1. 软件工程学概述
2. **软件的特点**
   1. 是逻辑部件，不是物理部件
   2. 缺乏可见性
   3. 管理和控制生产过程相当困难
   4. 维护困难和复杂
   5. 生产只需复制
   6. 软件开发尚未完全摆脱手工的开发方式
   7. 软件开发复杂，费用较高且其性质如成本、进度等难以估计
   8. 软件的开发和运行受计算机系统的限制，而开发更加依赖于开发人员的业务素质、智力、人员的合作、组织和管理
3. **软件危机**：计算机软件的开发和维护过程中所遇到的一系列严重问题

（如何开发软件，已满足日益增长的需求；如何维护数量不断膨胀的已有软件）

1. **软件工程**：是从管理和技术两个方面指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，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，以经济地开发出高质量的软件(基本目标) 并有效地维护。
2. **方法学**：指在软件生命周期全过程中使用的一整套技术的集合，也称为范型。（三个要素：方法、工具、过程）
3. **软件生命周期**：（3大时期、8大阶段）
   1. 软件定义
      1. 问题定义
      2. 可行性研究
      3. 需求分析：产出规格说明文档。
   2. 软件开发
      1. 概要设计
      2. 详细设计
      3. 编码和单元测试
      4. 综合测试（集成测试、验收测试）
   3. 软件维护/运行维护
      1. 软件维护（改正性维护、适应性维护、完善性维护、预防性维护）
4. **软件过程**：是为获得高质量软件所需完成的一系列任务的框架，他规定了完成各项任务的工作步骤。
5. **软件文档**：是指的是一些记录的数据和数据媒体，它具有固定不变的形式，可被任何计算机阅读。在软件工程中，文档常常用来表示对活动、需求、过程或结果进行描述、定义、规定或认证的任何书面或图示的信息，他们描述和规定了软件设计和实现细节，说明使用软件的操作命令。
6. 可行性研究
   * + 1. **可行性研究**：可行性研究是在较高层次上以较抽象的方式进行设计的过程
7. 需求分析
   * + 1. 确定对系统的综合要求： 1 、功能需求； 2 、性能需求； 3 、可靠性和可用性需求； 4 、出错处理需求； 5 、接口需求； 6 、约束； 7 、逆向需求； 8 、将来可以提出的要求，分析系统的数据要求。
8. 总体设计
9. 详细设计
   * + 1. **模块化**：把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能，把这些模块集成起来构成一个集体，可以完成指定的功能满足用户的需求。
       2. **模块独立性**：是模块化、抽象化、信息隐蔽概念的一个直接产物。它是好设计的关键，而设计又是决定软件质量的关键环节。
       3. **耦合**：不同模块之间互联程度的度量。
       4. **内聚**：模块内各个元素彼此结合的紧密程度。
10. 实现
    * + 1. **结构程序设计**：程序只有一个入口和出口。是尽可能少用GOTO语句的程序设计方法。最好仅在检测出错误时才使用GOTO语句，而且应该总是使用前向GOTO语句。（基本结构：顺序、循环、选择）
        2. **人机界面设计的四个方面**
           1. 系统响应时间
           2. 用户帮助设施
           3. 出错信息处理
           4. 命令交互
11. 维护
    * + 1. **编码风格**：包括程序内部良好的文档（恰当的标识符、适当的注释）、数据说明、语句构造、输入输出规则、效率保证
        2. **测试**：是为了发现程序中错误而执行程序的过程
        3. **测试目的**：竭力证明程序中有错误不能按照预定要求正确工作
        4. **测试方法**：
           1. 白盒测试：把程序看成装在一个透明的白盒子，测试者完全知道程序的结构和处理算法。这种方法按照程序内部的逻辑测试程序，检测程序中主要执行通路是否都能按预定要求正确工作。白盒测试又称为结构测试。（在测试过程早期使用）
           2. 黑盒测试：把程序看作一个黑盒子，完全不考虑程序的内部结构和处理过程，黑盒测试是在程序接口进行的测试。

检查程序功能是否能按照规格说明书的规定正常使用

程序是否能适当地接收输入数据产生正确的输出信息

程序运行过程中能否保持外部信息

黑盒测试又称为功能测试。（在测试过程后期使用）

* + - 1. **测试用例**：一组用于测试的输入数据和预期得出的正确输出
      2. **测试方案**：测试用例和用例预定要检验的功能
      3. **测试步骤**：
         1. 模块测试
         2. 子系统测试（注重接口）
         3. 系统测试（检测+组装，集成测试）
         4. 验收测试（用户参与）
         5. 平行运行（比较新旧系统的处理结果）
      4. **调试**：是在测试发现错误之后排除错误的过程，是把症状和原因联系起来的尚未被人深入认识的智力过程。
      5. **软件可靠性**：是程序在给定的时间间隔内，按照规格说明书的规定成功地运行的概率。
      6. **软件可用性**：程序在给定的时间点，按照说明书的规定，成功地运行的概率。

1. 维护
   * + 1. **软件维护**：在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程。
       2. **软件可维护性**：维护人员理解、改正、改动和改进这个软件的难易程度。
2. 面向对象方法学
   * + 1. **面向对象方法**：是一种以数据和信息为主线，把数据和处理相结合的方法
       2. **类**：是对具有相同数据和相同操作的一组相似对象的定义。
       3. **对象**：既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有明确边界和意义的东西。
       4. **继承**：是子类自动地共享基类中定义的数据和方法的机制。
       5. **类的多态性**：向不同的对象发送同一条消息，不同的对象在接收时会产生不同的行为。（所谓消息，就是调用函数，不同的行为就是指不同的实现，即执行不同的函数）
       6. **重载**：是指在同一作用域内的若干个参数特征不同的函数可以使用相同的函数名字
       7. **UML**：统一建模语言，是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持。
       8. **对象模型**：是对模拟客观世界实体的对象以及对象彼此间的关系的映射。
3. UML与面向对象的分析与设计
4. 软件项目管理
   * + 1. **项目管理**：就是通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程
       2. **项目计划**：是根据对未来的项目决策，项目执行机构选择制定包括项目目标、工程标准、项目预算、实施程序及实施方案等的活动。
       3. **软件质量**：软件与明确地和隐含地定义的需求相一致的程度。
       4. **软件配置**：软件配置项的统称。
       5. **软件配置项**：软件生命周期各个阶段活动的产物经审批后即可称之为软件配置项。
          1. 计算机程序(源代码和可执行程序);
          2. 描述计算机程序的文档(供技术人员或用户使用)；
          3. 数据(程序内包含的或在程序外的)。
       6. **基线**：通过了正式复审的软件配置项。
       7. **质量因素**：

