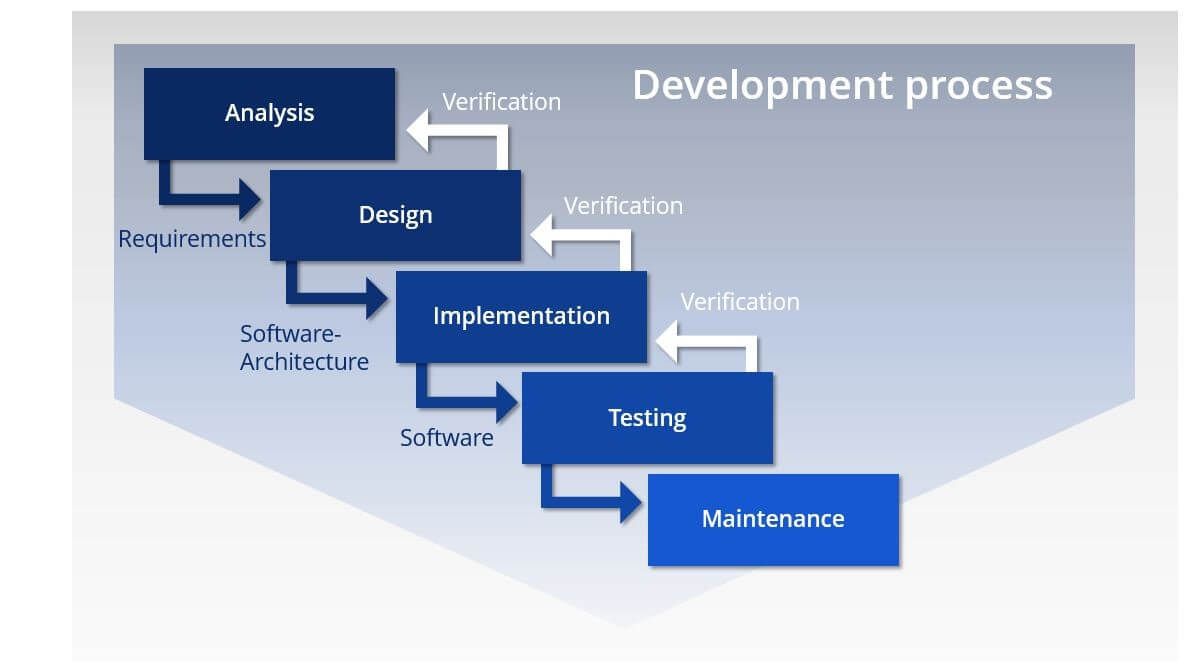
**阅读并参考课件，完成以下问题：**

**1.解释和比较以下软件过程模型：瀑布、增量、迭代、敏捷、RUP**

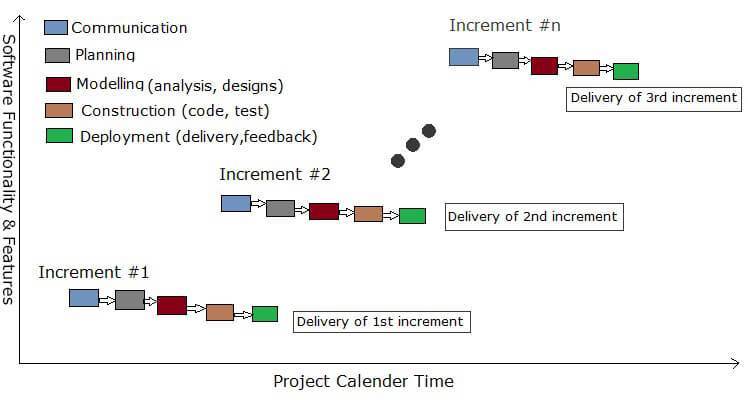
**- 瀑布模型**

瀑布模型核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，即采用结构化的分析与设计方法将逻辑实现与物理实现分开。将软件生命周期划分为制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落。



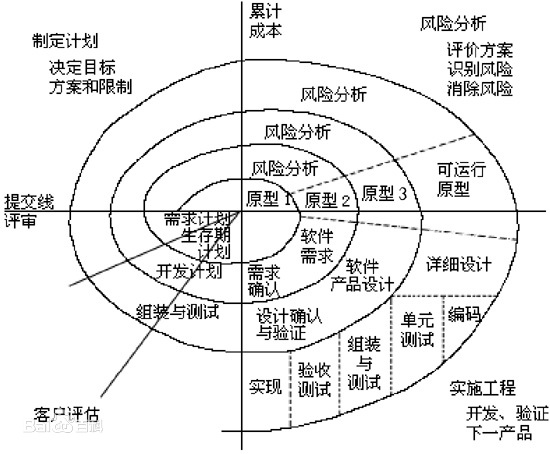
**- 增量模型**

增量模型是把待开发的软件系统模块化，将每个模块作为一个增量组件，从而分批次地分析、设计、编码和测试这些增量组件。运用增量模型的软件开发过程是递增式的过程。相对于瀑布模型而言，采用增量模型进行开发，开发人员不需要一次性地把整个软件产品提交给用户，而是可以分批次进行提交。



**- 迭代模型**

迭代模型是RUP推荐的周期模型。被定义为：迭代包括产生产品发布（稳定、可执行的产品版本）的全部开发活动和要使用该发布必需的所有其他外围元素。在某种程度上，开发迭代是一次完整地经过所有工作流程的过程：需求分析、设计、实施和测试工作流程。实质上，它类似小型的瀑布式项目。RUP认为，所有的阶段都可以细分为迭代。每一次的迭代都会产生一个可以发布的产品，这个产品是最终产品的一个子集。

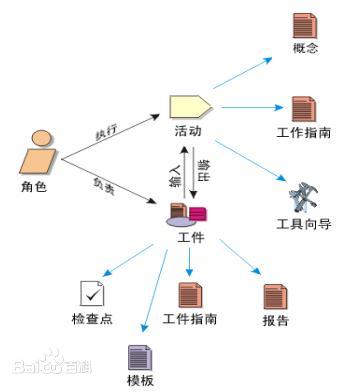


**- 敏捷开发**

敏捷开发是一种能应对快速变化需求的软件开发能力。它们的具体名称、理念、过程、术语都不尽相同，相对于“非敏捷”，更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本、紧凑而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重软件开发过程中人的作用。



**- RUP：**RUP（Rational Unified Process），统一软件过程开发，统一软件过程是一个面向对象且基于网络的程序开发方法论。软件统一过程（RUP）是Rational软件公司（Rational公司被IBM并购）创造的软件工程方法。RUP描述了如何有效地利用商业的可靠的方法开发和部署软件，是一种重量级过程（也被称作厚方法学），因此特别适用于大型软件团队开发大型项目。



**2.解释软件体系结构设计的一些重要原则：稳定性、可扩展性、可复用、可修改、可移植、可伸缩、可维护等**

**- 稳定性：**意味着即使发生故障，也能使系统正常工作。故障可以是硬件,软件，以及人类。容错技术可以隐藏最终用户的某些类型的故障。  
**- 可扩展性**：即使在负载增加的情况下也有保持性能的策略。  
**- 可复用**：设计的部件可使用于多个软件，这样可以加快开发效率。  
**- 可修改**：面对需求的变化，软件也需要进行修改、更新。  
**- 可移植**：对于不同的软件、硬件平台，软件能够在不同的平台运行。良好的可移植性可以降低维护成本。  
**- 可伸缩**：可伸缩性是一-种对软件系统计算处理能力的设计指标，高可伸缩性代表-种弹性， 在系统扩展成长过程中，软件能够保证旺盛的生命力，通过很少的改动甚至只是硬件设备的添置，就能实现整个系统处理能力的线性增长，实现高吞吐量和低延迟高性能。  
**- 可维护**：为软件的维护团队提供便利，如使用良好的抽象来降低软件的复杂性，使系统更易于修改和适应新的用例等。

**3.了解设计模式的概念，用java代码展示其中的一到两种设计模式的编程框架（如：观察者模式）**

**- 定义**

设计模式（Design Pattern）是前辈们对代码开发经验的总结，是解决特定问题的一系列套路。它不是语法规定，而是一套用来提高代码可复用性、可维护性、可读性、稳健性以及安全性的解决方案。

**- 根据目的来分**

**创建型模式**：用于描述“怎样创建对象”，它的主要特点是“将对象的创建与使用分离”。GoF 中提供了单例、原型、工厂方法、抽象工厂、建造者等 5 种创建型模式。

**结构型模式**：用于描述如何将类或对象按某种布局组成更大的结构，GoF 中提供了代理、适配器、桥接、装饰、外观、享元、组合等 7 种结构型模式。

**行为型模式**：用于描述类或对象之间怎样相互协作共同完成单个对象都无法单独完成的任务，以及怎样分配职责。GoF 中提供了模板方法、策略、命令、职责链、状态、观察者、中介者、迭代器、访问者、备忘录、解释器等 11 种行为型模式。

**- 根据作用范围来分**

**类模式**：用于处理类与子类之间的关系，这些关系通过继承来建立，是静态的，在编译时刻便确定下来了。GoF中的工厂方法、（类）适配器、模板方法、解释器属于该模式。

**对象模式**：用于处理对象之间的关系，这些关系可以通过组合或聚合来实现，在运行时刻是可以变化的，更具动态性。GoF 中除了以上 4 种，其他的都是对象模式。

**- 单例（Singleton）模式**

某个类只能生成一个实例，该类提供了一个全局访问点供外部获取该实例，其拓展是有限多例模式。

**- 懒汉式单例**

|  |
| --- |
| public class LazySingleton  {  private static volatile LazySingleton instance=null; //保证 instance 在所有线程中同步  private LazySingleton(){} //private 避免类在外部被实例化  public static synchronized LazySingleton getInstance()  {  //getInstance 方法前加同步  if(instance==null)  {  instance=new LazySingleton();  }  return instance;  }  } |

**- 饿汉式单例**

|  |
| --- |
| public class HungrySingleton  {  private static final HungrySingleton instance=new HungrySingleton();  private HungrySingleton(){}  public static HungrySingleton getInstance()  {  return instance;  }  } |

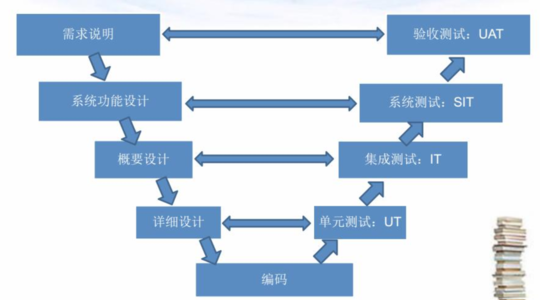
**- 原型模式**

用一个已经创建的实例作为原型，通过复制该原型对象来创建一个和原型相同或相似的新对象。在这里，原型实例指定了要创建的对象的种类。用这种方式创建对象非常高效，根本无须知道对象创建的细节。

|  |
| --- |
| //具体原型类  class Realizetype implements Cloneable  {  Realizetype()  {  System.out.println("具体原型创建成功！");  }  public Object clone() throws CloneNotSupportedException  {  System.out.println("具体原型复制成功！");  return (Realizetype)super.clone();  }  }  //原型模式的测试类  public class PrototypeTest  {  public static void main(String[] args)throws CloneNotSupportedException  {  Realizetype obj1=new Realizetype();  Realizetype obj2=(Realizetype)obj1.clone();  System.out.println("obj1==obj2?"+(obj1==obj2));  }  } |

**4.什么是软件测试的V模型？请解释说明之。**

RAD（Rapid Application Development，快速应用开发）模型是软件开发过程中的一个重要模型，由于其模型构图形似字母V，所以又称软件测试的V模型。V模型大体可以划分为以下几个不同的阶段步骤：需求分析、概要设计、详细设计、软件编码、单元测试、集成测试、系统测试、验收测试。



**5.调研中-大型软件公司中的职位名称（头衔）和对应的要求，并说明你的兴趣和方向。**



**6.什么是计算机辅助软件工程（CASE）？**

计算机辅助软件工程(Computer Aided Software Engineering，CASE)原来是指用来支持MIS开发的、由各种计算机辅助软件和工具组成的一个大型综合性软件开发环境，随着各种工具及软件技术的发展、完善和不断集成，逐步由单纯的辅助开发工具环境转化为一种相对独立的方法。

**CASE的功能**

一般情况下，CASE工具应陔具有以下几个功能。

①用户通过CASE丁具能创建软件开发各阶段所需的图表。

②收集有关图表上的对象以及对象之间关系的信息，以便建立一个完整的信息集合。

③在一个中央资源库中，应将图表所表示的语义而不是图标本身存储起来。

④根据准确性、一致性、完整性检查图表。

⑤使用户能以图表来描述条件、循环、CASE结构和其他结构化程序结构。

⑥使用户能以多种图表类型表示一个分析或设计的不同方面。

⑦实施结构化的模型和设计，尽可能达到准确和一致。

⑧协调多个图表上的信息，检查信息的一致性，并集中检查信息的准确性、一致性和完整性。

**CASE的作用**

归纳起来，CASE有三大作用，这些作用从根本上改变了软件系统的开发方式。CASE的作用如下所示。

①一个具有快速响应、专用资源和早期查错功能的交互式开发环境。

②对软件的开发和维护过程中的许多环节实现了自动化。

③通过一个强有力的图形接口。实现了直观的程序设计。