一、面向对象基础与术语 *00:01*

- 面向对象定义：一种程序设计思想，将数据与操作封装成对象
- 常见误解：误认为面向对象是编程语言的语法特性
- 语言无关性：可在不支持面向对象语法的语言(如C语言)中实现
- 核心特征：围绕对象进行程序设计，强调数据与行为的封装
- 与面向过程区别：C语言是面向过程，C++支持面向对象
- 学习建议：避免死记硬背语法，应理解设计思想本质

1.类和对象 01:12

- 类定义：类是对象的模板或蓝图，描述对象的结构和功能。
- 类组成：类包含属性（如年龄、姓名）和行为方法（如吃、睡）。
- 对象生成：通过类定义代码生成具体对象（如小明、小刚）。

- 方法调用：通过对象调用封装的行为方法（如小明吃饭、小刚睡觉）。
- 类与实例关系：类是抽象描述，实例是具体占用内存的对象。
- 实例特性：实例拥有类定义的属性和方法，是独立的程序对象。

2.构造方法和析构方法 02:05

- 构造方法定义：在对象创建时自动调用的特殊方法，用于初始化对象成员和资源分配。
- 构造方法特点：自动调用、与类同名、无返回值、支持重载。
- 析构方法定义：在对象销毁时自动调用的特殊方法，用于资源释放（如内存、文件句柄）。
- 析构方法特点：自动调用、类名前加~、无返回值且无参数、不可重载（唯一性）。
- 默认生成机制：未显式定义时，编译器自动生成构造/析构函数。
- 生命周期关联：构造函数对应对象创建阶段，析构函数对应销毁阶段。

3.属性修饰符 03:43

- 修饰符定义：控制类成员可见性的语法元素
- 核心作用：区分公开/私有成员，防止外部误修改
- 安全说明：仅限制编码时的语法访问，非运行时安全机制
- 私有字段：用private声明，类外不可直接访问
- 访问接口：通过get/set方法提供可控的读写通道
- 内存本质：修饰符不改变数据存储位置，仍可通过特殊方式修改
- 典型应用：公有属性允许任意修改，私有属性需通过方法操作

4.封装 04:52

- 封装定义：将数据和对应行为逻辑放入同一类中，并提供访问接口。
- 封装目的：通过属性私有化和统一接口控制数据访问，减少人为失误。
- 封装优势：集中管理属性访问路径，便于调试和追踪问题源头。
- 封装必要性：复杂对象操作时能规范行为逻辑，确保数据一致性。
- 工程价值：将运行时问题提前至编译时检查，提升代码健壮性。

5.继承 06:05

- 继承定义：类之间通过继承机制共享属性和方法的关系。
- 继承特征：子类可重写父类逻辑并扩展新属性（如Student类扩展学号/校名）。
- 访问权限：protected修饰符允许子类访问父类属性，外部不可访问。
- 继承关系：体现"is a"关系（如Student is a Human），且父子关系不可逆。
- 方法调用：子类实例可调用自身及父类方法（如Student调用Human方法）。
- 抽象层级：继承通过逐级细化抽象实现更精确的子类描述。

6.多态 07:04

- 多态目的:保持接口一致性，在继承关系中为同一方法提供不同逻辑
- 多态机制:运行时自动调用子类重写逻辑，无需手动检查类型
- 重写定义:子类重新实现父类虚函数，实现运行时动态绑定
- 重载定义:同一作用域内同名函数参数列表不同，实现编译时静态绑定
- 重载特征:函数名相同，参数类型/数量/顺序不同
- 绑定方式:重载是静态绑定(编译时)，重写是动态绑定(运行时)
- 重载限制:仅参数列表差异，不能仅靠返回类型区分

7.封装、继承、多态总结 08:19

- 重写定义:子类重新定义父类虚函数，修改实现以实现动态多态。
- 重写特征:函数签名（名称、参数类型/数量）必须与父类虚函数完全相同。

- 动态绑定:重写函数在运行时决定调用，属于动态行为。
- 重写示例:Student类重写Human类的c方法，提供自身逻辑。
- 重载vs重写:关键区分点为函数签名差异（参数列表是否相同）及绑定时机（编译期静态/运行期动态）。

8.抽象类和接口 08:56

- 抽象类定义：包含至少一个纯虚函数的类，无法直接实例化。
- 纯虚函数特征：仅在基类中声明，无实现，需在派生类中重写。
- 抽象类作用：提供接口约定，强制派生类实现具体功能（如载具类与汽车类示例）。
- 接口实现方式：C++中通过纯虚类（仅含纯虚函数）模拟（如打印接口和扫描接口）。
- 接口本质：规范约定，实现类必须提供接口要求的功能（如多功能打印机示例）。
- 抽象类与接口区别：抽象类可含非纯虚成员，接口仅定义行为规范。
- 面向对象核心：思想重于语法，用于工程规范而非代码累赘。