1. 入门 结构化编程 面向对象，数据和操作数据的函数组成独立模块

2、方法学 描述 产品内容 文档 形式（规范和目录，图标）

Rup xp极限编程

1. 对象 是对问题涉及的事物的抽象， 粒度可以由自己掌握

面向对象研究的不是问题，而是事物，**设计重于代码，接口大于实现**

对象 属性attribute 行为behavior

一切都可以是对象，从数据类型 数字，字符，到日期，文件，窗口，游戏人物，设计模式都可以是对象

类和类型的关系

面向对象要接近现实的存在，但是不可能完美模仿现实

为业务建模，模型model 模型是对现实的模拟， 数学模型，人体模型

对象的相等和相同，数据相同，但是互相独立就不是同一个对象

Uml表示对象 对象名 属性 操作（参数） 属性通过操作访问

对象间的互相通信，对象间的关系

封装：将属性隐藏起来，设置为私有属性，只能用方法来访问和修改，防止合作编程中互相干扰

**对象可以作为其他对象的属性使用，或者在方法中生成作为局部变量，也可以用参数传入**

关联： 关联的其他对象 可以去除

聚合：对象放一起组成更大的对象 部分整体（不能去除）

对象树

对象图

对象之间连接，可以作为属性

对象通信：消息发送接收

用自己的方法 获取或操作其他对象（来自参数，属性，局变，全局/外部）的属性， 调用其他对象的方法来修改其属性或者得到返回值（调用时可以传入包含信息的参数）

消息发送者，接收者，消息处理 也可以对自己发消息，获取自己的属性（this）

让接收对象执行操作（可以带上参数），或者从接收对象提取信息

Get set

创建对象——连接对象——彼此发送消息协作

垃圾收集，对象创建了就需要回收，一般是引用计数，所以要注意消除引用，如果可能的话应该设置管理模块，

类 工厂 集合 模板 字典

类是对象的模板，对象是类的实例 instance=对象

分类，接近真实世界

类的关系:继承 父子 整体-部分

字段:属性 构造函数 消息get set

静态方法和静态属性:一个类所有对象的共享数据和操作

对象行为:指所有的方法

接口，对象协议:指所有的方法调用

属性=变量=特性=数据成员

调用=消息=执行 get set

聚合=复合=关联=委托

重用:重复多次使用代码

避免把属性和操作分散

重用实现:公共函数调用，方法调用,对象全局 类库，函数库

设计模式，框架

样式规则，

完善的文档说明:有意义的类名 类的总结 分段注释 如何使用（参数，返回）

每个对象只做一件事:高聚合

接口行为分离

对象之间不应该双向通信,客户—提供者层次结构

消息可能是问题:get 或者命令set等

1. 继承

从父类获取 特性，添加自己的特性（属性） 继承所有字段 消息 方法 断言

集合类：可以包含其他对象，<T>, 集合如列表，向量的成员可以是对象，字符串，包装对象 链表 背包

消息:是提供给外部的接口（函数） 有些方法可能是私有方法，只能由接口调用

未完成的消息：抽象方法 UML中抽象是斜体

抽象类：至少一个抽象方法 抽象类不应该建立实例，应该被继承实现

方法的重定义:抽象变具体，实现方法 完成额外工作

继承和复合：复合引入了其他对象 可能要new其他类

多重继承 接近真实，允许混合继承mixin

1. 类型系统 禁止误用值（拼错，用错） 静态类型，动态

性能高，可以检查拼写错误

多态 多态变量 值在不同时候表示不同类型 多态方法:多个方法和对象相关

变量多态由继承控制 父类型变量可以关联子类对象

多态消息,消息可以关联多个方法 消息指接口调用 由继承控制，调用子类方法，但是不能调用父类不存在的消息

类型转换， 当进行相加等活动，合成新的值时 参数传入不同类型时

隐式 显式

**对象指针的转换 ，传入不同类型，强制转为不同类型()**

**传参过程隐式向上转型，然后取出值的时候类型就可能变化，无法直接获取，需要强制向下转型()**

**向上转型**

**向下转型**

利用 模板泛型技术，比转型好

4、动态绑定

在超类链找方法

依赖

1. 软件开发方法学

编码不是最重要的部分 代码需要规则 概念错误 实践错误

扩展性（容易 修改） 可靠性（可用来解决其他问题） 易调试（）

对边界的考虑，对if else的测试 ，每一步证明自己的代码没问题（这是在设计阶段就应该想好的）

文档说明，每阶段都有

交付，交流，可重复

规划：做什么 调度:时间确定 分配资源 工作流 制品

需求分析 业务建模

分析 相关实体（数据 对象） 属性和互相关系

设计 如何解决问题 子系统设计（分解成模块），这部分用文字/图完成

规范

实现

测试

软件工程和瀑布流方法 每个阶段都顺利完成，这是不可能的

螺旋方法学 :大致设计，前期工作没完成就可以写代码，写完就测试，展示。。然后和客户讨论，更新需求。。。继续开发

迭代方法学，允许回退，来回移动

递增方法学 基本功能，增加功能，交付1.1 2.0 要求前面的代码可重用

面向对象的方法学:uml rup xp

软件原型:对产品部分功能的试验

Xp：配对编程，测试驱动的开发（在开发前进行，不断测试）

需要观察的：跟踪信息 编写顺序 修改历史 用户访问权限 减少冗余文件，在多处更新

UML 图 用例 类图 对象图 活动图（类似流程） 状态机图（对象的各种状态）

通信图（对象发送的消息） 顺序图 包图 时间图 复合结构图

1. 需求

理解业务（所有问题组成的集合） 业务建模 运作模式，参与者，入口点

打开程序，登录

参与者和用例的关系

用例（参与者进行的操作? 行为） 修改密码

用例关系 特殊化（继承） 前提条件 后置条件

**用户界面草案**

用例优先级

1. 问题分析

静态分析 动态分析 通信

将需求转化为 对象模型，对象的属性和关系

类 系统类 参与者类 边界（一般指用户界面，连接参与者和系统）

实体

类关系 继承 关联 聚合 复合（一同消亡）

类操作，职责，状态（可能比较复杂的状态（属性有多个值，而且随着外部的消息改变） ，比如战斗状态）

1. 设计系统
2. 技术选择

客户层技术 网页还是客户端 平台 兼容性

中间层协议 http tcp/ip ssl rmi

中间层技术（服务器）

数据层

1. 子系统

十一、设计模式

十二、类接口

十三、不间断测试