到子类中，使得子类可以不改变一个算法的结构即可重新定义算法的某些特定步骤。  
访问者模式（Visitor）：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

**试题答案**

（45）D（46）A（47）D

# **试题35(2015年上半年试题37)**

对象、类、继承和消息传递是面向对象的4个核心概念。其中对象是封装（  ）的整体。  
（37）A．命名空间  
B.要完成任务  
C.一组数据  
D.数据和行为

**试题分析**

本题考查面向对象的基本知识。  
面向对象的4个核心概念是对象、类、继承和消息传递。其中，对象是基本的运行时的实体，它既包括数据（属性），也包括作用于数据的操作（行为）。所以，一个对象把属性和行为封装为一个整体。类定义了一组大体上相似的对象。一个类所包含的方法和数据描述一组对象的共同行为和属性。在进行类设计时，有些类之间存在一般和特殊关系，即一些类是某个类的特殊情况，某个类是一些类的一般情况，这就是继承关系。消息是对象之间进行通信的一种构造，包含要求接收对象去执行某些活动的信息。

**试题答案**

（37）D

# **试题36(2015年上半年试题38)**

面向对象（  ）选择合适的面向对象程序设计语言，将程序组织为相互协作的对象集合，每个对象表示某个类的实例，类通过继承等关系进行组织。  
（38）A．分析   
B.设计   
C.程序设计   
D.测试

**试题分析**

本题考查面向对象的基本知识。  
在采用面向对象技术开发系统时，主要步骤有面向对象分析、面向对象设计、面向对象程序设计和面向对象测试。面向对象分析主要包括：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。面向对象设计是设计分析模型和实现相应源代码。面向对象程序设计选择合适的面向对象程序设计语言，将程序组织为相互协作的对象集合，每个对象表示某个类的实例，类通过继承等关系进行组织。面向对象测试是尽可能早的开始进行系统测试，以发现系统中可能存在的错误并进行修复，进而保证系统质量。

**试题答案**

（38）C

# **试题37(2015年上半年试题39)**

一个类可以具有多个同名而参数类型列表不同的方法，被称为方法（  ）。  
（39）A．重载   
B.调用   
C.重置   
D.标记

**试题分析**

重载，简单说，就是函数或者方法有同样的名称，但是参数列表不相同的情形，这样的同名不同参数的函数或者方法之间，互相称之为重载函数或者方法。

**试题答案**

（39）A

# **试题38(2015年上半年试题40-41)**

UML中有4种关系：依赖、关联、泛化和实现。（  ）是一种结构关系，描述了一组链，链是对象之间的连接；（  ）是一种特殊／一般关系，使子元素共享其父元素的结构和行为。  
（40）A．依赖   
B.关联   
C.泛化   
D.实现   
  
（41）A．依赖   
B.关联   
C.泛化   
D.实现

**试题分析**

UML 用关系把事物结合在一起，主要有下列四种关系：

（1）依赖（dependency）。依赖是两个事物之间的语义关系，其中一个事物发生变化会影响另一个事物的语义。

（2）关联（association）。关联描述一组对象之间连接的结构关系。

（3）泛化（generalization）。泛化是一般化和特殊化的关系，描述特殊元素的对象可替换一般元素的对象。

（4）实现（realization）。实现是类之间的语义关系，其中的一个类指定了由另一个类保证执行的契约。

**试题答案**

（40）B（41）C

# **试题39(2015年上半年试题42-43)**

UML图中，对新开发系统的需求进行建模，规划开发什么功能或测试用例，采用（  ）最适合。而展示交付系统的软件组件和硬件之间的关系的图是（  ）。

（42）A．类图  
B.对象图  
C.用例图  
D.交互图  
  
（43）A．类图  
B.部署图  
C.组件图  
D.网络图

**试题分析**

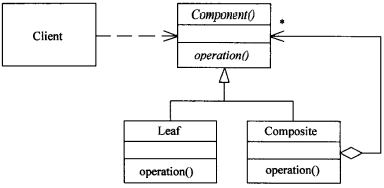
本题考查统一建模语言（UML）的基本知识。  
UML中提供了多种建模系统需求的图，体现系统的静态方面和动态方面。  
类图（Class Diadram）展现了一组对象、接口、协作和它们之间的关系。在面向对象系统的建模中，最常见的就是类图，它给出系统的静态设计视图。对象图（Object Diagram）展现了某一时刻一组对象以及他们之间的关系。对象图描述了在类图中所建立的事物的实例的静态快照，给出系统的静态设计视图或静态进程视图。用例图（Use Case Diagram）展现了一组用例、参与者（Actor）以及它们之间的关系。这个视图主要支持系统的行为，即该系统在它的周边环境的语境中所提供的外部可见服务。用例图用于对一个系统的需求进行建模，包括说明这个系统应该做什么（从系统外部的一个视点出发），而不考虑系统应该怎样做。交互图用于对系统的动态方面进行建模。一张交互图表现的是一个交互，由一组对象和它们之间的关系组成，包含它们之间可能传递的消息。交互图表现为序列图、通信图、交互概览图和时序图，每种针对不同的目的，能适用于不同的情况。序列图是强调消息时间顺序的交互图；通信图是强调接收和发送消息的对象的结构组织的交互图；交互概览图强调控制流的交互图。时序图（Timing Diagram）关注沿着线性时间轴、生命线内部和生命线之间的条件改变。  
部署图（Deploy Diagram）是用来对面向对象系统的物理方面建模的方法，展现了运行时处理结点以及其中构件（制品）的配置。组件图（Component Diagram）展现了一组组件之间的组织和依赖。

**试题答案**

（42）C（43）B

# **试题40(2015年上半年试题44-46)**

下图所示为（  ）设计模式，属于（  ）设计模式，适用于（  ）。



（44）A．代理（Proxy）  
B.生成器《Builder)  
C.组合(C amposite)  
D.观察者(Observer)  
  
（45）A．创建型  
B.结构型  
C.行为  
D.结构型和行为  
  
（46）A．表示对象的部分一整体层次结构时  
B.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时  
C.当创建复杂对象的算法应该独立于该对象的组成部分及其装配方式时  
D.在需要比较通用和复杂的对象指针代替简单的指针时

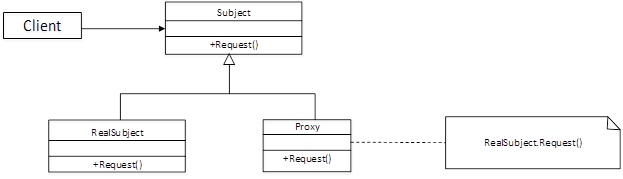
**试题分析**

本题考查设计模式的基本概念。

 每一个设计模式都集中于一个特定的面向对象设计问题或设计要点，描述了什么时候使用它，在另一些设计约束条件下是否还能使用，以及使用的效果和如何取舍。按照设计模式的目的可以分为创建型、结构型和行为型三大类。创建型模式与对象的创建有关；结构型模式处理类或对象的组合；行为型模式对类或对象怎样交互和怎样分配职责进行描述。每种设计模式都有其适应性，描述适用于解决的问题场合。

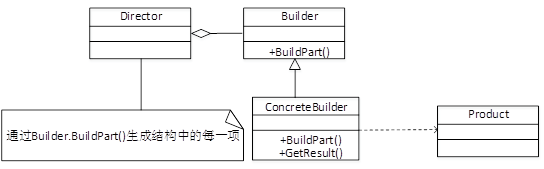
创建型模式包括 Factory Method 、Abstract Factory、 Builder 、Portotype 和 Singleton 。 结构型模式包括 Adapter (类)、 Adapter (对象)、 Bridge 、Composite 、Decorator 、Facade、 Flyweight 和 Proxy 。行为型模式包括Intepreter 、Template Method 、Chain of Responsibility 、 Command 、Iterator 、Mediator 、Memento Observer State Strategy 和Vìsitor。

Proxy 模式的结构图如下所示：



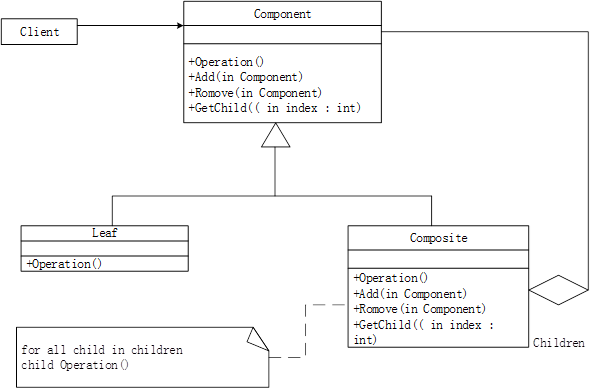
Proxy 模式适用于在需要比较通用和复杂的对象指针代替简单的指针的时候，常见 情况有：远程代理  (Remote   Proxy)   为一个对象在不同地址空间提供据不代表；虚代理 (Virtual Proxy)  根据需要创建开销很大的对象；保护代理 (Protection  Proxy)  控制对原 始对象的访问，用于对象应该有不同的访问权限的时候；智能指引  (Smart  Reference) 取代了简单的指针，它在访问对象时执行一些附加操作。

Builder 模式的结构图如下所示：



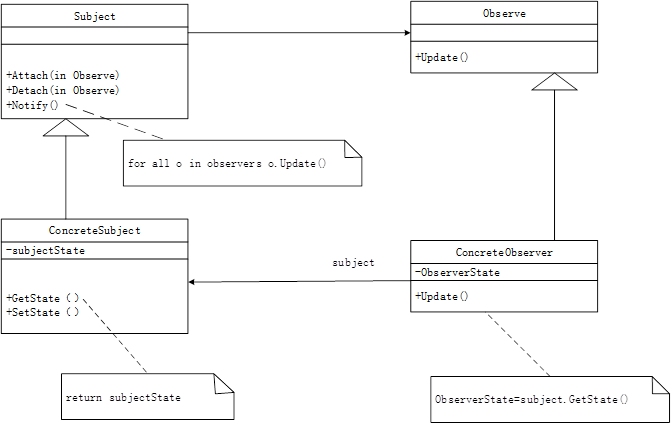
Builder  模式适用于当创建复杂对象的算法应该独立于该对象的组成部分以及它们的装配方式时；当构造过程必须允许被构造的对象有不同的表示时。

Composite 模式的结构图如下所示：



Composite 模式适用于：想表示对象的部分-整体层次结构；希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所有对象。

Observer 模式的结构图如下所示：



Observer 模式适用于：当一个抽象模型有两个方面，其中一个方面依赖于另一个方 面。将这两者封装在独立地对象中以使它们可以各自独立地改变和复用；当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象有待改变时；当一个对象必须通知其他对象，而它又不能假定其他对象是谁，即不希望这些对象是紧耦合的。

**试题答案**

（44）C（45）B（46）A

# **试题41(2015年上半年试题47)**

某些设计模式会引入总是被用作参数的对象（  ）对象是一个多态 accept方法的参数。

（47）A．Visitor  
B.Command  
C.Memento  
D.Observer

**试题分析**

本题考查设计模式的概念，对行为模式进行比较。  
很多行为模式注重封装变化。当一个程序的某个方面的特征经常发生改变时，这些模式就定义一个封装这个方面的对象。这样，当该程序的其他部分依赖于这个方面时，它们都可以与此对象协作。一些模式引入总是被用作参数的对象。有些模式定义一些可作为令牌进行传递的对象，这些对象将在稍后被调用。  
在Visitor模式中，一个Visitor对象是一个多态的accept操作的参数，这个操作作用于该Visitor对象访问的对象。在Command模式中，令牌代表一个请求；在Memento模式中，它代表在一个对象在某个特定时刻的内部状态。在这两种情况下，令牌都可以有一个复杂的内部表示，但客户并不会意识到这一点。在Observer模式中，通过引入Observer和Subject对象来分布通信。

**试题答案**

（47）A

# **试题42(2014年下半年试题37)**

类（  ）之间存在着一般和特殊的关系。  
（37）A．汽车与轮船  
B.交通工具与飞机  
C.轮船与飞机  
D.汽车与飞机

**试题分析**

飞机属于交通工具的一种。

**试题答案**

（37）B

# **试题43(2014年下半年试题38)**

多态分为参数多态、包含多态、过载多态和强制多态四种不同形式，其中（  ）多态在许多语言中都存在，最常见的例子就是子类型化。  
（38）A．参数  
B.包含  
C.过载  
D.强制

**试题分析**

多态分为两种：通用的多态和特定的多态。两者的区别是前者对工作的类型不加限制，允许对不同类型的值执行相同的代码；后者只对有限数量的类型有效，而且对不同类型的值可能要执行不同的代码。

通用的多态又分为参数多态（parametric）和包含多态（inclusion）；特定的多态分为过载多态（overloading）和强制多态（coercion）。

强制多态：编译程序通过语义操作，把操作对象的类型强行加以变换，以符合函数或操作符的要求。程序设计语言中基本类型的大多数操作符，在发生不同类型的数据进行混合运算时，编译程序一般都会进行强制多态。程序员也可以显示地进行强制多态的操作(Casting)。举个例子，比如，int+double，编译系统一般会把int转换为double，然后执行double+double运算，这个int-》double的转换，就实现了强制多态，即可是隐式的，也可显式转换。

过载（overloading）多态：同一个名（操作符﹑函数名）在不同的上下文中有不同的类型。程序设计语言中基本类型的大多数操作符都是过载多态的。通俗的讲法，就是c++中的函数重载。在此处中“overload”译为“过载”，其实就是所谓的“重载”，也许“overload”就应翻译为“过载，重载”吧，那“override”就只能是“覆盖”了。

参数多态：采用参数化模板，通过给出不同的类型参数，使得一个结构有多种类型。

包含多态：同样的操作可用于一个类型及其子类型。（注意是子类型，不是子类。）包含多态一般需要进行运行时的类型检查。

**试题答案**

（38）B

# **试题44(2014年下半年试题39-40)**

在面向对象程序设计语言中，对象之间通过（  ）方式进行通信。以下关于好的面向对象程序设计语言的叙述中，不正确的是（  ）。  
（39）A．消息传递  
B.继承  
C.引用  
D.多态  
  
（40）A．应该支持被封装的对象  
B.应该支持类写实例的概念  
C.应该支持通过指针进行引用  
D.应该支持继承和多态

**试题分析**

对象之间通信通过消息传递来进行，面向对象程序设计语言不通过指针进行。

**试题答案**

（39）A（40）C

# **试题45(2014年下半年试题41-42)**

UML中有4种事物：结构事物、行为事物、分组事物和注释事物。类、接口、构建属于（  ）事物；依附于一个元素或一组元素之上对其进行约束或解释的简单符号为（  ）事物。  
（41）A．结构  
B.行为  
C.分组  
D.注释  
  
（42）A．结构  
B.行为  
C.分组  
D.注释

**试题分析**

事物：模型中的基本成员。UML中包括结构事物、行为事物、分组事物和注释事物。

（1）结构事物：

     模型中静态部分。

    【类Class】+【接口Interface】+【协作Collaboration】

    +【用例Use Case】+【活动类】+【组件Commponent】

    +【结点Node】

（2）行为事物：

     模型中的动态部分。

     【交互】+【状态机】

（3）分组事物

    可以把分组事物看成是一个"盒子"，模型可以在其中被分解。目前只有一种分组事物，即包（package）。结构事物、动作事物甚至分组事物都有可能放在一个包中。包纯粹是概念上的，只存在于开发阶段，而组件在运行时存在。

（4）注释事物

    注释事物是UML模型的解释部分。

**试题答案**

（41）A（42）D

# **试题46(2014年下半年试题43-44)**

一组对象以定义良好但是复杂的方式进行通信，产生的相互依赖关系结构混乱且难以理解。采用（  ）模式，用一个中介对象来封装一系列的对象交互，从而使各对象不需要显式地相互引用，使其耦合松散。而且可以独立地改变它们之间的交互。此模式与（  ）模式是相互竞争的模式，主要差别是：前者的中介对象封装了其它对象间的通信，而后者通过引入其它对象来分布通信。  
（43）A．解释器(Interpreter)  
B.策略(Strategy)  
C.中介者(Mediator)  
D.观察者(Observer)  
  
（44）A．解释器(Interpreter)  
B.策略(Strategy)  
C.中介者(Mediator)  
D.观察者(Observer)

**试题分析**

中介者就是一个处于众多对象，并恰当地处理众多对象之间相互之间的联系的角色。中介者模式与观察者模式是相互竞争的模式。

**试题答案**

（43）C（44）D

# **试题47(2014年下半年试题45)**

UML图中，一张交互图显示一个交互。由一组对象及其之间的关系组成，包含它们之间可能传递的消息。（  ）不是交互图。  
（45）A．序列图  
B.对象图  
C.通信图  
D.时序图

**试题分析**

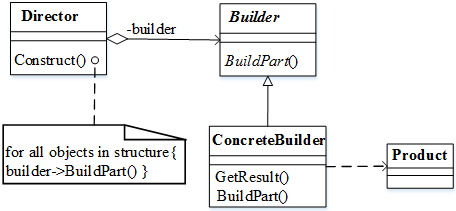
交互图由一组对象和它们之间的关系构成，其中包括：需要什么对象，对象相互之间发送什么消息，什么角色启动消息以及消息按什么顺序发送。交互图的分类：交互图主要分为这两种：顺序图和协作图。对象图不是交互图的一种。

**试题答案**

（45）B

# **试题48(2014年下半年试题46-47)**

图所示为（  ）设计模式，适用于（  ）。



（46）A．抽象工厂(Abstract Factory)  
B.生成器(Builder)  
C.工厂方法(Factory Method)．  
D.原型(Prototype)  
  
（47）A．一个系统要由多个产品系列中的一个来配置时  
B.当一个类希望由它的子类来指定它所创建的对象时  
C.当创建复杂对象的算法应该独立于该对象的组成部分及其装配方式时  
D.当一个系统应该独立于它的产品创建、构成和表示时

**试题分析**

图中所示模式为生成器模式，该模式适用于当创建复杂对象的算法应该独立于该对象的组成部分及其装配方式时。

**试题答案**

（46）B（47）C

# **试题49(2014年上半年试题37)**

（  ）是一个类与它的一个或多个细化类之间的关系，即一般与特殊的关系。

（37）A．泛化   
B.关联   
C.聚集   
D.组合

**试题分析**

泛化(generalization)关系是一个类（称为子类、子接口）继承另外的一个类（称为父类、父接口）的功能，并可以增加它自己的新功能的能力，继承是类与类或者接口与接口之间最常见的关系。  
关联(association)关系: 表示类与类之间的联接, 它使一个类知道另一个类的属性和方法。  
聚合(aggregation)关系: 关联关系的一种特例, 是强的关联关系. 聚合是整体和个体之间的关系,即has-a的关系，此时整体与部分之间是可分离的，他们可以具有各自的生命周期，部分可以属于多个整体对象，也可以为多个整体对象共享。  
组合(合成)关系(composition): 也是关联关系的一种特例，他体现的是一种contains-a的关系，这种关系比聚合更强，也称为强聚合；他同样体现整体与部分间的关系，但此时整体与部分是不可分的，整体的生命周期结束也就意味着部分的生命周期结束。

**试题答案**

（37）A

# **试题50(2014年上半年试题38)**

某些程序设计语言中，在运行过程中当一个对象发送消息请求服务时，根据接收对象的具体情况将请求的操作与实现的方法进行连接，称为（  ）。

（38）A．静态绑定   
B.通用绑定   
C.动态绑定   
D.过载绑定

**试题分析**

静态绑定：绑定的是对象的静态类型，某特性（比如函数）依赖于对象的静态类型，发生在编译期。  
动态绑定：绑定的是对象的动态类型，某特性（比如函数）依赖于对象的动态类型，发生在运行期。

**试题答案**

（38）C

# **试题51(2014年上半年试题39-40)**

在面向对象技术中，不同的对象在收到同一消息时可以产生完全不同的结果， 这一现象称为（  ），’它由（  ）机制来支持。利用类的层次关系，把具有通用功能的消息存放在高层次，而不同的‘实现这一功能的行为放在较低层次，在这些低层次上生成的对象能够给通用消息以不同的响应。

（39）A．绑定   
B.继承   
C.消息   
D.多态   
  
（40）A．绑定   
B.继承   
C.消息   
D.多态

**试题分析**

本题考察面向对象多态的概念  
  
多态实质上是将子类的指针对象或者引用对象传递给父类指针对象后，通过这个父类指针对象调用的函数（此函数在父类中声明为虚函数，且在各个子类中重写这个函数），不是父类中定义的，而是传递进来的子类对象中重写的函数。

**试题答案**

（39）D（40）B

# **试题52(2014年上半年试题41)**

对一个复杂用例中的业务处理流程进行进一步建模的最佳工具是UML（  ）。

（41）A．状态图   
B.顺序图   
C.类图   
D.活动图

**试题分析**

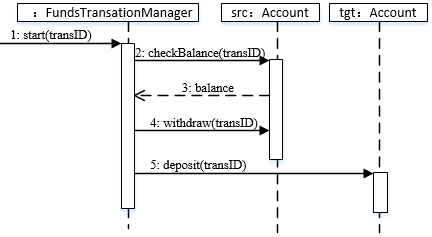
（1）类图（class diagram）。类图描述一组类、接口、协作和它们之间的关系。在OO系统的建模中，最常见的图就是类图。类图给出了系统的静态设计视图，活动类的类图给出了系统的静态进程视图。  
（2）对象图（object diagram）。对象图描述一组对象及它们之间的关系。对象图描述了在类图中所建立的事物实例的静态快照。和类图一样，这些图给出系统的静态设计视图或静态进程视图，但它们是从真实案例或原型案例的角度建立的。  
（3）构件图（component diagram）。构件图描述一个封装的类和它的接口、端口，以及由内嵌的构件和连接件构成的内部结构。构件图用于表示系统的静态设计实现视图。对于由小的部件构建大的系统来说，构件图是很重要的。构件图是类图的变体。  
（4）组合结构图（composite structure diagram）。组合结构图描述结构化类（例如，构件或类）的内部结构，包括结构化类与系统其余部分的交互点。组合结构图用于画出结构化类的内部内容。  
（5）用例图（use case diagram）。用例图描述一组用例、参与者及它们之间的关系。用例图给出系统的静态用例视图。这些图在对系统的行为进行组织和建模时是非常重要的。  
（6）顺序图（sequence diagram，序列图）。顺序图是一种交互图（interaction diagram），交互图展现了一种交互，它由一组对象或参与者以及它们之间可能发送的消息构成。交互图专注于系统的动态视图。顺序图是强调消息的时间次序的交互图。  
（7）通信图（communication diagram）。通信图也是一种交互图，它强调收发消息的对象或参与者的结构组织。顺序图和通信图表达了类似的基本概念，但它们所强调的概念不同，顺序图强调的是时序，通信图强调的是对象之间的组织结构（关系）。在UML 1.X版本中，通信图称为协作图（collaboration diagram）。  
（8）定时图（timing diagram，计时图）。定时图也是一种交互图，它强调消息跨越不同对象或参与者的实际时间，而不仅仅只是关心消息的相对顺序。  
（9）状态图（state diagram）。状态图描述一个状态机，它由状态、转移、事件和活动组成。状态图给出了对象的动态视图。它对于接口、类或协作的行为建模尤为重要，而且它强调事件导致的对象行为，这非常有助于对反应式系统建模。  
（10）活动图（activity diagram）。活动图将进程或其他计算结构展示为计算内部一步步的控制流和数据流。活动图专注于系统的动态视图。它对系统的功能建模和业务流程建模特别重要，并强调对象间的控制流程。  
（11）部署图（deployment diagram）。部署图描述对运行时的处理节点及在其中生存的构件的配置。部署图给出了架构的静态部署视图，通常一个节点包含一个或多个部署图。  
（12）制品图（artifact diagram）。制品图描述计算机中一个系统的物理结构。制品包括文件、数据库和类似的物理比特集合。制品图通常与部署图一起使用。制品也给出了它们实现的类和构件。  
（13）包图（package diagram）。包图描述由模型本身分解而成的组织单元，以及它们之间的依赖关系。  
（14）交互概览图（interaction overview diagram）。交互概览图是活动图和顺序图的混合物。

**试题答案**

（41）D

# **试题53(2014年上半年试题42-43)**

如下所示的序列图中（  ）表示返回消息，Accunt类必须实现的方法有（  ）。



（42）A．tansID   
B.balance   
C.withdraw   
D.deposit   
  
（43）A．start()   
B.checkBalance()和withdraw()   
C.deposit()   
D.checkBalance()、withdraw()和deposit()

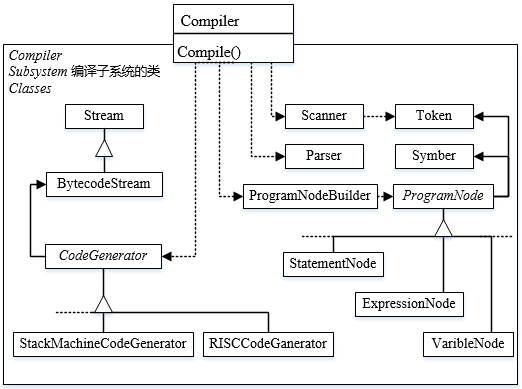
**试题分析**

UML序列图中，返回消息使用虚线带箭头表示，同步消息（调用消息）使用实线带实心箭头表示，异步消息使用实线带箭头表示。  
由图示，FundsTransactionManager会调用Account的checkBalance()、withdraw()和deposit()消息。

**试题答案**

（42）B（43）D

# **试题54(2014年上半年试题44-45)**

下图所示为（  ）设计模式，适用于（  ）。  
  


（44）A．适配器（Adapter）   
B.责任链（Chain of Responsibility）   
C.外观（Facade）   
D.桥接（Bridge）   
  
（45）A．有多个对象可以处理一个请求，在运行时刻自动确定由哪个对象处理   
B.想使用一个已经存在的类，而其接口不符合要求   
C.类的抽象和其实现之间不希望有一个固定的绑定关系   
D.需要为一个复杂子系统提供一个简单接口

**试题分析**

外观模式是一种使用频率非常高的结构型设计模式，它通过引入一个外观角色来简化客户端与子系统之间的交互，为复杂的子系统调用提供一个统一的入口，降低子系统与客户端的耦合度，且客户端调用非常方便。

**试题答案**

（44）C（45）D

# **试题55(2014年上半年试题46-47)**

下列设计模式中，（  ）模式既是类结构型樽式，又是对象结构型模式。此模式与（  ）模式类似的特征是，都给另一个对象提供了一定程度上的间接性，都涉及到从自身以外的一个接口向这个对象转发请求。

（46）A．桥接（Bridge）   
B.适配器（Adapter）   
C.组成(Composite)   
D.装饰器 (Decorator)   
  
（47）A．桥接（Bridge）   
B.适配器（Adapter）   
C.组成(Composite)   
D.装饰器 (Decorator)

**试题分析**

适配器（adapter）模式。适配器模式将一个接口转换成客户希望的另一个接口，从而使接口不兼容的那些类可以一起工作。适配器模式既可以作为类结构型模式，也可以作为对象结构型模式。在类适配器模式中，通过使用一个具体类将适配者适配到目标接口中；在对象适配器模式中，一个适配器可以将多个不同的适配者适配到同一个目标。  
桥接（bridge）模式。桥接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。它是一种对象结构型模式，又称为柄体（handle and body）模式或接口（interface）模式。桥接模式类似于多重继承方案，但是多重继承方案往往违背了类的单一职责原则，其复用性比较差，桥接模式是比多重继承方案更好的解决方法。  
组合（composite）模式。组合模式又称为整体-部分（part-whole）模式，属于对象的结构模式。在组合模式中，通过组合多个对象形成树形结构以表示整体-部分的结构层次。组合模式对单个对象（即叶子对象）和组合对象（即容器对象）的使用具有一致性。  
装饰（decorator）模式。装饰模式是一种对象结构型模式，可动态地给一个对象增加一些额外的职责，就增加对象功能来说，装饰模式比生成子类实现更为灵活。通过装饰模式，可以在不影响其他对象的情况下，以动态、透明的方式给单个对象添加职责；当需要动态地给一个对象增加功能，这些功能可以再动态地被撤销时可使用装饰模式；当不能采用生成子类的方法进行扩充时也可使用装饰模式。

**试题答案**

（46）B（47）A

# **试题56(2013年下半年试题37)**

在领域类模型中不包含（  ）。

（37）A．属性   
B.操作   
C.关联   
D.领域对象

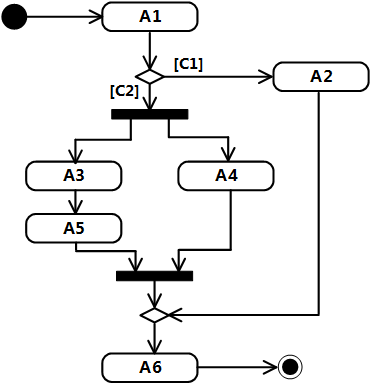
**试题分析**

在领域类模型中会涉及描述类自身情况的属性与操作，还会有描述类之关系的关联，但不会有对象层次的内容。

**试题答案**

（37）D

# **试题57(2013年下半年试题38)**

在执行如下所示的UML活动图时，能同时运行的最大线程数为（  ）。  


（38）A．4   
B.3   
C.2   
D.1

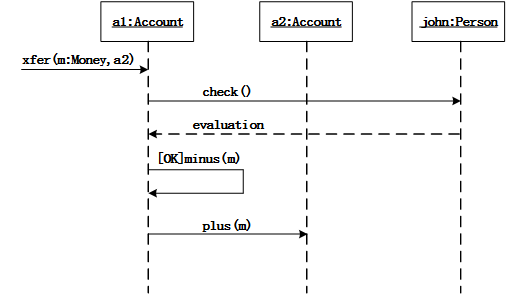
**试题分析**

该活动图能同时运行的最大线程数为2。

**试题答案**

（38）C

# **试题58(2013年下半年试题39-40)**

下图所示的UML序列图中，（  ）表示返回消息，Account应该实现的方法有（  ）。  


（39）A．xfer   
B.check   
C.evaluation   
D.minus   
  
（40）A．xfer()   
B.xfer()、plus()和minus()   
C.check()、plus()和minus()   
D.xfer()、evaluation()、plus()和minus()

**试题分析**

序列图主要用于按照交互发生的一系列顺序，显示对象之间的这些交互。以本题为例，第1个消息——xfer（m:Money,a2）表示外部有对象要调取a1这个对象的xfer()操作。而evaluation就是在a1调用john的check()操作之后的一个消息反馈。

在整个图中, evaluation是一个消息反馈，不需要由Account来实现，而check()操作应由Person实现，而非Account。所以Account需要实现的包括：xfer()、plus()和minus()。

**试题答案**

（39）C（40）B

# **试题59(2013年下半年试题41-43)**

在面向对象技术中，（  ）定义了超类和子类之间的关系，子类中以更具体的方式实现从父类继承来的方法称为（  ），不同类的对象通过（  ）相互通信。

（41）A．覆盖   
B.继承   
C.消息   
D.多态   
  
（42）A．覆盖   
B.继承   
C.消息   
D.多态   
  
（43）A．覆盖   
B.继承   
C.消息   
D.多态

**试题分析**

继承：父类和子类之间共享数据的方法的机制。这是类之间的一种关系，在定义和实现一个类的时候，可以在一个已经存在的类的基础上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

消息：对象之间进行通信的一种构造叫做消息。当一个消息发送给某个对象时，包含要求接收对象去执行某些活动的信息。接收到信息的对象经过解释，然后予以响应。这种通信机制叫做消息传递。发送消息的对象不需要知道接收消息的对象如何对请求予以响应。

多态：在收到消息时，对象要予以响应。不同的对象收到同一消息可以产生完全不同的结果，这一现象叫做多态。

覆盖：如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的覆盖（override），也称为方法的重载。

**试题答案**

（41）B（42）A（43）C

# **试题60(2013年下半年试题44)**

（  ）设计模式定义一系列算法，把它们一个个封装起来，并且使它们可相互替换。这一模式使得算法可独立于它的客户而变化。

（44）A．策略（Strategy）   
B.抽象工厂（Abstract Factory）   
C.观察者（Observer）   
D.状态（State）

**试题分析**

观察者模式（有时又被称为发布-订阅Subscribe>模式、模型-视图View>模式、源-收听者Listener>模式或从属者模式）是软件设计模式的一种。在此种模式中，一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实作事件处理系统。

单例模式是一种常用的软件设计模式。在它的核心结构中只包含一个被称为单例类的特殊类。通过单例模式可以保证系统中一个类只有一个实例而且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。如果希望在系统中某个类的对象只能存在一个，单例模式是最好的解决方案。

适配器模式（有时候也称包装样式或者包装）将一个类的接口适配成用户所期待的。一个适配允许通常因为接口不兼容而不能在一起工作的类工作在一起，做法是将类自己的接口包裹在一个已存在的类中。

装饰者模式。装饰模式是在不必改变原类文件和使用继承的情况下，动态的扩展一个对象的功能。它是通过创建一个包装对象，也就是装饰来包裹真实的对象。

**试题答案**

（44）A

# **试题61(2013年下半年试题45)**

在发布-订阅（Publish-Subscibe）消息模式中，订阅者订阅一个主题后，当该主题有新消息到达时，所有订阅者都会收到通知。（  ）设计模式最适合之一模式。

（45）A．适配器（Adapter）   
B.通知（Notifier）   
C.状态（State）   
D.观察者（Observer）

**试题分析**

观察者模式（有时又被称为发布-订阅Subscribe>模式、模型-视图View>模式、源-收听者Listener>模式或从属者模式）是软件设计模式的一种。在此种模式中，一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实作事件处理系统。

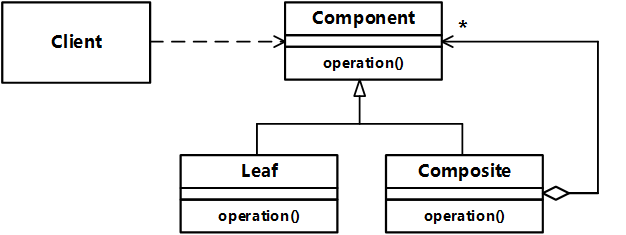
适配器模式（有时候也称包装样式或者包装）将一个类的接口适配成用户所期待的。一个适配允许通常因为接口不兼容而不能在一起工作的类工作在一起，做法是将类自己的接口包裹在一个已存在的类中。

状态模式：当一个对象的内在状态改变时允许改变其行为，这个对象看起来像是改变了其类。

**试题答案**

（45）D

# **试题62(2013年下半年试题46-47)**

下图所示为（  ）设计模式，适用于（  ）。  


（46）A．组件（Component）   
B.适配器（Adapter）   
C.组合（Composite）   
D.装饰器（Decorator）   
  
（47）A．表示对象的部分-整体层次结构   
B.不希望在抽象和它的实现部分之间有一个固定的绑定关系   
C.在不影响其他对象的情况下，以动态、透明的方式给单个对象添加职责   
D.使所有接口不兼容类可以一起工作

**试题分析**

本题中的图，即为组合模式的标准UML图，所以应对软件设计师考试，要求掌握常见设计模式的UML图示。

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

**试题答案**

（46）C（47）A

# **试题63(2013年上半年试题37)**

在多态的几种不同形式中，（  ）多态是一种特定的多态，指同一个名字在不同上下文中可代表不同的含义。

（37）A．参数   
B.包含   
C.过载   
D.强制

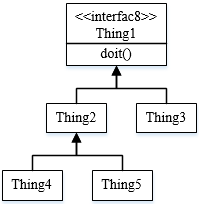
**试题分析**

一般将多台分为通用多态和特殊多态。通用多态包括参数多态和包含多态。  
参数多态采用参数化模板，通过给出不同的类型参数，使得一个结构有多种类型。  
包含多态同样的操作可用于一个类型及其子类型。（注意是子类型，不是子类。）包含多态一般需要进行运行时的类型检查。如Pascal中的子界。  
特殊多态包括强制多态和过载多态。  
强制多态编译程序通过语义操作，把操作对象的类型强行加以变换，以符合函数或操作符的要求。程序设计语言中基本类型的大多数操作符，在发生不同类型的数据进行混合运算时，编译程序一般都会进行强制多态  
过载多态是一种特定的多态，指同一个名（操作符、函数名）在不同上下文中可代表不同的含义。

**试题答案**

（37）C

# **试题64(2013年上半年试题38-39)**

继承是父类和子类之间共享数据和方法的机制。以下关于继承的叙述中，不正确的是（  ）。有关于图中doIt()方法的叙述中，正确的是（  ）。  


（38）A．一个父类可以有多个子类，这些子类都是父类的特例   
B.父类描述了这些子类的公共属性和操作   
C.子类可以继承它的父类（或祖先类）中的属性和操作而不必自己定义   
D.子类中可以定义自己的新操作而不能定义和父类同名的操作   
  
（39）A．doIt()必须由Thing3实现，同时可能由Thing4实现   
B.doIt()必须由Thing5实现   
C.doIt()必须由Thing2、Thing3、Thing4、Thing5实现   
D.doIt()已经由Thing1实现，因为无需其它类实现

**试题分析**

 继承，就是在已经存在的类的基础上，再进行扩展，从而产生新的类。已经存在的类称为父类、超类或基类，而新产生的类称为子类或派生类。一个父类可以有多个子类，这些子类都是父类的特例，父类描述了这些子类的公共属性和操作。子类可以继承它的父类（或祖先类）中的属性和操作而不必自己定义。子类可以继承父类（或祖先类）的属性和操作这些在子类中不必定义，子类中可以定义自己的属性和操作  
题中的Thing1为接口，doIt（）为接口中的抽象方法，必须由实现它的类去实现该方法。因此在Thing3中必须实现，而Thing4也是Thing1的子类，但不是直接子类，所以可能有Thing4实现。

**试题答案**

（38）D（39）A

# **试题65(2013年上半年试题40)**

以下关于UML部署图的叙述中，正确的是（  ）。

（40）A．因为一条消息总是有某种响应，所以部署组件之间的依赖是双向的   
B.部署组件之间的依赖关系类似于包依赖   
C.部署图不用于描述代码的物理模块   
D.部署图不用于描述系统在不同计算机系统的物理分布

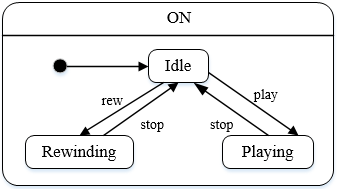
**试题分析**

部署图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。它与构建视图相关，通产一个结点包含一个或多个构件。

**试题答案**

（40）B

# **试题66(2013年上半年试题41-42)**

以下关于UML状态图的叙述中，不正确的是（  ）。对下图的描述，正确的是（  ）。  


（41）A．用于描述一个对象在多个用例中的行为   
B.用于某些具有多个状态的对象而不是系统中大多数或全部对象   
C.用于描述多个对象之间的交互   
D.可以用于用户界面或控制对象   
  
（42）A．ON是一个并发状态   
B.因为此状态图中没有终点(final)状态，所以此图是无效的   
C.play stop 和rew 是动作   
D.ON是超状态

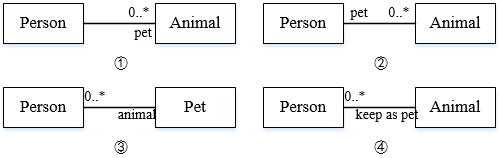
**试题分析**

状态图展现一个状态机，它由状态、转换、事件和活动组成。可以用状态图对系统的动态方面建模。这些动态方面可以包括现在系统体系结构的任何视图中的任何一种对象的按事件排序的行为，这些对象包括类（主动类）、借口、构建和结点。所以状态不表示多个对象之间的交互。根据ON状态的内部行为可以发现状态为超状态。

**试题答案**

（41）C（42）D

# **试题67(2013年上半年试题43)**

描述一些人（Person）将动物（Animal）养为宠物(Pet)的是图（  ）。  


（43）A．①   
B.②   
C.③   
D.④

**试题分析**

该题考查UML中类图的表示，由Person类和Animal类之间1:N 的关系以及类图的画法可知选项为A。

**试题答案**

（43）A

# **试题68(2013年上半年试题44-47)**

（  ）设计模式能使一个对象的状态发生改变时通知所有依赖它的监听者。（  ）设计模式限制类的实例对象只能有一个。适配器（Adapter）设计模式可以用于（  ）。用于为一个对象添加更多功能而不使用子类的是（  ）设计模式

（44）A．责任链   
B.命令   
C.抽象工厂   
D.观察者   
  
（45）A．原型   
B.工厂方法   
C.单例   
D.生成器   
  
（46）A．将已有类的接口转换成和目标接口兼容   
B.改进系统性能   
C.将客户端代码数据转换成目标接口期望的合适的格式   
D.使所有接口不兼容可以一起工作   
  
（47）A．桥接   
B.适配器   
C.组合   
D.装饰器

**试题分析**

观察者模式（有时又被称为发布-订阅Subscribe>模式、模型-视图View>模式、源-收听者Listener>模式或从属者模式）是软件设计模式的一种。在此种模式中，一个目标物件管理所有相依于它的观察者物件，并且在它本身的状态改变时主动发出通知。这通常透过呼叫各观察者所提供的方法来实现。此种模式通常被用来实作事件处理系统。  
单例模式是一种常用的软件设计模式。在它的核心结构中只包含一个被称为单例类的特殊类。通过单例模式可以保证系统中一个类只有一个实例而且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。如果希望在系统中某个类的对象只能存在一个，单例模式是最好的解决方案。  
适配器模式（有时候也称包装样式或者包装）将一个类的接口适配成用户所期待的。一个适配允许通常因为接口不兼容而不能在一起工作的类工作在一起，做法是将类自己的接口包裹在一个已存在的类中。  
装饰者模式。装饰模式是在不必改变原类文件和使用继承的情况下，动态的扩展一个对象的功能。它是通过创建一个包装对象，也就是装饰来包裹真实的对象。

**试题答案**

（44）D（45）C（46）A（47）D

# **试题69(2012年下半年试题37)**

在面向对象技术中，对象具有以下特性：（  ）。  
①清晰的边界  ②良好定义的行为  ③确定的位置和数量 ④可扩展性

（37）A．②④   
B.①②③④   
C.①②④   
D.①②

**试题分析**

在面向对象技术中，对象是类的一个实例，它的特性主要包括清晰的边界、良好定义的行为和可扩展性。

**试题答案**

（37）C

# **试题70(2012年下半年试题38-39)**

在面向对象技术中，（  ）说明一个对象具有多种形态，（  ）定义超类与子类之间的关系。

（38）A．继承   
B.组合   
C.封装   
D.多态   
  
（39）A．继承   
B.组合   
C.封装   
D.多态

**试题分析**

继承描述的是父类与子类之间的关系，表示特殊与一般；组合描述的是整体和部分的关系，且整体和部分的生命周期一致；封装描述的是对象的隐蔽性，只需对外公开接口即可；多态描述的是一个对象或方法的多种表现形式。

**试题答案**

（38）D（39）A

# **试题71(2012年下半年试题40)**

如果要表示待开发软件系统中软件组件和硬件之间的物理关系，通常采用UML中的（  ）。

（40）A．组件图   
B.部署图   
C.类图   
D.网络图

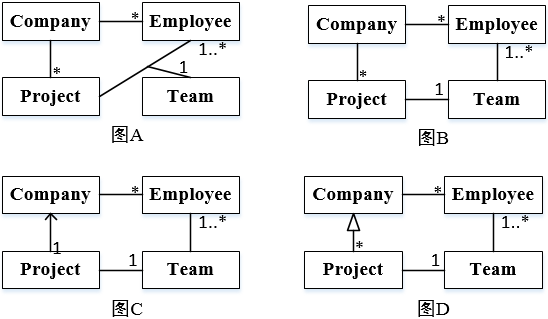
**试题分析**

类图描述一组类、接口、协作和它们之间的关系。在面向对象系统的建模中，最常见的图就是类图。类图给出了系统的静态设计视图，活动类的类图给出了系统的静态进程视图。  
部署图描述对运行时的处理节点及在其中生存的构件的配置。部署图给出了架构的静态部署视图，通常一个节点包含一个或多个部署图。  
而网络图和组件图在UML并没有这种描述。  
而本题中描述的是待开发软件系统中软件组件和硬件之间的物理关系，描述的正是软件在硬件中的部署，是部署图。

**试题答案**

（40）B

# **试题72(2012年下半年试题41)**

对于场景：一个公司负责多个项目，每个项目（Project）由一个员工（Employee）团队（ Team）来开发。下列UML概念图中，（  ）最适合描述这一场景。  
 

（41）A．图A   
B.图B   
C.图C   
D.图D

**试题分析**

本题主要考查类图和多重度的知识。  
在UML中，关联的多重度是指一个类的实例能够与另一个类的多少个实例相关联。它又称为重复度。多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用\*号表示）。最常见的多重性有0..1、0..\*、1和1..\*。  
在本题中，根据题目的描述，我们不难看出有4个类，而且由描述“一个公司负责多个项目”可知，公司和项目两个类之间的关联是一种一对多的关联，即项目端是多端，这样就可以排除答案C，另外公司和项目之间并不是一种继承的关系，这样就可以排除答案D。  
而在A和B选项间，他们的区别就在于A的关联牵涉到了3个类，但其实这种表达方式是不存在的，而B选项中很好地表明了题目的意思，即一个项目由一个团队来开发，而团队中的成员个数至少是一个，或者是多个。

**试题答案**

（41）B

# **试题73(2012年下半年试题42)**

UML中接口可用于（  ）。

（42）A．提供构造型（ stereotype）名称为〈〈terface〉〉的具体类   
B.Java和C++程序设计中，而C#程序设计中不支持   
C.定义可以在多个类中重用的可执行逻辑   
D.声明对象类所需要的服务

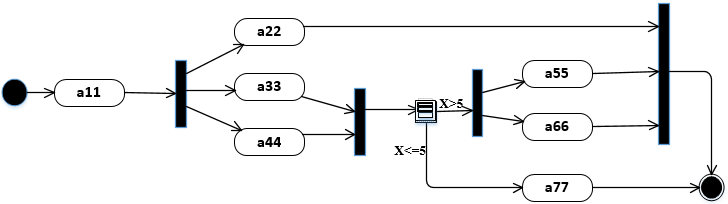
**试题分析**

本题主要考查我们对接口的理解。  
接口是描述某个类或构件的一个服务操作集，在接口中我们可以声明对象类所需要的服务。而接口不仅可以在Java和C++程序设计中，也可以在C#等各种程序设计中使用。

**试题答案**

（42）D

# **试题74(2012年下半年试题43)**

下列活动图中可以同时执行的活动是（  ）。   


（43）A．a44和a66   
B.a22，a33和a44   
C.a11和a77   
D.a66和a77

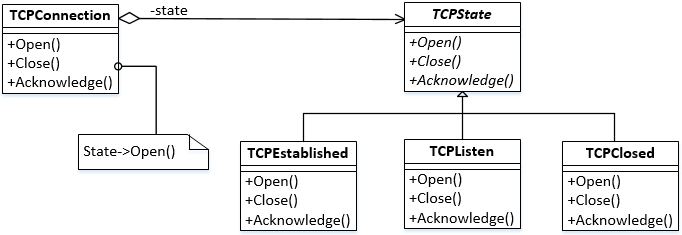
**试题分析**

活动图可以更好地描述出行为的细节，阐明了业务用例实现的工作流程。活动图主要由以下一些元素组成：  
（1）初始节点和活动终点。在活动图中有两个特殊的节点，一个用来表示活动的初始节点，它用一个实心圆表示，在一张不包括子图的活动图中有且只有一个初始节点。而另一个则是表示活动处理完成的活动终点，它用一个圆圈内加一个实心圆来表示，在活动图中可能包含多个活动终点。  
（2）活动状态。是活动图中最主要的元素之一，它用来表示一个活动，如图中的a11等。  
（3）转换。当一个活动结束时，控制流就会马上传递给下一个活动状态，在活动图中称之为转换，用一条带箭头的直线来表示，转移表示各种活动状态的先后顺序。  
（4）分支与监护条件。对于任何一个控制流而言，都一定会存在分支、循环等形式的控制流。在活动图中，分支用一个菱形表示，它有一个进入转换（箭头从外指向分支符号），一个或多个离开转换（箭头从分支符号指向外）。而每个离开转换上都会有一个监护条件，用来表示满足什么条件的时候执行该转换。但要注意，在多个离开转换上的监护条件不能有矛盾，否则就会使得流程产生混乱。  
（5）分岔与汇合。在实际的控制流中，除了顺序结构、分支结构和循环结构之外，还可能存在并发的事件流。在UML中，可以采用一个同步线来说明这些并行控制流的分岔和汇合。  
在本题中，要求出那些活动是可以同时执行的，这就要求他们没有先后关系，显然选项A和C不正确，而对于D选项，a66和a77分别是判定条件连个不同结果下要执行的活动，也不是可以同时执行的活动，因此答案选B。

**试题答案**

（43）B

# **试题75(2012年下半年试题44-46)**

每种设计模式都有特定的意图。（  ）模式使得一个对象在其内部状态改变时通过调用另一个类中的方法改变其行为，使这个对象看起来如同修改了它的类。下图是采用该模式的有关TCP连接的结构图实例。该模式的核心思想是引入抽象类（  ）来表示TCP连接的状态，声明不同操作状态的公共接口，其子类实现与特定状态相关的行为。当一个（  ）对象收到其它对象的请求时，它根据自身的当前状态做出不同的反应。  


（44）A．适配器（ Adapter）   
B.命令（Command）   
C.观察者（ Visitor）   
D.状态（State）   
  
（45）A．TCPConnection   
B.state   
C.TCPState   
D.TCPEstablished   
  
（46）A．TCPConnection   
B.state   
C.TCPState   
D.TCPEstablished

**试题分析**

状态模式的意图是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的类。状态模式把所研究的对象的行为包装在不同的状态对象里，每一个状态对象都属于一个抽象状态类的一个子类，因此该模式的核心思想是引入抽象类TCPState来表示TCP连接的状态，声明不同操作状态的公共接口，其子类实现与特定状态相关的行为。当一个TCPConnection对象收到其它对象的请求时，它根据自身的当前状态做出不同的反应。

**试题答案**

（44）D（45）C（46）A

# **试题76(2012年下半年试题47)**

欲使类A的所有使用者都使用A的同一个实例，应（  ）。

（47）A．将A标识为final   
B.将A标识为abstract   
C.将单例（ Singleton）模式应用于A   
D.将备忘（ Memento）模式应用于A

**试题分析**

本题主要考查单例模式。单例模式的最大特点是一个类只有一个实例，因此采用这种设计模式，就可以实现类A的所有使用者都使用A的同一个实例。

**试题答案**

（47）C

# **试题77(2012年上半年试题37)**

面向对象技术中，组合关系表示（  ）。

（37）A．包与其中模型元素的关系   
B.用例之间的一种关系   
C.类与其对象的关系   
D.整体与其部分之间的一种关系

**试题分析**

在面向对象技术中，组合描述的是整体与部分的关系，组合关系中，整体与部分的生命周期一致。比如公司与部门就是一种组合关系，公司不存在了，部门自然就不存在了。

**试题答案**

（37）D

# **试题78(2012年上半年试题38)**

以下关于封装在软件复用中所充当的角色的叙述，正确的是（  ）。

（38）A．封装使得其他开发人员不需要知道一个软件组件内部如何工作   
B.封装使得软件组件更有效地工作   
C.封装使得软件开发人员不简要编制开发文档   
D.封装使得软件组件开发更加容易

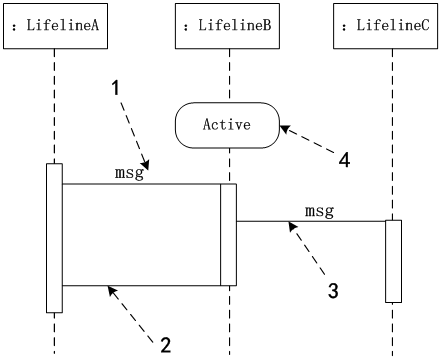
**试题分析**

封装是面向对象技术的三大特点之一，封装的目的是使对象的定义和实现分离，这样，就能减少耦合。封装可以使得其他开发人员不需要知道一个软件组件内部是如何工作的，只需要使用该组件提供的接口来完成交互即可，如果在另外一个地方需要完成同样的功能，我们就可以将该组件使用在另外一个地方，这样提供了软件的复用性。

**试题答案**

（38）A

# **试题79(2012年上半年试题40-43)**

UML序列图是一种交互图，描述了系统中对象之间传递消息的时间次序。其中，异步消息与同步消息不同，（  ）。下图中（  ）表示一条同步消息，（  ）表示一条异步消息，（  ）表示一条返回消息。  


（40）A．异步消息并不引起调用者终止执行而等待控制权的返回   
B.异步消息和阻塞调用有相同的效果   
C.异步消息是同步消息的响应   
D.异步消息和同步消息一样等待返回消息   
  
（41）A．1   
B.2   
C.3   
D.4   
  
（42）A．1   
B.2   
C.3   
D.4   
  
（43）A．1   
B.2   
C.3   
D.4

**试题分析**

在本题中，首先我们要理解同步消息与异步消息的区别。如果一个对象发送了一个同步消息，那么它要等待对方对消息的应答，收到应答后才能继续自己的操作。而发送异步消息的对象不需要等待对方对消息的应答便可以继续自己的操作。

在本题中，1表示的是同步消息，而2表示的是返回消息，3表示的是异步消息。一般情况下，在顺序图中，同步消息、异步消息、返回消息都是用本题图中的箭头表示，请注意它们的区别。

**试题答案**

（40）A（41）A（42）C（43）B

# **试题80(2012年上半年试题44)**

设计模式根据目的进行分类，可以分为创建型、结构型和行为型三种。其中结构型模式用于处理类和对象的组合。（  ）模式是一种结构型模式。

（44）A．适配器（ Adapter）   
B.命令（Command）   
C.生成器（Builder）   
D.状态（State）

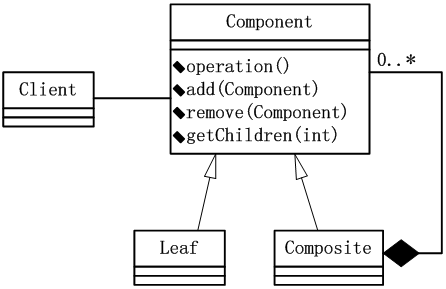
**试题分析**

常见的创建型模式主要有工厂方法（Factory Method）、抽象工厂（Abstract Factory）、单例（Singleton）、构建（Builder）、原型（Prototype）模式；结构型模式有适配器（Adapter）、合成（Composite）、装饰（Decorator）、代理（Proxy）、享元（Flyweight）、门面（Façade）、桥接（Bridge）模式；行为型模式有策略（Strategy）、模板方法（Template Method）、迭代器（Iterator）、责任链（Chain of Responsibility）、命令（Command）、备忘录（Memento）、状态（State）、访问者（Visitor）、解释器（（Interpreter）、调停者（Mediator）、观察者（Observer）模式。

**试题答案**

（44）A

# **试题81(2012年上半年试题45-47)**

设计模式中的（  ）模式将对象组合成树形结构以表示“部分一整体”的层次结构，使得客户对单个对象和组合对象的使用具有一致性。下图为该模式的类图，其中，（  ）定义有子部件的那些部件的行为；组合部件的对象由（  ）通过component提供的接口操作。  


（45）A．代理（Proxy）   
B.桥接器（Bridge）   
C.组合（ Composite）   
D.装饰器（Decorator）   
  
（46）A．Client   
B.Component   
C.Leaf   
D.Composite   
  
（47）A．Client   
B.Component   
C.Leaf   
D.Composite

**试题分析**

本题主要组合设计模式。组合设计模式将对象组合成树形结构以表示“部分一整体”的层次结构，使得客户对单个对象和组合对象的使用具有一致性。  
在类图中，Component为合成的对象声明接口；某些情况下，实现从此接口派生出所有类共有的默认行为，定义一个接口可以访问及管理它的多个部分（GetChild），如果必要也可以在递归结构中定义一个接口访问它的父节点，并且实现它；Leaf在合成中表示叶节点对象，叶节点没有子节点；Composite用来定义有子节点（子部件）的部件的行为，存储子节点（子部件）；Client通过Component接口控制组合部分的对象。

**试题答案**

（45）C（46）D（47）A

# **试题82(2011年下半年试题37)**

采用面向对象开发方法时，对象是系统运行时基本实体。以下关于对象的叙述中，正确的是（  ）。

（37）A．对象只能包括数据（属性）  
B.对象只能包括操作（行为）  
C.对象一定有相同的属性和行为  
D.对象通常由对象名、属性和操作三个部分组成

**试题分析**

对象通常由对象名、属性和操作三个部分组成，对象不一定都有相同的属性和行为。

**试题答案**

（37）D

# **试题83(2011年下半年试题38-39)**

一个类是（  ）。在定义类时，将属性声明为private的目的是（  ）。

（38）A．一组对象的封装  
B.表示一组对象的层次关系  
C.一组对象的实例  
D.一组对象的抽象定义  
  
（39）A．实现数据隐藏，以免意外更改  
B.操作符重载  
C.实现属性值不可更改  
D.实现属性值对类的所有对象共享

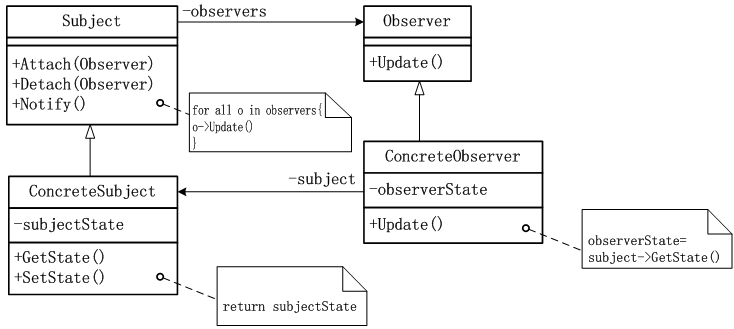
**试题分析**

类与对象的关系是抽象与具体的关系，类是一组对象的抽象，而对象是类的实例。在定义类时，将属性声明为private，即只允许自身对其进行访问、修改等操作，而外界不可见，从而达到实现数据隐藏，以免意外更改的母的。

**试题答案**

（38）D（39）A

# **试题84(2011年下半年试题40-41)**

（  ）设计模式允许一个对象在其状态改变时，通知依赖它的所有对象。该设计模式的类图如下图，其中，（  ）在其状态发生改变时，向它的各个观察者发出通知。  


（40）A．命令( Command)  
B.责任链(Chain of Responsibility)  
C.观察者( Observer)  
D.迭代器(Iterator)  
  
（41）A．Subject  
B.ConcreteSubject  
C.Observer  
D.ConcreteObserver

**试题分析**

观察者模式的意图是：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。  
在观察者模式类图中，Subject是被观察对象，了解其多个观察者，任意数量的观察者可以观察一个对象，提供一个接口用来绑定以及分离观察者对象。  
Concrete Subject是具体被观察对象，存储具体观察者Concrete Observer有兴趣的状态。当其状态改变时，发送一个通知给其所有的观察者对象。  
Observer是观察者，定义一个更新接口，在一个被观察对象改变时应被通知。  
Concrete Observer是具体观察者，维护一个对Concrete Subject对象的引用。

**试题答案**

（40）C（41）B

# **试题85(2011年下半年试题42)**

在面向对象软件开发中，封装是一种（  ）技术，其目的是使对象的使用者和生产者分离。

（42）A．接口管理  
B.信息隐藏  
C.多态  
D.聚合

**试题分析**

面向对象系统中的封装单位是对象，对象之间只能通过接口进行信息交流，对象外部不能对对象中的数据随意地进行访问。封装的目的是使对象的定义和实现分离，这样，就能减少耦合，类内部的实现可以自由改变而不会影响其他的类或对象。

**试题答案**

（42）B

# **试题86(2011年下半年试题43)**

欲动态地给一个对象添加职责，宜采用（  ）模式。

（43）A．适配器(Adapter)  
B.桥接(Bridge)  
C.组合( Composite)  
D.装饰器(Decorator)

**试题分析**

适配器模式将一个接口转换成为客户想要的另一个接口，适配器模式使接口不兼容的那些类可以一起工作。  
桥接模式将抽象部分与实现部分分离，使得它们两部分可以独立地变化。  
合成模式组合多个对象形成树型结构以表示整体—部分的结构层次。  
装饰器模式动态地给一个对象增加其他职责，就增加对象功能来说，装饰模式比生成子类实现更为灵活。

**试题答案**

（43）D

# **试题87(2011年下半年试题44)**

（  ）模式通过提供与对象相同的接口来控制对这个对象的访问。

（44）A．适配器( Adapter)  
B.代理(Proxy)  
C.组合( Composite)  
D.装饰器(Decorator)

**试题分析**

适配器模式将一个接口转换成为客户想要的另一个接口，适配器模式使接口不兼容的那些类可以一起工作。  
代理模式通过提供与对象相同的接口来控制对这个对象的访问。  
组合模式组合多个对象形成树型结构以表示整体—部分的结构层次。  
装饰器模式动态地给一个对象增加其他职责，就增加对象功能来说，装饰模式比生成子类实现更为灵活。

**试题答案**

（44）B

# **试题88(2011年下半年试题45)**

采用UML进行面向对象开发时，部署图通常在（  ）阶段使用。

（45）A．需求分析  
B.架构设计  
C.实现  
D.实施

**试题分析**

部署图描述了一个运行时的硬件结点，以及在这些结点上运行的软件组件的静态视图。 部署图显示了系统的硬件，安装在硬件上的软件，以及用于连接异构的机器之间的中间件。因此它是在实施阶段被使用。

**试题答案**

（45）D

# **试题89(2011年下半年试题46-47)**

业务用例和参与者一起描述（  ），而业务对象模型描述（  ）。

（46）A．工作过程中的静态元素  
B.工作过程中的动态元素  
C.工作过程中的逻辑视图  
D.组织支持的业务过程  
  
（47）A．业务结构  
B.结构元素如何完成业务用例  
C.业务结构以及结构元素如何完成业务用例  
D.组织支持的业务过程

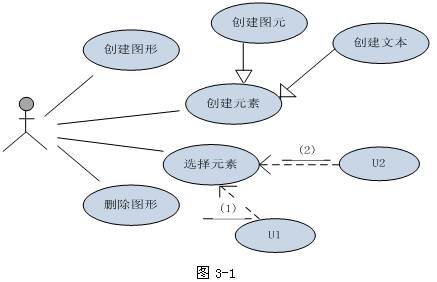
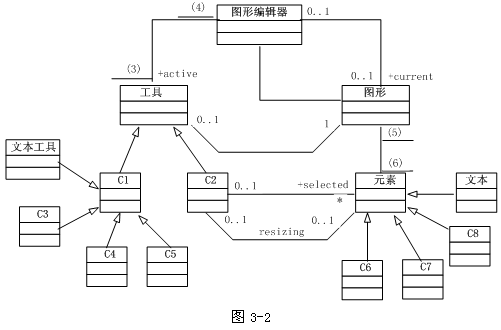
**试题分析**

这里需要区分业务用例与用例（用例即系统）。业务用例其实是对用例思想的一种延续，只是改变了使用场合。用例是从使用者的角度定义“软件系统”的需求。而业务用例不研究“软件系统”需求，它关心是一个“业务组织”对外提供哪些服务，支持那些业务过程。业务用例描述的是业务参与者如何使用业务组织提供的服务的过程。因此业务用例其实是一种业务流程。  
业务对象模型是描述业务用例实现的对象模型，即业务结构以及结构元素如何完成业务用例。

**试题答案**

（46）D（47）C

# **试题90(2011年上半年试题3)**

阅读下列说明和图，回答问题l至问题3，将解答填入答题纸的对应栏内。  
【说明】  
    一个简单的图形编辑器提供给用户的基本操作包括：创建图形、创建元素、选择元素以及删除图形。图形编辑器的组成及其基本功能描述如下：  
   （1）图形由文本元素和图元元素构成，图元元素包括线条、矩形和椭圆。  
   （2）显示在工作空间中，一次只能显示一张图形(即当前图形，current)。  
   （3）提供了两种操作图形的工具：选择工具和创建工具。对图形进行操作时，一次只能使用一种工具(即当前活动工具，active)  
    ① 创建工具用于创建文本元素和图元元素。  
    ② 于显示在工作空间中的图形，使用选择工具能够选定其中所包含的元素，可以选择一个元素，也可以同时选择多个元素。被选择的元素称为当前选中元素( selected)。  
    ③ 种元素都具有对应的控制点。拖拽选定元素的控制点，可以移动元素或者调整元素的大小。  
    现采用面向对象方法开发该图形编辑器，使用UML进行建模。构建出的用例图和类图分别如图3-1和3-2所示。  
  


【问题1】 (4分)  
    根据说明中的描述，给出图3-1中U1和U2所对应的用例，以及(1)和(2)处所对应的关系。  
【问题2】（8分）  
    根据说明中的描述，给出图3.2中缺少的C1～C8所对应的类名以及(3)～(6)处所对应的多重度。  
【问题3】（3分）  
    图3-2中的类图设计采用了桥接(Bridge)设计模式，请说明该模式的内涵。

**试题分析**

    本题考查面向对象开发相关知识，涉及UML用例图、类图以及类图设计时的设计模式。UML目前在面向对象软件开发中广泛使用，是面向对象软件开发考查的重要内容。   
【问题1】  
    本题主要考查用例图。  
    用例之间的关系主要有以下三种：  
   （1）包含关系。当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示它们。用《include》表示。  
   （2）扩展关系。如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。用《extend》表示。  
   （3）泛化关系。当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。  
    在本题中，从用例图中，我们不难看出U1和U2都与选择元素用例有关系。然后根据题目的描述，可知U1和U2应该分别是移动元素和调整元素的大小，这里我们假定U1是移动元素用例，而U2是调整元素的大小用例。那么接着我们再来确定空（1）与空（2）的内容。这里很显然U1和U2与选择元素用例的关系是扩展关系，因此空（1）与空（2）都应该填<<extend>>。  
【问题2】  
    本问题考查类图。对于这个题目，我们应该结合题目的描述及给出的类图来求解。从题目给出的类图中我们可以看出，C1和C2是继承（泛化）于工具类的，而题目描述告诉我们系统提供了两种操作图形的工具，即选择工具和创建工具，因此C2与C2应该分别是选择工具和创建工具之一，然后我们可以看到文本工具类是继承于C1的，结合题目描述“创建工具用于创建文本元素和图元元素”，我们可以知道C1应该为创建工具类，而C2应该为选择工具类，另外，根据题目描述“图元元素包括线条、矩形和椭圆”，可以知道C6至C8应该分别是线条类、矩形类及椭圆类，当然这三者的答案可以互换。而要能得到这些图形元素，就应该有相应的画图工具，因此C3至C5应该分别是线条工具类、矩形工具类及椭圆工具类，这三者的答案也可以互换。  
    在UML中，多重度又称重复度，多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用\*号表示）。最常见的多重性有0..1、0..\*、1..1和1..\*，而\*与0..\*是等价的。  
    由于一个图形编辑器实例可以有一个工具实例，当然也可以没有工具实例，而一个工具实例只能属于一个图形编辑器实例，因此空（3）与空（4）分别为0..1和1。而一个图形至少需要包含一个图形元素，也可以包含多个图形元素，而一个图形元素实例只能属于一个图形实例，所以空（5）与空（6）应该分别是1和1..\*。  
【问题３】  
    本问题主要考查桥接模式的基本内容。  
    桥接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，对一个抽象的实现部分的修改应该对使用它的程序不产生影响。

**试题答案**

（3）

【问题1】  
UI:移动元素    U2：调整元素大小（Ul和U2的答案可以互换）  
(1)<<extend>>    (2)<<extend>>  
【问题2】  
C1：创建工具    C2：选择工具  C3：线条工具  C4：矩形工具  
C5:椭圆工具    C6：线条    C7：矩形    C8：椭圆  
注：C3～C5的答案可以互换；C6-C8的答案可以互换。  
(3)0..1 (4)1    (5)1    (6)1..\*  
【问题3】  
桥接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，对一个抽象的实现部分的修改应该对使用它的程序不产生影响。

# **试题91(2011年上半年试题37)**

面向对象分析的第一步是（  ）。

（37）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**

面向对象分析的任务是了解问题域所涉及的对象、对象间的关系和操作，然后构造问题的对象模型。问题域是指一个包含现实世界事物与概念的领域，这些事物和概念与所设计的系统要解决的问题有关。因此面向对象分析的第一步是要确定问题域。

**试题答案**

（37）C

# **试题92(2011年上半年试题38)**

下列关于一个类的静态成员的描述中，不正确的是（  ）。

（38）A．类的静态方法只能访问该类的静态数据成员   
B.静态数据成员可被该类的所有方法访问   
C.该类的对象共享其静态数据成员的值   
D.该类的静态数据成员的值不可修改

**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。它们一般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。静态成员和普通数据成员区别较大，体现在下面几点： 　　  
（1）普通数据成员属于类的一个具体的对象，只有对象被创建了，普通数据成员才会被分配内存。而静态数据成员属于整个类，即使没有任何对象创建，类的静态数据成员变量也存在。  
（2）外部访问类的静态成员只能通过类名来访问。  
（3）类的静态成员函数无法直接访问普通数据成员（可以通过类的指针等作为参数间接访问），而类的任何成员函数都可以访问类的静态数据成员。  
（4）类的静态方法只能访问该类的静态数据成员。　　  
另外，静态成员和类的普通成员一样，也具有public、protected、private3种访问级别，也可以具有返回值及被修改等性质。

**试题答案**

（38）D

# **试题93(2011年上半年试题39-41)**

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（  ）和（  ）表现；动态方面由交互图、（  ）表现。

（39）A．类图   
B.状态图   
C.活动图   
D.序列图   
  
（40）A．交互图   
B.对象图   
C.通信图   
D.定时图   
  
（41）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**

本题主要考查UML的设计视图。  
UML 2.0包括14种图，分别列举如下：  
（1）类图。描述一组类、接口、协作和它们之间的关系。在OO系统的建模中，最常见的图就是类图。类图给出了系统的静态设计视图，活动类的类图给出了系统的静态进程视图。  
（2）对象图。描述一组对象及它们之间的关系。对象图描述了在类图中所建立的事物实例的静态快照。和类图一样，这些图给出系统的静态设计视图或静态进程视图，但它们是从真实案例或原型案例的角度建立的。  
（3）构件图。描述一个封装的类和它的接口、端口，以及由内嵌的构件和连接件构成的内部结构。构件图用于表示系统的静态设计实现视图。对于由小的部件构建大的系统来说，构件图是很重要的。构件图是类图的变体。  
（4）组合结构图。描述结构化类（例如，构件或类）的内部结构，包括结构化类与系统其余部分的交互点。组合结构图用于画出结构化类的内部内容。  
（5）用例图。描述一组用例、参与者及它们之间的关系。用例图给出系统的静态用例视图。这些图在对系统的行为进行组织和建模时是非常重要的。  
（6）顺序图。是一种交互图（interaction diagram），交互图展现了一种交互，它由一组对象或参与者以及它们之间可能发送的消息构成。交互图专注于系统的动态视图。顺序图是强调消息的时间次序的交互图。  
（7）通信图。也是一种交互图，它强调收发消息的对象或参与者的结构组织。顺序图和通信图表达了类似的基本概念，但它们所强调的概念不同，顺序图强调的是时序，通信图强调的是对象之间的组织结构（关系）。在UML 1.X版本中，通信图称为协作图（collaboration diagram）。  
（8）定时图。也是一种交互图，它强调消息跨越不同对象或参与者的实际时间，而不仅仅只是关心消息的相对顺序。  
（9）状态图。描述一个状态机，它由状态、转移、事件和活动组成。状态图给出了对象的动态视图。它对于接口、类或协作的行为建模尤为重要，而且它强调事件导致的对象行为，这非常有助于对反应式系统建模。  
（10）活动图。将进程或其他计算结构展示为计算内部一步步的控制流和数据流。活动图专注于系统的动态视图。它对系统的功能建模和业务流程建模特别重要，并强调对象间的控制流程。  
（11）部署图。描述对运行时的处理节点及在其中生存的构件的配置。部署图给出了架构的静态部署视图，通常一个节点包含一个或多个部署图。  
（12）制品图。描述计算机中一个系统的物理结构。制品包括文件、数据库和类似的物理比特集合。制品图通常与部署图一起使用。制品也给出了它们实现的类和构件。  
（13）包图。描述由模型本身分解而成的组织单元，以及它们之间的依赖关系。  
（14）交互概览图。是活动图和顺序图的混合物。  
其中类图、对象图、用例图、组件图及配置图为静态图，其它的为动态图。

**试题答案**

（39）A（40）B（41）D

# **试题94(2011年上半年试题42)**

UML中关联的多重度是指（  ）。

（42）A．一个类中被另一个类调用的方法个数   
B.g一个类的某个方法被另一个类调用的次数   
C.一个类的实例能够与另一个类的多少个实例相关联   
D.两个类所具有的相同的方法和属性

**试题分析**

在UML中，关联的多重度是指一个类的实例能够与另一个类的多少个实例相关联。它又称为重复度多重度表示为一个整数范围n..m，整数n定义所连接的最少对象的数目，而m则为最多对象数（当不知道确切的最大数时，最大数用\*号表示）。最常见的多重性有0..1、0..\*、1..1和1..\*。

**试题答案**

（42）C

# **试题95(2011年上半年试题43)**

在面向对象软件开发过程中，采用设计模式（  ）。

（43）A．以复用成功的设计   
B.以保证程序的运行速度达到最优值   
C.以减少设计过程创建的类的个数   
D.允许在非面向对象程序设计语言中使用面向对象的概念

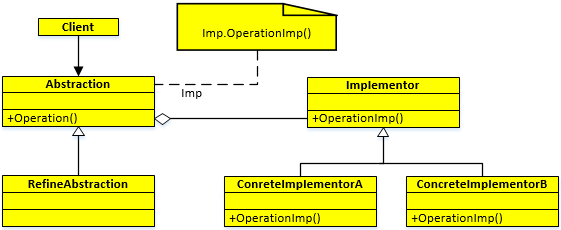
**试题分析**

模式是一种问题的解决思路，它已经适用于一个实践环境，并且可以适用于其它环境。设计模式通常是对于某一类软件设计问题的可重用的解决方案，将设计模式引入软件设计和开发过程，其目的就在于要重用成功的软件开发经验。

**试题答案**

（43）A

# **试题96(2011年上半年试题44-45)**

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


（44）A．Bridge（桥接）   
B.Composite（组合）   
C.Facade（外观）   
D.Singleton（单例）   
  
（45）A．Abstraction   
B.ConcreteImplementorA   
C.ConcreteImplementorB   
D.Implementor

**试题分析**

本题主要考查常见的设计模式。  
在本题中，根据题目给出的图，我们不难看出该图描述的是桥接模式，它的显著特征是它将抽象部分与实现部分分离，使它们可以相互独立地变化。我们不难从题目给出的图中看出，左边的是抽象类接口，而右边都是实现类接口，显然实现了分离。抽象类接口的下面是抽象的扩充，而实现类接口的下面是具体实现，因此他们可以相互独立地变化。其中：  
Abstraction：抽象类定义抽象类的接口。维护一个Implementor（实现抽象类）的对象。  
RefinedAbstraction：扩充的抽象类，扩充由Abstraction定义的接口。  
Implementor：实现类接口，定义实现类的接口，这个接口不一定要与Abstraction的接口完全一致，事实上这两个接口可以完全不同，一般的讲Implementor接口仅仅给出基本操作，而Abstraction接口则会给出很多更复杂的操作。  
ConcreteImplementor：具体实现类，实现Implementor定义的接口并且具体实现它。

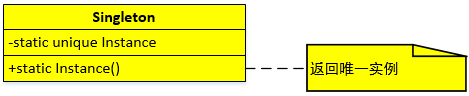
**试题答案**

（44）A（45）D

# **试题97(2011年上半年试题46)**

以下关于Singleton（单例）模式的描述中，正确的是（  ）。  
（46）A．它描述了只有一个方法的类的集合  
B.它描述了只有一个属性的类的集合  
C.它能够保证一个类的方法只能被一个唯一的类调用  
D.它能够保证一个类只产生唯一的一个实例

**试题分析**

单例模式的意图是确保某个类只有一个实例，且能自行实例化，并向整个系统提供这个实例。单例模式模式的结构图如下。  
  
其中，Singleton即单例，提供一个instance（实例化）的方法，让客户可以使用它的唯一实例。内部实现只生成一个实例。

**试题答案**

（46）D

# **试题98(2011年上半年试题47)**

 （  ）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。

（47）A．Adapter（适配器）模式   
B.Command（命令）模式   
C.Singleton（单例）模式   
D.Strategy（策略）模式

**试题分析**

本题也是考查设计模式。在做类题时，需要我们对各设计模式有一个正确的理解才能完成。  
适配器模式的意图就是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。  
命令模式的意图是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，以及支持可撤消的操作。  
单例模式的意图是确保某个类只有一个实例，且能自行实例化，并向整个系统提供这个实例。  
策略模式的意图是定义一系列的算法，把它们一个个封装起来，并且使它们可相互替换，该模式使得算法可独立于使用它的客户而变化。

**试题答案**

（47）A

# **试题99(2010年下半年试题37-42)**

开-闭原则（Open-Closed Principle，OCP )是面向对象的可复用设计的基石。开-闭原则是指一个软件实体应当对（  ）开放，对（  ）关闭；里氏代换原则（Liskov  Substitution Principle，LSP )是指任何（  ）可以出现的地方，（  ）一定可以出现。依赖倒转原则（Dependence Inversion Principle，DIP）就是要依赖于（  ）而不依赖于（  ），或者说要针对接口编程，不要针对实现编程。

（37）A．修改   
B.扩展   
C.分析   
D.设计   
  
（38）A．修改   
B.扩展   
C.分析   
D.设计   
  
（39）A．变量   
B.常量   
C.基类对象   
D.子类对象   
  
（40）A．变量   
B.常量   
C.基类对象   
D.子类对象   
  
（41）A．程序设计语言   
B.建模语言   
C.实现   
D.抽象   
  
（42）A．程序设计语言   
B.建模语言   
C.实现   
D.抽象

**试题分析**

开-闭原则要求一个软件实体应当对扩展开放，对修改关闭。也就是说，我们在设计一个模块的时候，应当使这个模块可以在不被修改的前提下被扩展，换句话说就是，应当可以在不必修改源代码的情况下改变这个模块的行为。  
里氏代换原则要求子类型必须能够替换它们的基类型，所以在里氏代换原则中，任何可基类对象可以出现的地方，子类对象也一定可以出现。  
依赖倒转原则是：要依赖于抽象，不要依赖于具体。也就是常说的要针对接口编程，不要针对实现编程。

**试题答案**

（37）B（38）A（39）C（40）D（41）D（42）C

# **试题100(2010年下半年试题43-45)**

（  ）是一种很强的“拥有”关系，“部分”和“整体”的生命周期通常一样。整体对象完全支配其组成部分，包括它们的创建和销毁等;（  ）同样表示“拥有”关系，但有时候“部分”对象可以在不同的“整体”对象之间共享，并且“部分”对象的生命周期也可以与“整体”对象不同，甚至“部分”对象可以脱离“整体”对象而单独存在。上述两种关系都是（  ）关系的特殊种类。

（43）A．聚合   
B.组合   
C.继承   
D.关联   
  
（44）A．聚合   
B.组合   
C.继承   
D.关联   
  
（45）A．聚合   
B.组合   
C.继承   
D.关联

**试题分析**

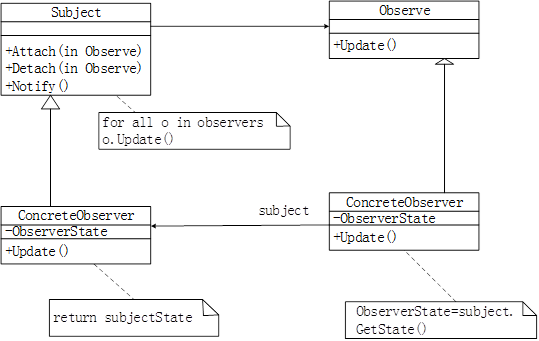
本题主要考查我们对类之间一些常用关系的理解。本题我们主要要清楚组合与聚合的联系和区别。组合与聚合都体现着“部分”和“整体”的关系，但组合是一种很强的“拥有”关系，“部分”和“整体”的生命周期通常一样。整体对象完全支配其组成部分，包括它们的创建和销毁等；而聚合有时候“部分”对象可以在不同的“整体”对象之间共享，并且“部分”对象的生命周期也可以与“整体”对象不同，甚至“部分”对象可以脱离“整体”对象而单独存在。  
组合与聚合都是关联关系的特殊种类。

**试题答案**

（43）B（44）A（45）D

# **试题101(2010年下半年试题46-47)**

下面的UML类图描绘的是（  ）设计模式。关于该设计模式的叙述中，错误的是（  ）。



（46）A．桥接   
B.策略   
C.抽象工厂   
D.观察者   
  
（47）A．该设计模式中的Observer需要维护至少一个Subject对象   
B.该设计模式中的ConcreteObserver可以绕过Subject及其子类的封装   
C.该设计模式中一个Subject对象需要维护多个Observer对象   
D.该设计模式中Subject需要通知Observer对象其自身的状态变化

**试题分析**

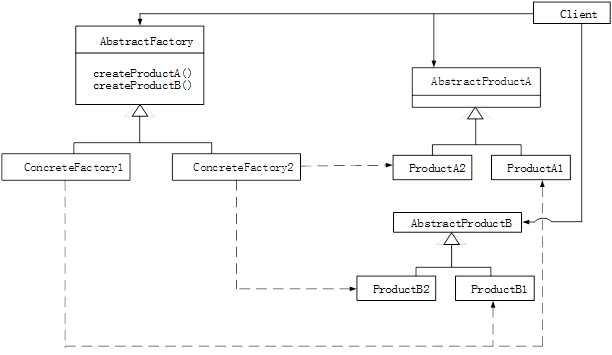
本题主要考查设计模式的相关知识。  
桥接模式将抽象部分与实现部分分离，使得它们两部分可以独立地变化。  
策略模式定义一系列的算法，将每一个算法封装起来，并让它们可以相互替换。策略模式让算法独立于使用它的客户而变化。  
抽象工厂提供了一个创建一系列相关或相互依赖对象的接口，而无须指定它们具体的类。抽象工厂模式又称为Kit模式，属于对象创建型模式。  
观察者模式定义对象间的一种一对多依赖关系，使得每当一个对象改变状态，则其相关依赖对象皆得到通知并被自动更新。  
从本题给出的图我们可以判断是观察者模式，其中被观察对象（Subject），具体被观察对象（ConcreteSubject），观察者（Observer）及具体观察者（ConcreteObserver）。在本设计模式中，Observer需要维护至少一个Subject对象，一个Subject对象需要维护多个Observer对象，另外Subject需要通知Observer对象其自身的状态变化。

**试题答案**

（46）D（47）B

# **试题102(2010年上半年试题43-44)**

 如下UML类图表示的是（  ）设计模式。以下关于该设计模式的叙述中，错误是（  ）。



（43）A． 工厂方法  
B.策略  
C.抽象工厂  
D.观察者  
  
（44）A．提供创建一系列相关或相互依赖的对象的接口，而无需指定这些对象所属的具体类  
B.可应用于一个系统要由多个产品系列中的一个来配置的时候  
C.可应用于强调一系列相关产品对象的设计以便进行联合使用的时候  
D. 可应用于希望使用已经存在的类，但其接口不符合需求的时候

**试题分析**

本题考查面向对象方法中的设计模式知识。  
　　题中的类图是抽象工厂设计模式，该设计模式的意图是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。使用抽象工厂设计模式的常见情形是：一个系统要独立于其产品的创建、组合和表示时；一个系统要由多个产品系列中的一个来配置时；当需要强调一系列相关的产品对象的设计以便进行联合使用时；当提供一个产品类库，而只想显示它们的接口不是实现时。对于希望使用已经存在的类，但其接口不符合需求的情形，应当考虑适配器设计模式。

**试题答案**

（43）C（44）D