一、名词解释

1. 软件工程

软件工程就是采用工程的概念、原理、技术和方法实现软件的开发与维护，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，经济地开发出高质量的软件并有效地维护。

2. 模块化

模块化是指解决一个复杂问题时自顶向下逐层分解成若干模块的过程。每个模块完成一个特定的子功能，所有模块按系统结构组合起来，完成整个系统所要求的功能。

3. 对象

对象可以是客观世界中存在的事物，也可以是概念化的实体，它由一组属性和操作组成。属性是用来描述对象静态特征的数据项，是对客观世界实体所具有性质的抽象。操作是用来描述对象的动态特征。

4. 软件维护

软件维护是指在软件产品交付给用户之后，为了改正软件测试阶段未发现的缺陷、改进软件产品的性能、补充软件产品的新功能等，所进行的修改软件的过程。

5.软件项目管理

软件项目管理就是通过计划、组织、控制等一系列活动，合理地配置和使用各种资源，以便在预定成本和期限内开发出符合客户需要的软件的过程。

6.软件生命周期

软件生命周期是指从软件产品的构思开始，到软件需求的确定、软件设计、软件实现、产品测试与验收、投入使用以及产品版本的不断更新，再到最终该产品被市场淘汰的全过程。

7. 黑盒测试

黑盒测试是依据软件的需求规格说明书，检查程序功能是否满足需求的要求，程序是否能适当地接收输入数锯而产生正确的输出信息，并且保持外部信息（如数据库或文件）的完整性。

8. 类

类是对对象的抽象，是对具有相同属性和相同操作的一组相似对象的定义，通常在这种定义中包括对怎样创建该类的新对象的说明。

9. 软件文档

软件文档是用来表示对需求、过程或结果进行描述、定义、规定或认证的图示信息。它描述或规定了软件设计和实现的细节。

10. 代码行技术

代码行技术是根据以往开发类似产品的经验和历史数据，估计实现一个功能需要的源程序行数。把实现每个功能需要的源程序行数累加起来，就得到实现整个软件需要的源程序行数。

11.模型

模型就是为了理解事物而对事物进行的一种抽象，是对事物的一种无歧义的书面描述。通常，模型由一组图形符号和组织这些符号的规则组成。

12.抽象

抽象是指在现实世界中一定事物、状态或过程之间总存在着某些相似的方面，把这些相似的方面集中和概括起来，暂时忽略它们之间的差异。或者说，抽象就是抽取事物的本质特征而暂时不考虑它们的细节。

13.消息

消息就是要求某个对象执行在定义它的那个类中所定义的某个操作的规格说明。一个消息通常包含三个部分的内容：接收消息的对象、消息名和零个或多个变元。

14.基于使用的测试

首先测试几乎不使用服务器类的那些类（称为独立类），把独立类都测试完之后，接下来测试使用独立类的下一个层次的类（称为依赖类）。对依赖类的测试一个层次一个层次地持续进行下去，直至把整个软件系统构造完为止。

15.工程网络图

工程网络图是制定进度计划的一种常用的图形工具，能描绘任务分解情况以及每项作业的开始时间和结束时间，能显式地描绘各个作业彼此间的依赖关系，是系统分析和系统设计的有力工具。

16.结构化需求分析

结构化需求分析是一种面向数据流的需求分析方法。它基于“分解”和“抽象”的基本思想，逐步建立目标系统的逻辑模型，进而描绘出满足用户要求的软件系统。

17.白盒测试法

按照程序内部的逻辑测试程序，检查程序中的每条通路是否都能按照预定要求正确工作。

18.封装

封装就是对象的数据和实现操作的代码集中起来放在对象的内部，组成一个独立的单元，不能从对象的外面直接访问或修改这些数据和代码。

19.划分测试

首先把输入和输出分类，然后设计测试用例以测试划分出的每个类别，以减少测试类时所需要的测试用例数量。划分类别的方法主要有基于状态的划分、基于属性的划分和基于功能的划分。

20.软件质量

软件质量就是软件与明确地和隐含地定义的需求相一致的程度，是软件符合明确叙述的功能和性能需求，文档中明确描述的开发标准，以及所有专业开发的软件都应具有的隐含特征的程度。

21. 软件快速原型

软件快速原型就是快速建立起来的旨在演示目标系统主要功能的程序。构建原型的要点是实现用户看得见的功能，省略目标系统的“隐含”功能。

22. 模块的耦合

模块的耦合是对一个软件结构内不同模块之间互连程度的度量。耦合的强弱取决于模块间接口的复杂程度，进入或访问一个模块的点，以及通过接口的数据。

23. 继承

继承表示类之间的层次关系，使得某些类的对象可以自动拥有另外一个或多个对象的全部属性和操作。继承具有传递性，是子类自动地共享基类中定义的数据和方法的机制。

24. 基于线程的测试

基于线程的测试是把响应系统的一个输入或一个事件所需要的一组类集成起来，分别集成并测试每个线程，同时应用回归测试以保证没有产生副作用。

25. 软件配置管理

软件配置管理是在计算机软件的整个生命周期内管理变化的一组活动。具体地说，这组活动用来实现四个功能：标识变化、控制变化、确保适当地实现了变化、向需要知道这方面信息的人报告变化。

26. 数据流图

数据流图是结构化分析方法中用于表示逻辑系统模型的一种工具，它从数据传递和加工的角度，以图形的方式来刻画数据流从输入到输出的变换过程。

27. 模块的内聚

模块的内聚标志着一个模块内各个元素彼此结合的紧密程度，它是信息隐藏和局部化概念的自然扩展。 设计模块时应力求做到高内聚。

28. 多态

多态是指在类等级的不同层次中可以共享一个行为的名字，但在不同层次中的每个类却各自按照自己的需要来实现这个行为。当对象接收到发送给它的消息时，根据该对象所属于的类动态选用在类中定义的实现算法。

29. 随机测试

随机测试就是让类实例随机执行一些类内定义的操作，以测试类的状态。如果应用系统的性质对操作的应用施加了一些限制，则可在最小操作序列的基础上随机增加一些操作作为测试该类的测试用例。

30. 基线

基线是指已经通过了正式复审的规格说明或中间产品，可以作为进一步开发的基础，并且只有通过正式的变化控制过程才能改变它。

二、简述题

1. 简述在需求分析的过程中将建立的模型，以及每一种模型的建立工具。

在需求分析的过程中，通常将建立三种模型：数据模型、功能模型和行为模型。通常使用“实体—关系”图建立数据模型。通常使用数据流图建立功能模型。通常使用状态转换图建立行为模型。

2．简述软件设计与需求分析的关系。

软件设计可看作将需求规格说明书逐步转换为软件源代码的中间过渡过程。

（1）活动输入：需求规格说明书。

（2）活动输出：设计规格说明书。

（3）活动目标：需求模型转换成设计模型。

（4）活动过程：层层分解，逐步求精。

3．简述面向对象方法学的要点。

1）认为客观世界是由各种对象组成的，任何事物都是对象，复杂的对象可以由比较简单的对象以某种方式组合而成。

（2）把所有对象都划分成各种对象类，每个对象类都定义了一组数据和一组方法。

（3）按照子类与父类的关系，把若干个对象类组成一个层次结构的系统。

（4）对象彼此之间仅能通过传递消息互相通信。

4. 简述面向对象分析的基本任务。

面向对象分析的基本任务是运用面向对象的方法，以了解基本的用户需求，对问题域和系统责任进行分析和理解。主要包括5个方面的工作：

（1）获取客户对系统的需求。

（2）标识类和对象。

（3）定义类的结构和层次。

（4）建造对象—关系模型。

（5）建立对象—行为模型。

5. 简述软件文档在软件工程中的作用。

1）提高软件开发的能见度；

（2）提高软件开发的效率；

（3）作为开发人员阶段工作成果和结束标志；

（4）记录开发过程的有关信息，便于软件的使用和维护；

（5）提供软件运行、维护和培训有关资料；

（6）便于用户了解软件的功能和性能。

6. 简述需求分析阶段的基本任务是什么,要进行哪几方面的工作。

需求分析的基本任务是要准确的定义新系统的目标。为了满足用户需要，回答系统必须“做什么”的问题。本阶段主要进行：问题识别，分析与综合，导出软件的逻辑模型，编写文档。

7．简述6条在软件设计过程中常用的启发规则。

（1）改进软件结构提高模块的独立性。

（2）模块规模应适中。

（3）深度、宽度、扇入和扇出都应适当。

（4）模块的作用域应该在控制域之内。

（5）力争降低模块接口的复杂程度。

（6）设计单入口单出口的模块。

（7）模块功能应该预测。

8．简述面向对象技术的主要优点。

（1）符合人类的思维习惯。面向对象技术的最重要的特点就是把事物的属性和操作组成一个整体，以对象为核心，更符合人类的思维习惯。

（2）可复用性好。面向对象技术由于采用了继承和多态的机制，极大地提高了代码的可复用性。

（3）稳定性好。基于对象的概念，当目标系统的需求发生变化时，只要实体和实体之间的联系不发生变化，就不会引起软件系统结构的变化，而只需对部分对象进行局部修改即可。

（4）可维护性好。由于利用面向对象技术实现的软件系统的稳定性好和可复用性好，而且采用了封装和信息隐藏的机制，易于对局部软件进行调整，因此系统的可维护性比较好。

9. 简述面向对象的设计准则。

模块化；抽象；信息隐藏；弱耦合；强内聚；可重用；

10. 简述提高软件可维护性的方法。

（1）建立明确的软件质量目标。

（2）利用先进的软件开发技术和工具。

（3）建立明确的质量保证工作。

（4）选择可维护的程序设计语言。

（5）改进程序的文档。

（6）注重可维护性的评审环节。

11. 简述软件生命存周期可分为几个阶段，每个阶段的提交物是什么。

（1）可行性研究和项目开发计划，提交项目开发计划和可行性分析报告。

（2）需求分析，提交软件需求说明书。

（3）概要设计，提交概要设计说明书。

（4）详细设计，提交详细设计说明书。

（5）编码，提交源程序清单。

（6）测试，提交测试报告。

（7）维护，提交维护报告。

12．简述软件测试应遵循的准则。

（1）所有的测试都应该能追溯到用户需求。

（2）应该在测试开始之前的相当长时间，就制定出测试计划。

（3）把Pareto原理应用于软件测试。

（4）测试应该从“小规模”开始，并逐步进行“大规模”测试。

（5）穷举测试是不可能的。

（6）为了达到最佳的测试效果，应该由独立的第三方来从事测试工作。

13．简述类与类之间的关系。

类与类之间最常见的关系是关联关系和泛化关系。

（1）关联关系：关联表示两个类的对象之间存在某种语义上的联系。关联关系包含普通关联和聚集。聚集表示类与类之间的关系是整体与部分的关系。

（2）泛化关系：就是通常所说的继承关系，是通用元素和具体元素之间的一种分类关系。泛化关系表示类与类之间存在一般与特殊的关系。

14. 简述面向对象的启发规则。

（1）设计结果应该清晰易懂。

（2）一般/特殊结构的深度应该适当。

（3）设计简单的类，因为简单的类更容易使用。

（4）使用简单的协议。一般来说，消息中的参数不要超过3个。

（5）使用简单的服务。

（6）把设计变动减至最小。

15. 简述整个软件测试过程中的步骤及其基本任务。

整个软件测试的过程包含四个步骤：单元测试、集成测试、确认测试、系统测试。

（1）单元测试：对源程序中的每个程序单元进行测试，检查各模块是否正确实现规定的功能，从而发现模块在编码中或算法中的错误。

（2）集成测试：检查与设计相关的软件结构的有关问题。

（3）确认测试：主要检查已实现的软件是否满足需求规格说明书中确定了的各种需求。

（4）系统测试：把已确定的软件与其他系统元素结合在一起进行测试。

16. 简述数据流图的作用。

数据流图可以用来抽象地表示系统或软件。从信息传递和加工的角度，它以图形的方式刻画数据流从输入到输出的移动变换过程，同时可以按自顶向下、逐步分解的方法表示内容不断增加的数据流和功能细节。因此，数据流图既提供了功能建模的机制，也提供了信息流建模的机制，从而可以建立起系统或软件的功能模型。

17．简述软件设计模块化的原因。

（1）可以将问题简化，划分模块可使每一个模块完成单一的功能。

（2）可以独立地进行模块的编码和测试，便于软件开发工作的组织。

（3）把每一个模块要解决的问题局限在有限的范围，减少出错的机会。

（4）便于纠错，便于对特定的模块进行优化处理。

（5）一个模块可被重复使用，以提高软件产品的复用率。

（6）程序易于理解，利于估计工作量和开发成本。

18．简述面向对象的特征。

（1）对象唯一性：每个对象都有自身唯一的标识，通过这种标识，可以找到相应的对象。

（2）分类性：分类性是指将具有一致性的数据结构和行为的对象抽象成类。

（3）继承性：继承性是子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。

（4）多态性：多态性是指相同的操作或函数、过程作用于多种类型的对象上并获得不同的结果。不同的对象收到同一消息可以产生不同的结果。

19. 简述设计简单的类的原因，以及如何设计简单的类。

经验表明，如果一个类的定义不超过一页纸（或两屏），则使用这个类是比较容易的。为了保持类的简单，应该注意以下几点：

（1）避免包含过多的属性。

（2）有明确的定义。为了使类的定义明确，分配给每个类的任务应该简单。

（3）尽量简化对象之间的合作关系。

（4）不要提供太多服务。

20. 简述软件配置管理的作用及其主要内容。

软件配置管理的作用是用于整个软件工程过程，其目的是协调整个软件的开发，使得混乱减少到最小，使得变更所产生的错误达到最小并最有效的提高生产率。软件配置管理的主要内容是：标识变更，控制变更，确保变更的正确实现，报告有关变更。

21.简述数据字典的作用。

分析模型中包含了对数据对象、功能和控制的表示。在每一种表示中，数据对象和控制项都扮演一定的角色。为表示每个数据对象和控制项的特性，建立了数据字典。数据字典精确地、严格地定义了每一个与系统相关的数据元素，并以字典式顺序将它们组织起来，使得用户和分析员对所有的输入、输出、存储成分和中间计算有共同的理解。

22．简述什么是逻辑覆盖？逻辑覆盖有哪几种不同的覆盖标准。

逻辑覆盖是对一系列测试过程的总称，这组测试过程逐渐进行越来越完整的通路测试。

逻辑覆盖的不同标准有：

（1）语句覆盖：选择足够多的测试数据，使被测试程序中每个语句至少执行一次。

（2）判定覆盖：不仅每个语句必须至少执行一次，而且每个判定的每种可能的结果都应该至少执行一次。

（3）条件覆盖：不仅每个语句必须至少执行一次，而且使判定表达式中的每个条件都取到各种可能的结果。

（4）判定/条件覆盖：选择足够多的测试数据，使得判定表达式中的每个条件都取到各种可能的值，而且每个判定表达式也都取到各种可能的结果。

（5）条件组合覆盖：选择足够多的测试数据，使得每个判定表达式中条件的各种可能组合都至少出现一次。

23．简述面向对象分析模型的内涵。

面向对象分析模型通常包括对象模型、动态模型和功能模型。对象模型是最基本、最重要、最核心的模型，描述软件系统的静态结构。动态模型描述系统的控制结构。功能模型描述系统必须完成的功能。这三种模型各自从不同的侧面反映软件系统的内容，相互影响、相互制约，有机地结合在一起，全面地表达对目标系统的需求。

24. 简述对基于面向对象思想的设计而言，如何实现信息隐藏。

通常，模块的信息隐藏可以通过接口来实现。模块通过接口与外部进行通信，而把模块的具体实现细节（如数据结构、算法等内部信息）隐藏起来。一般来说，一个模块具有有限个接口，外部模块通过调用相应的接口来实现对目标模块的操作。

25. 简述软件质量保证的措施。

软件质量保证的措施主要包括：基于非执行的测试（也称为复审）、基于执行的测试（即通常所说的软件测试）和程序正确性证明。

技术复审主要用来保证在编码之前各阶段产生的文档的质量。

基于执行的测试需要在程序编写出来之后进行，是保证软件质量的最后一道防线。

程序正确性证明使用数学方法来严格验证程序是否与它的说明完全一致。

26. 简述结构化需求分析方法的基本思想。

结构化需求分析是一种面向数据流的需求分析方法。它基于“分解”和“抽象”的基本思想，逐步建立目标系统的逻辑模型，进而描绘出满足用户要求的软件系统。

“分解”是指对于一个复杂系统，为了将复杂性降低到可以掌握的程度，可以把大问题分解为若干个小问题，然后分别解决。

最顶层描述了整个目标系统，中间层将目标系统划分为若干个模块，每个模块完成一定的功能，而最底层是对每个模块实现方法的细节描述。可见，在逐层分解的过程中，起初并不考虑细节性的问题，而是先关注问题最本质的属性，随着分解自顶向下进行，才逐渐考虑越来越具体的细节。这种用最本质的属性表示一个软件系统的方法就是“抽象”。

27．简述软件测试的目的。

（1）测试是一个为了发现程序中的错误为目的的程序执行过程。

（2）一个好的测试用例是能最大限度地找到迄今为止尚未发现的错误。

（3）成功的测试是能发现了那些尚未发现的错误。

28．简述建立对象模型的过程。

（1）确定类与对象。

（2）确定关联。

（3）划分主题。

（4）确定属性。

（5）识别继承关系。

（6）反复修改。

29. 简述良好的面向对象程序设计风格主要有哪些准则。

（1）提高可重用性：提高方法的内聚；减小方法的规模；保持方法的一致性；把策略和实现分开；全面覆盖；尽量不使用全局信息；利用继承机制。

（2）提高可扩充性：封装实现策略；不要用一个方法遍历多条关联链；避免使用多分支语句；精心确定共有方法。

（3）提高健壮性：预防用户的操作错误；检查参数的合法性；不要预先确定限制条件；先测试后优化。

30. 简述制定软件项目进度计划的两种方式。

制定软件项目进度计划有两种方式：Gantt图和工程网络图。

（1）Gantt图：用线段标明子任务的分解，标明各个任务的计划开始时间、完成时间，标明各个任务与参与工作的人数，各子任务间的并行和串行的关系的衔接情况。

（2）工程网络图：是制定进度计划的一种常用的图形工具，能描绘任务分解情况以及每项作业的开始时间和结束时间，能显式地描绘各个作业彼此间的依赖关系，是系统分析和系统设计的有力工具。