1. 软件体系结构风格：是描述某一特定应用领域中系统组织方式的惯用模式。
2. 高内聚度：是对一个类中的各个职责之间相关程度和集中程度的度量。
3. 模式：描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，可以无数次地使用那些已有的解决方案，无需再重复相同的工作。
4. 构件是可预制和可重用的软件部件，是组成体系结构的基本计算单元或数据存储单元。
5. 连接件也是可预制和可重用的软件部件，是构件之间的连接单元，表现为构件之间交互建模的体系结构构造及其支配这些交互的规则。
6. 约束描述构件和连接件之间的关系，体现为构件与连接件的连接图，又称体系结构配置。
7. 设计模式是一套被反复使用的，多数知晓的，经过分类编目的，代码设计经验的总结。
8. 软件设计是把许多事物和问题抽象起来，并且抽象出它们不同的层次和角度，一般包括结构设计、数据设计、接口设计和过程设计。
9. 逻辑视图（Logic View）：主要是用来描述系统的功能需求，即系统提供给最终用户的服务。

10、进程视图（Process View）：侧重系统的运行特性，关注非功能性的需求（性能，可用性），服务于系统集成人员，方便后续性能测试。

1、在找出了类的继承关系后，通常可以用抽象类(接口)来表示最上层的基类。

2、当我们想创建一个具体的对象而又不希望指定具体的类时，可以使用“抽象工厂”模式。

3、当我们想将抽象部分和实现部分分离时，可以使用桥接模式。

4、当我们想用不同的请求对客户进行参数化时，可以使用命令(用户接口的请求)模式。

5、当我们想封装不同算法并使它们可相互替换时，可以使用策略模式。

6、面向对象系统中功能复用的两种最常用技术是类继承,接口继承。

7、设计模式中应优先使用的复用技术是接口继承。

8、接口是可以在整个模型中反复使用的一组行为，是一个没有属性而只有操作的类。

9、在泛化关系中，子类可以替代父类。也就是说，后者出现的地方，前者都可以出现。但是反过来却不成立。

10、实现的符号和继承的符号有相似之处，两者的唯一差别是实现关系用虚线表示，继承关系用实线表示。

11、UML提供了一系列的图支持面向对象的分析与设计，其中类图给出系统的静态设计视图；用例图对系统的行为进行组织和建模，是非常重要的；时序图和协作图都是描述系统动态视图的交互图，其中时序图描述了以时间顺序组织的对象之间的交互活动，协作图强调收发消息的对象的组织结构。

12、软件模式并非仅限于设计模式，还包括架构模式、分析模式和过程模式（实现模式）等。

13、软件体系结构包括构件（Component）、连接件（Connector）和约束（Constrain）或配置（Configuration）三大要素，它是可预制和可重构的软件框架结构。

14、构件可以是一组代码，如程序的模块；也可以是一个独立的程序，如数据库服务器。连接件可以是过程调用、管道、远程过程调用（RPC）等，用于表示构件之间的相互作用。约束一般为对象连接时的规则，或指明构件连接的形式和条件

15、软件体系结构的建立是需求分析的衔接和体现，关键需求是软件体系结构设计的核心，软件体系结构设计既要满足功能性需求，同时又要满足适应需求变化的非功能性需求。

16、在面向对象技术中，通过抽象、封装、继承，可以用对象模型来代表逻辑视图，可以用类图（Class Diagram）来描述逻辑视图。

17、在逻辑视图中，系统分解成一系列的功能抽象、功能分解与功能分析，这些主要来自问题领域（Problem Definition）。

18、开发视图描述了在开发环境中软件的静态组织结构，即关注软件开发环境下实际模块的组织，服务于软件编程人员。

19、架构：简单的说架构就是一个蓝图，是一种设计方案，将客户的不同需求抽象成为抽象组件，并且能够描述这些抽象组件之间的通信和调用。

20、框架：软件框架是项目软件开发过程中提取特定领域软件的共性部分形成的体系结构，不同领域的软件项目有着不同的框架类型。框架不是现成可用的应用系统,而是一个半成品，提供了诸多服务，开发人员进行二次开发，实现具体功能的应用系统。

21、事件系统风格软件体系结构的构件是一些模块，这些模块既可以是一些过程，又可以是一些事件的集合，是非命名的过程，它们之间交互的连接件往往是以过程之间的隐式调用(Implicit Invocation)来实现的。

22、在建立软件体系结构时，设计师主要从结构的角度对整个系统进行分析，选择恰当的构件、构件间的相互作用以及它们的约束，最后形成一个系统框架以满足用户的需求，为软件设计奠定基础。

1、请简要说明面向对象设计的设计原则。

面向接口编程原则是面向对象设计（OOD）的第一个基本原则，它的含义是使用接口和同类型的组件通讯，即，对于所有完成相同功能的组件，应该抽象出一个接口，它们都实现该接口。

优先使用对象组合原则，对象组合是通过获得对其他对象的引用而在运行时刻动态定义的。组合要求对象遵守彼此的接口约定，进而要求更仔细地定义接口，而这些接口并不妨碍将一个对象和其他对象一起使用。

分离可变和不可变部分原则，这是面向对象实施的第三个原则。如果使用继承的复用技术，我们可以在抽象基类中定义好不可变的部分，而由其子类去具体实现可变的部分，不可变的部分不需要重复定义，而且便于维护。

2、请简要说明代理模式与适配器的区别

二者都属于一种衔接性质的功能。代理对象和被代理对象的接口是同一个，但是客户没法直接访问被代理者，只能通过代理对象去完成被代理对象的访问。而适配器模式是将多个子系统封装起来，提供一个统一的外部接口，客户只需要使用这个外部接口即可访问对象的子系统了。

3、请简要说明代理模式与装饰模式的区别

设计目的不同，代理模式的目的是当直接访问一个实体不方便或不符合需要时，为这个实体提供一个替代者。装饰模式适用于编制时不能确定对象的全部功能的情况。

4、请简述云计算平台的架构。

按照云计算平台提供的服务种类，划分出了云计算平台的三层架构，IaaS，提供 CPU、网络、存储等基础硬件的云服务，即将各种底层的计算（比如虚拟机）和存储等资源作为服务提供给用户。PaaS，提供类似于操作系统层次的服务与管理，即将一个应用的开发和部署平台作为服务提供给用户。SaaS，就是我们所熟悉的软件即服务，即将应用主要以基于Web的方式提供给客户。

5、请简要说明批处理风格与管道过滤器风格的异同

共同点是把任务都分解成一系列固定顺序的处理过程（计算单元），彼此间只通过数据传递交互，处理过程之间均互不调用。

不同点是前者是面向过程设计，数据以整体的形式传输，数据总量有限，输入时可随机存取，无合作性、无交互性，处理过程只能串行连接，允许处理过程内部递归，不允许跨越或涵盖多个处理过程递归，无法并发，构件粒度较大，延迟高、实时性差。后者是面向对象设计，数据用数据流的形式增量传输，数据量可以无限制，数据结果延迟小，输入时处理局部化，有反馈、可交互，构件粒度小，实时性好，可并发。

6、请简要说明事件系统风格软件体系结构

所谓事件驱动架构风格是指构件不直接调用一个过程，而是触发或广播一个或多个事件。系统中其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程，一个事件的触发就导致了另一个模块中过程的调用。

7、请简要说明需求的描述方法有哪些，并进行分别说明。

需求的描述方法主要有用例视图和软件概念模型。

用例视图包括用例和场景，这些用例和场景包括在架构方面具有重要意义的行为、类或技术风险中，它是用例模型的子集。

软件的概念模型（又称领域模型、信息模型）。用例着重于描述一个个具体的需求，概念模型则从业务的角度描述了整个软件系统所要完成的功能中涉及的所有概念以及彼此之间的关系。

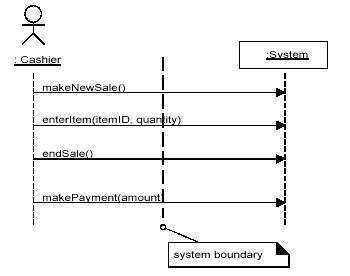
1、画出下面场景的时序图：

1.收款员启动一次销售(makeNewSale())

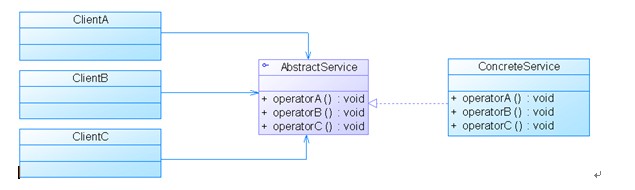
2.收款员输入商品标识(enterItem(itemID,quantity))

3.销售结束，系统计算并显示总金额（endSale（））

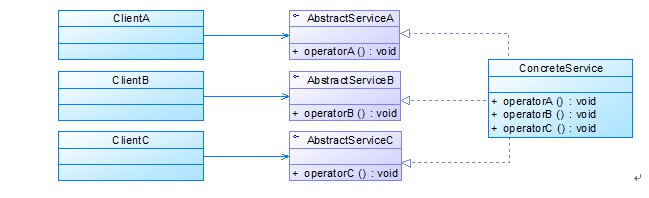
4.顾客付款，系统处理支付。(makePayment(amount))

****

2、一个拥有多个客户类的系统，在系统中定义了一个巨大的接口（胖接口）AbstractService来服务所有的客户类。如图所示，请指出设计存在的问题，并基于接口分离原则进行重构。

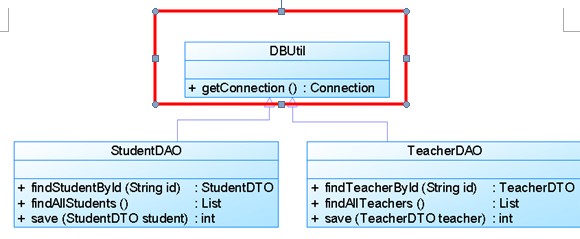


答案：类应该完全依赖相应的专门的接口，这样一个功能集中的接口违背了接口隔离原则，一旦针对某个客户的方法发生改变，就需要进行重新编译和发布，因此其他客户都会受到影响。



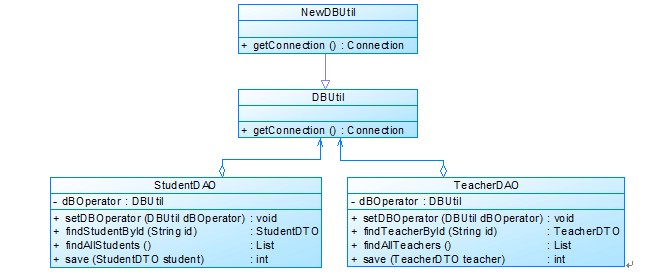
这样做既满足了接口隔离原则，又满足了单一原则，但是也带来了很多的不便，类增多了。

3、某教学管理系统部分[数据库](http://lib.csdn.net/base/mysql)访问类设计如图所示，请指出设计存在的问题，并基于合成复用原则进行重构。



答案：因为如果需要更换数据库连接方式，则需要修改DBUtil类源代码。

例如原来采用JDBC连接数据库，现在采用数据库连接池连接；如果StudentDAO采用JDBC连接，但是TeacherDAO采用连接池连接，则需要增加一个新的DBUtil类，并修改StudentDAO或TeacherDAO的源代码，使之继承新的数据库连接类，这样导致违背开闭原则，系统扩展性较差，所以应尽量使用对象组合而不是用继承来达到复用的目的。



1、以组装电脑为例，请使用Java利用创建者模式实现电脑产品的组装实现。

参考答案：

public class Commodity{

String commodity ="";

public Commodity (Part partA,Part partB,Part partC) {//由各个部分组成

this. commodity = partA.part+"\n";

this. commodity = product+partB.part+"\n";

this. commodity = product+partC.part;

System.out.println("我的机器配置为：\n"+ commodity);

}

}

public class Part {

String part="";

public Part(String part){

this.part = part;

}

}

public interface Computer {

//组装部件A　比如CPU

void buildPartA();

//组装部件B 比如主板

void buildPartB();

//组装部件C 比如硬盘

void buildPartC();

//返回最后组装成品结果 (返回最后组装好的电脑)

//成品的组装过程不在这里进行，而是由组装师（Assembler）类完成的。

//从而实现了过程和部件的分离

Product getProduct();

}

public class Assembler{

private Computer computer;

public Assembler(Computer computer) { //主要任务是装电脑

this.computer = computer;

}

// 将部件partA partB partC最后组成复杂对象

//这里是将主板、CPU和硬盘组装成PC的过程

public void construct() {

computer.buildPartA();

computer.buildPartB();

computer.buildPartC();

}

}

public class MyComputer implements Computer {

Part partA, partB, partC;

public void buildPartA() {

partA = new Part("P42.4 CPU");

}

public void buildPartB() {

partB = new Part("Inter 主板");

}

public void buildPartC() {

partC = new Part("80G硬盘");

}

public Product getProduct() {

//返回最后组装成品结果

Commodity myComputer = new Commodity (partA,partB,partC);

return myComputer;

}

}

public class MyComputerTest {

public static void main(String args[]){

MyComputer myComputer = new MyComputer(); //组装我的电脑

Assembler assembler = new Assembler( myComputer ); //派某一位组装师

assembler.construct(); //组装师进行组装过程

Commodity commodity = myComputer.getProduct(); //卖给我的电脑（商品）

}

}

2、饮品店制作各种果汁的过程都是相似的，先加水，再加水果，再加冰......步骤的顺序是不能改变的，老板特地写了一个详细的制作步骤在一张纸上，要求服务员根据具体制作果汁的种类来实现具体的步骤。请使用Java利用模板方法模式实现这个程序。

参考答案：

abstract class FruitJuice {

public abstract void AddWater(); // 加水

public abstract void AddFruit(); // 加水果

public abstract void AddIceBlock(); // 加冰

public abstract void AddLemonSlice(); // 加柠檬片

public void MakeJuice() // 制作果汁

{

AddWater();

AddFruit();

AddIceBlock();

AddLemonSlice();

System.out.println("果汁已经做好！");

}

}

class OrangeJuice extends FruitJuice { // 橘子汁

public void AddWater() {

System.out.println("加30ML的水");

}

public void AddFruit() {

System.out.println("加橘子");

}

public void AddIceBlock() {

System.out.println("加冰块");

}

public void AddLemonSlice() {

System.out.println("不加柠檬片");

}

}

class GrapeJuice extends FruitJuice { // 葡萄汁

public void AddWater() {

System.out.println("加50ML的水");

}

public void AddFruit() {

System.out.println("加葡萄");

}

public void AddIceBlock() {

System.out.println("不加冰块");

}

public void AddLemonSlice() {

System.out.println("加柠檬片");

}

}

public class Client {

public static void main(String args[]) {

OrangeJuice juice1 = new OrangeJuice();

GrapeJuice juice2 = new GrapeJuice();

System.out.println("开始制作橘子汁\n");

juice1.MakeJuice();

System.out.println("\n开始制作葡萄汁\n");

juice2.MakeJuice();

}

}

1、在找出了类的继承关系后，通常可以用 或者 来表示最上层的基类。

2、想创建一个具体的对象而又不希望指定具体的类时，可以使用 模式。

3、当我们想将抽象部分和实现部分分离时，可以使用 模式。

4、接口是可以在整个模型中反复使用的一组 ，是一个没有 而只有 的类。

5、实现的符号和继承的符号有相似之处，两者的唯一差别是实现关系用 表示，继承关系用 表示。

6、软件模式并非仅限于设计模式，还包括 模式、 模式和

模式等。

7、构件可以是一组 ，如程序的模块；也可以是一个独立的 ，如数据库服务器。连接件可以是 、 、 等，用于表示构件之间的相互作用。约束一般为对象连接时的 ，或指明构件连接的 和 。

1、答案：抽象类、接口

2、答案：抽象工厂模式

3、答案：桥接模式

4、答案：行为、属性、操作

5、答案：虚线、实现

6、答案：架构模式、分析模式、过程模式（实现模式）

7、答案：代码、程序、过程调用、管道、远程过程调用（RPC）、规则、形式、条件

1、算法是设计模式，因为算法致力于设计问题而非解决问题。（ ）

2、简单的说架构就是一个蓝图，是一种设计方案，将客户的不同需求抽象成为抽象组件，并且能够描述这些抽象组件之间的通信和调用。（ ）

3、设计模式中应优先使用的复用技术是接口继承。（ ）

4、数据流风格的软件则是系统中的组件相互独立，不存在一个组件调用另外一个组件的现象，程序的运行由相互调用控制。（ ）

5、在逻辑视图中，系统分解成一系列的功能抽象、功能分解与功能分析，这些主要来自问题领域。（ ）

1、答案：× 2、答案：√ 3、答案：√

4、答案：× 5、答案：√

1、 软件危机的表现不包括（ ）：

A、用户需求不明确 B、软件开发成本日益增长

B、开发进度难以控制 D、维护工作困难。

2、不属于模式构成部分的是 （ ）。

A、周境 B、动机 C、结果 D、解决方案

3、下列说法错误的一项的是（ ）。

A、逻辑视图主要支持系统的功能需求，即系统提供给最终用户的服务。

B、开发视图也称模块视图，主要侧重于软件模块的组织和管理。

C、进程视图侧重与系统的运行特性，主要关注一些功能性需求，例如系统的性能和可用性。

D、物理视图主要考虑如何把软件映射到硬件上，它不需要考虑到系统性能、规模、可靠性等。

4、软件模式的基础结构不包括。（ ）。

A、问题描述 B、前提条件

C、解法和效果 D、性能

5、( )模式一般是给出本地内存资源节省的一个方案，并不适合互联网上的分布式应用的情况。

A、原型模式 B、单例模式 C、享元模式 D、代理模式

6、面向对象设计思想中，类关系的耦合程度表述正确的是（ ）。

A、聚合→依赖→组合→继承，类与类之间的关系愈加紧密，耦合性越强。

B、依赖→聚合→组合→继承，类与类之间的关系愈加紧密，耦合性越强。

C、依赖→组合→聚合→继承，类与类之间的关系愈加紧密，耦合性越强。

D、继承→组合→聚合→依赖，类与类之间的关系愈加紧密，耦合性越强。

7、以下哪些特征不属于动态体系结构特征（ ）？

A、可构造性动态特征 B、适应性动态特征

C、智能性动态特征 D、安全性动态特征

8、“对一个类而言，应该仅有一个引起它变化的原因。”体现的是（ ）。

A、里氏代换原则 B、单一职责原则

C、接口隔离原则 D、迪米特法则

9、下面哪个不属于设计模式方法分类（ ）。

A、Coad的面向对象模式 B、代码模式

C、框架应用模式 D、遗传算法

10、面向对象的语言当中，( )使得实现和应用可以复用?

A、多态性。 B、数据的抽象化和继承的关系。

C、抽象化和封装。 D、关注点分离。

1、答案：A 2、答案：C 3、答案：D

4、答案：D 5、答案：C 6、答案：B

7、答案：D 8、答案：B 9、答案：D

10、答案：A

1、当我们想用不同的请求对客户进行参数化时，可以使用 模式。

2、当我们想封装不同算法并使它们可相互替换时，可以使用 模式。

3、在面向对象技术中，通过抽象、封装、继承，可以用 来代表逻辑视图，可以用 来描述逻辑视图。

4、在泛化关系中， 类可以替代 类。也就是说， 出现的地方， 都可以出现。但是反过来却不成立。

5、UML时序图描述了以时间顺序组织的对象之间的 ，协作图强调收发消息的对象的 。

6、软件体系结构包括 、 和 或 三大要素，它是 和 的软件框架结构。

7、软件体系结构的建立是 的衔接和体现， 是软件体系结构设计的核心，软件体系结构设计既要满足 需求，同时又要满足适应需求变化的 需求。

1、答案：命令模式

2、答：策略模式

3、答案：对象模型、类图

4、答案：子类、父类、后者（父类）、前者（子类）

5、答案：交互活动、组织结构

6、答案：构件（Component）、连接件（Connector）和约束（Constrain）或配置（Configuration）、可预制、可重构

7、答案：需求分析、关键需求、功能性需求、非功能性需求

1、在建立软件体系结构时，设计师主要从功能的角度对整个系统进行分析，选择恰当的构件、构件间的相互作用以及它们的约束，最后形成一个系统框架以满足用户的需求，为软件设计奠定基础。（ ）

2、软件框架是项目软件开发过程中提取特定领域软件的共性部分形成的体系结构，框架不是现成可用的应用系统，而是一个半成品，提供了诸多服务，开发人员进行二次开发，实现具体功能的应用系统。（ ）

3、事件系统风格软件体系结构的构件是一些模块，这些模块既可以是一些过程，又可以是一些事件的集合，是非命名的过程，它们之间交互的连接件往往是以过程之间的隐式调用来实现的。（ ）

4、开发视图描述了在开发环境中软件的静态组织结构，即关注软件开发环境下实际模块的组织，服务于软件编程人员。（ ）

5、C2是一种基于抽象层的架构风格，可用于创建灵活的、可伸缩的软件系统。（ ）

1、答案：× 2、答案：√ 3、答案：√

4、答案：√ 5、答案：×

1、实现高内聚低耦合行之有效的方式是:（ ）

A、抽象 B、信息隐藏

C、关注点分离 D、封装

2、选项中不属于“4+1”视图模型的是：（ ）

A、逻辑视图 B、物理视图

C、连接视图 D、开发视图

3、软件体系结构的文档要求与软件开发项目中的其他文档是相似的，下列哪一项属于体系结构文档化过程的主要输出结果（ ）？

A、体系结构需求规格说明 B、体系结构概要设计说明

C、体系结构详细设计说明 D、体系结构总体框架说明

4 、面向对象的语言当中，( )使得概念和定义可以复用？

A、多态性。 B、数据的抽象化和继承的关系。

C、抽象化和封装。 D、关注点分离。

5、对动态软件体系结构的形式化描述通常可以采用的方法是（ ）

A、图形化方法 B、代数进程方法 C、逻辑方法 D、以上皆是

6、“子类型必须能够替换掉它们的基类型。”体现的是（ ）。

A、迪米特法则 B、里氏代换原则

C、接口隔离原则 D、单一职责原则

7、当直接创建对象的代价比较大时,采用的设计模式是（ ）

A、原型模式 B、单例模式 C、享元模式 D、代理模式

8、软件设计是把许多事物和问题抽象起来，并且抽象出它们不同的层次和角度，一般不包括（ ）。

A、结构设计 B、数据设计

C、接口设计和过程设计 D、集成设计

9、软件体系结构设计有哪些过程？（ ）

①提出软件体系结构模型

②把已标识的构件映射到软件体系构件中

③分析构件之间的相互作用

④产生软件体系结构

⑤设计评审

A、①②③ B、①③④ C、①②③④ D、①②③④⑤

10、( )研究的是针对单一问题的设计思路和解决方法（ ）？

A、设计模式 B、软件架构 C、框架 D、类库

1、答案：C 2、答案：C 3、答案：A

4、答案：B 5、答案：D 6、答案：B

7、答案：A 8、答案：D 9、答案：D

10、答案：A

**注意：UML优化设计以及程序设计题主要来源于课后2、3章习题以及实验讲义。**