

面向对象程序设计概念

孙聪

网络与信息安全学院

2020-09-07

课程内容

- Java概述
- 面向对象程序设计概念
- Java语言基础
- Java面向对象特性
- Java高级特征
- 容器类
- 常用预定义类
- 异常处理
- 输入输出
- 线程

提要

1 面向对象程序设计的定义

2 对象和类

3 封装，继承和多态

提要

1 面向对象程序设计的定义

2 对象和类

3 封装，继承和多态

“我们之所以将自然界分解、组织成各种概念，并按其含义分类，主要是因为我们是整个口语交流社会共同遵守的协定的参与者，这个协定以语言的形式固定下来... 除非赞成这个协定中规定的有关语言信息的组织和分类，否者我们根本无法交谈”

—— Benjamin Lee Whorf (1897–1941)

为什么程序难写？

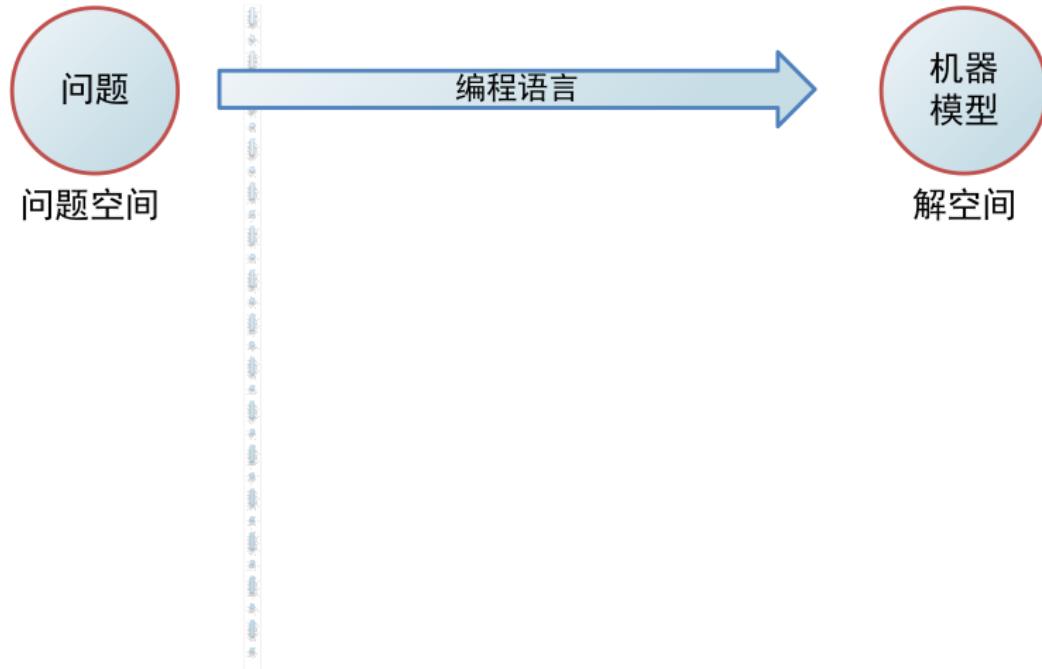


问题空间

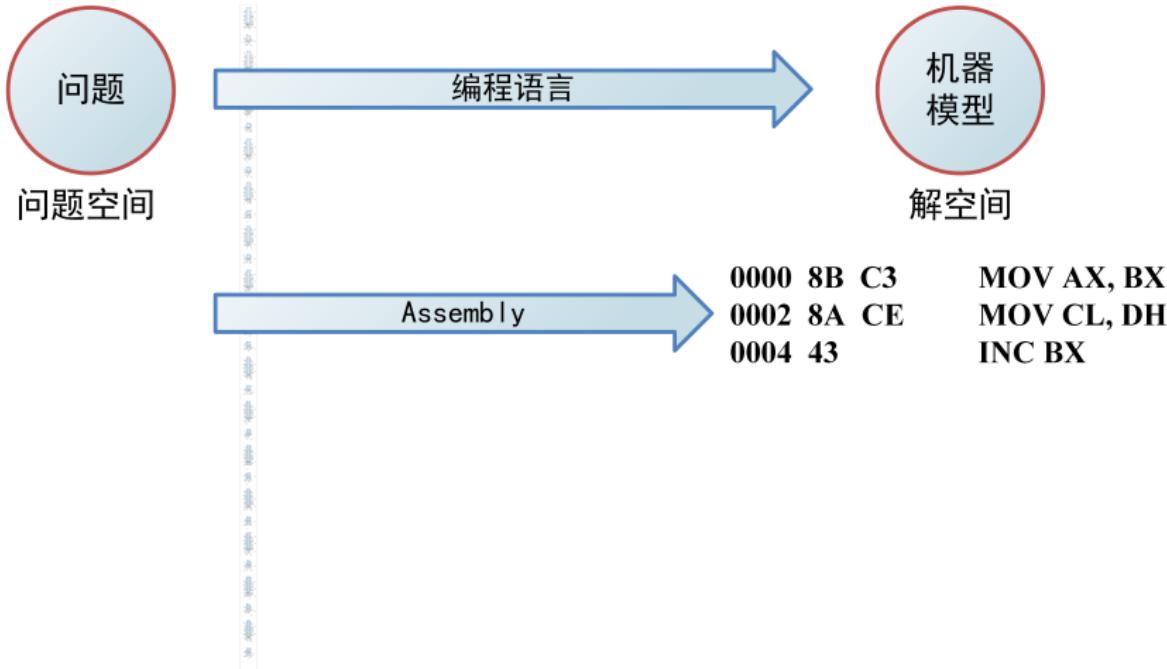


解空间

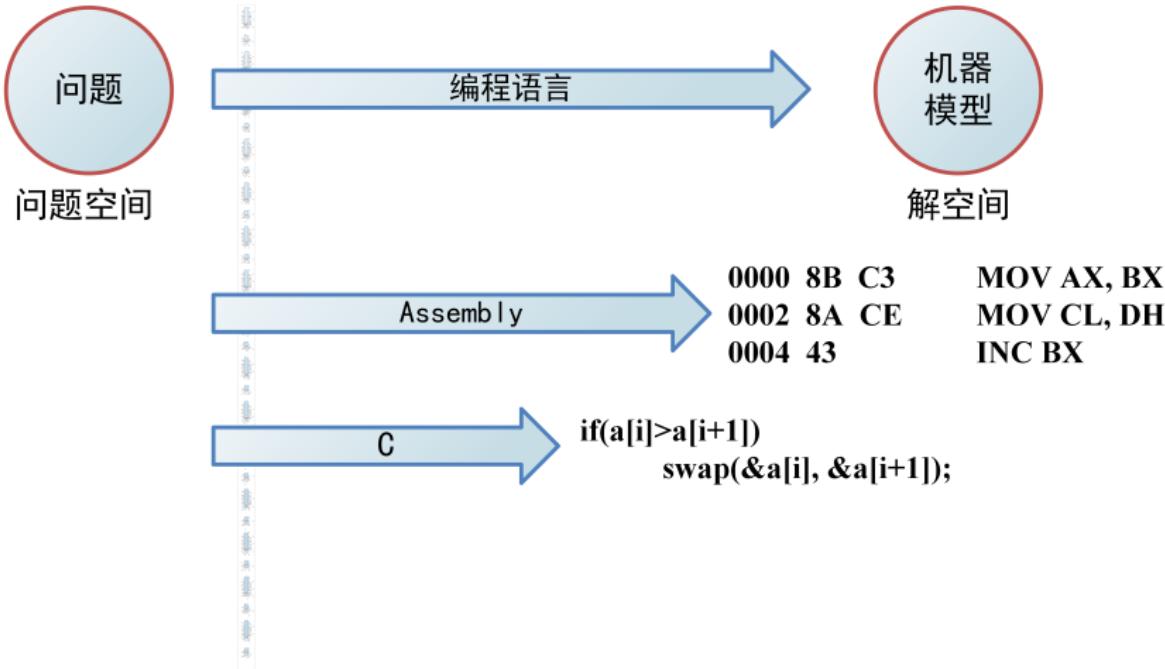
为什么程序难写？



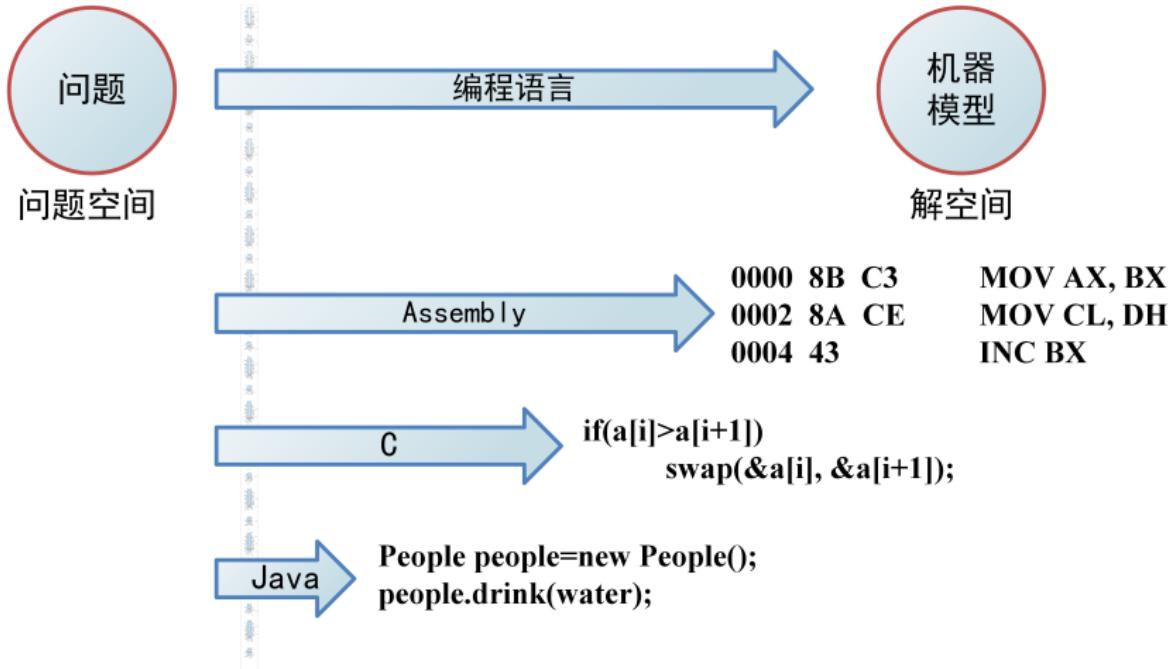
为什么程序难写?



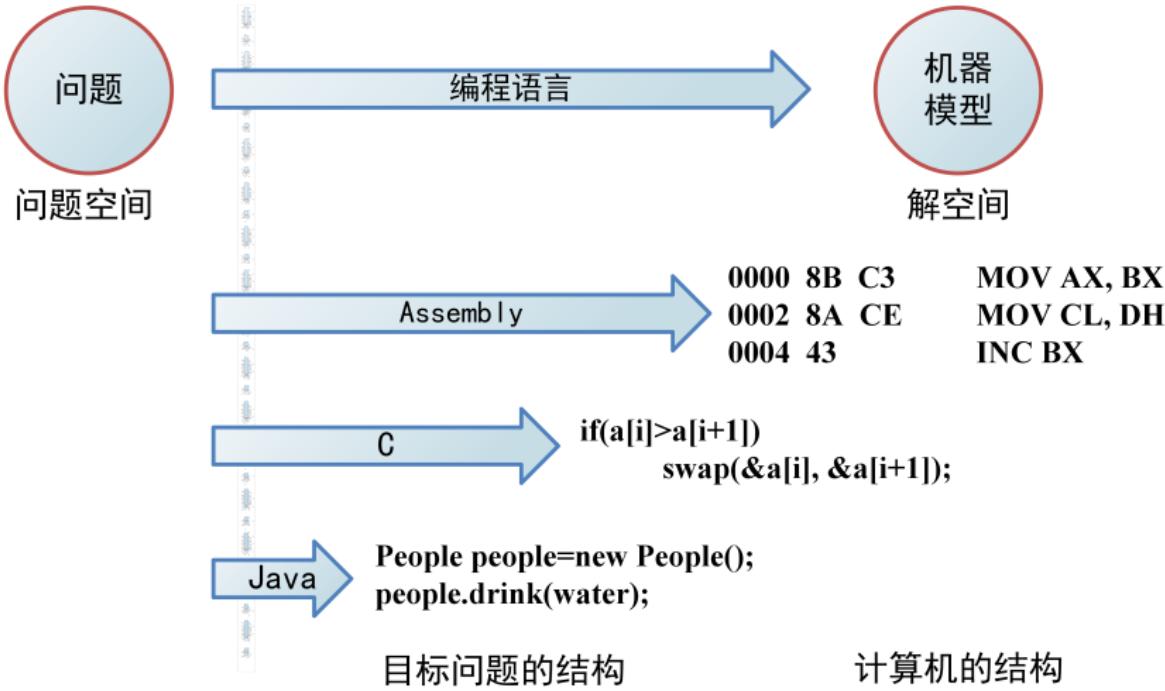
为什么程序难写?



为什么程序难写?



为什么程序难写？



面向对象程序设计的定义

Object-Oriented Programming (OOP) is a **programming paradigm** using “**objects**” – **data structures** consisting of **data fields** and **methods** together with their interactions – to design computer programs.

- 面向对象程序设计是一种基于“对象”概念的编程范式
- 对象是由数据字段、方法组成的数据结构
- 程序被设计为对象及对象之间的交互

面向对象程序设计的原则

Alan Kay的OOP五大原则

- 一切皆对象
- 程序是一系列对象的组合，对象间通过消息传递进行联系
- 每个对象都有自身内存空间，可容纳其他对象
- 每个对象都有一种类型
- 同一类型的所有对象都能够接收相同的消息

面向对象程序设计的原则

Alan Kay的OOP五大原则

- 一切皆对象
- 程序是一系列对象的组合，对象间通过消息传递进行联系
- 每个对象都有自身内存空间，可容纳其他对象
- 每个对象都有一种类型
- 同一类型的所有对象都能够接收相同的消息

面向对象方法的核心概念

抽象、对象、类、接口、封装、继承和多态

提要

1 面向对象程序设计的定义

2 对象和类

3 封装，继承和多态

对象

定义

- 问题空间中可以明确标识的实体
- 问题空间的实体或概念在解空间中的抽象表示

对象

定义

- 问题空间中可以明确标识的实体
- 问题空间的实体或概念在解空间中的抽象表示

对象的组成

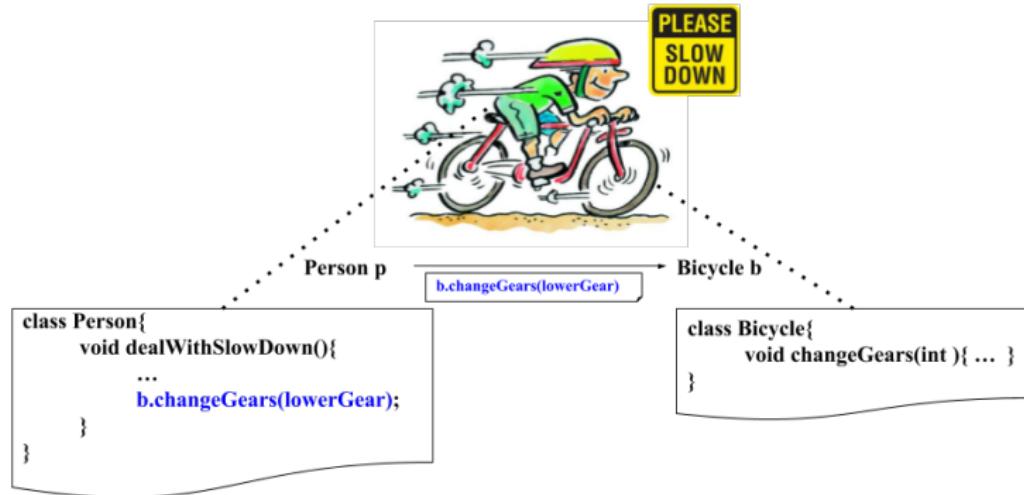
- 状态(特征、数据、属性): 具有当前值的数据域 (Java: 成员变量)
- 行为(动作、操作、服务): 用于设置或改变对象的状态, 或对外提供一种服务 (Java: 方法)

状态与行为之间的关系

- 对内: 方法能操作(访问/修改)成员变量
- 对外: 方法是对象与外部环境(其他对象)交互、通信的接口

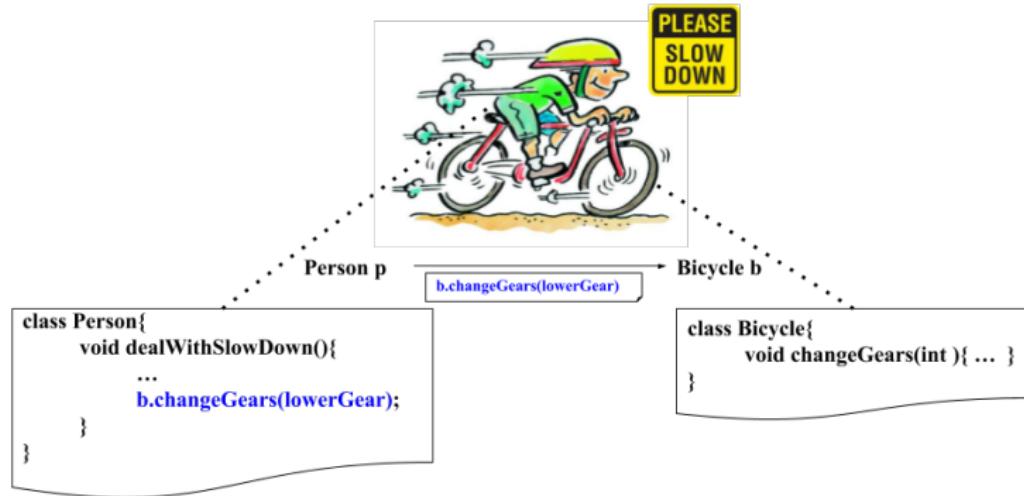
消息：对象之间交互与通信的手段

- 实现方式：消息通过对方法的调用来实现



消息：对象之间交互与通信的手段

- 实现方式：消息通过对方法的调用来实现

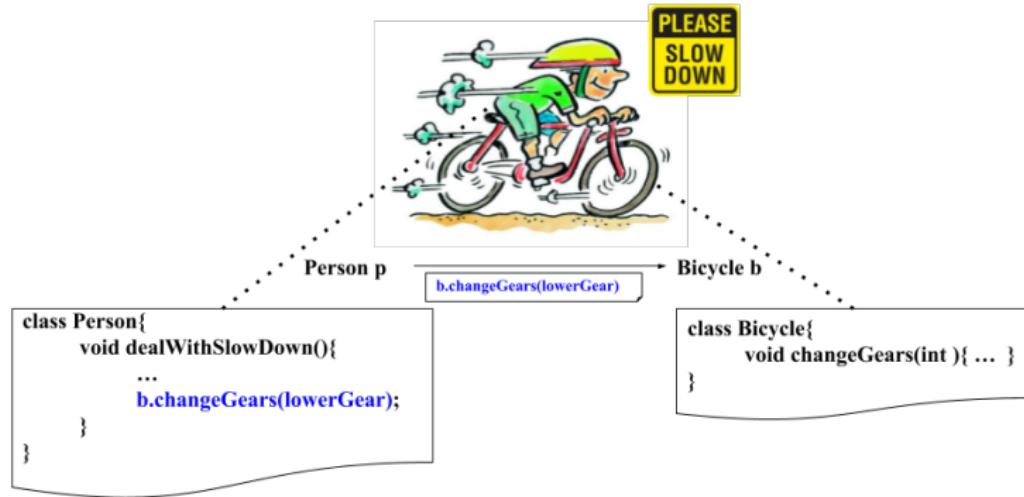


- 消息的组成

- 接收消息的对象
- 方法名称
- 方法参数

消息：对象之间交互与通信的手段

- 实现方式：消息通过对方法的调用来实现



- 消息由发送者对象编写，由接收者对象解释(定义)
- 消息可以：(1)引起接收者状态变化；(2)返回结果

类

In OOP, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

- 在面向对象程序设计中，类是一个可扩展的程序代码模板，用于创建一类对象并为对象提供成员变量初始值和方法实现

类与对象的关系

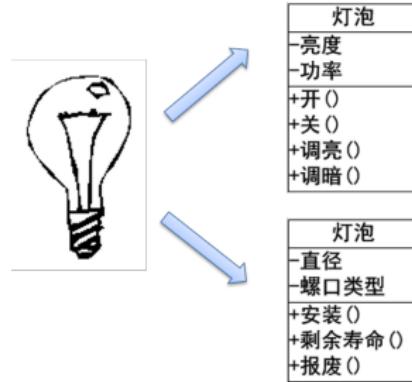
- 类是同种对象的集合与抽象
 - 类描述对象的共同的数据特征和行为特征
 - 类的实例化就是对象
-
- 抽象和实例化是对象与类联系的关键

抽象

- 抽象是一种设计技术，用以
 - 说明一个实体的本质方面，而忽略或掩盖其非本质方面
 - 将复杂现象简化到可以分析、实验或者可以理解的程度

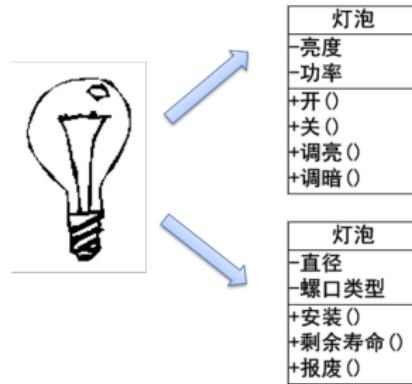
抽象

- 抽象是一种设计技术，用以
 - 说明一个实体的本质方面，而忽略或掩盖其非本质方面
 - 将复杂现象简化到可以分析、实验或者可以理解的程度



抽象

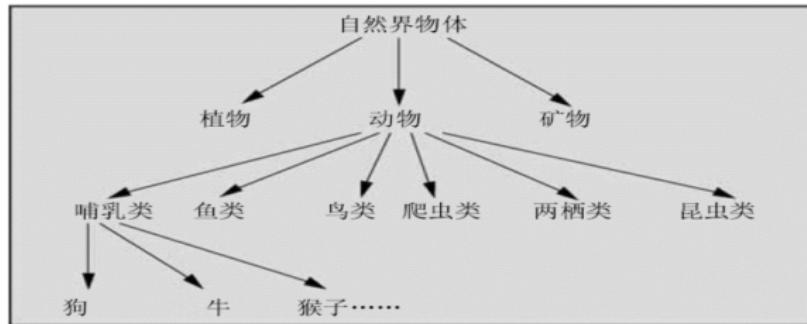
- 抽象是一种设计技术，用以
 - 说明一个实体的本质方面，而忽略或掩盖其非本质方面
 - 将复杂现象简化到可以分析、实验或者可以理解的程度



- 作用：简化。识别重要细节，忽略无关细节，过滤掉与问题空间中模型无关的侧面
- 单个实体可能有多种抽象
- OOP过程需要确定将哪些属性和行为包括在给定的抽象中

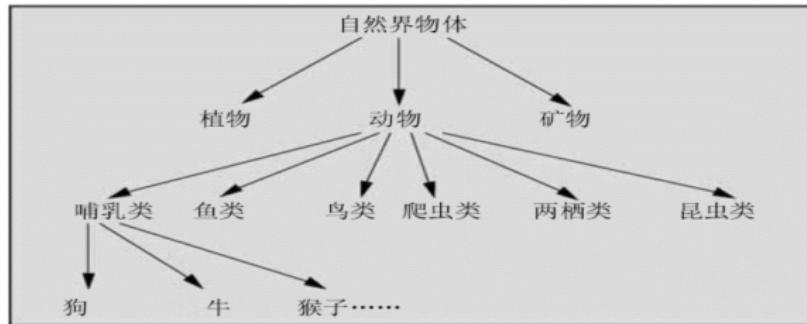
抽象

- 类是通过在对象抽象的基础上对抽象结果的组织获得的



抽象

- 类是通过在对象抽象的基础上对抽象结果的组织获得的



- 接口 (interface) 是“组织”的依据之一

接口

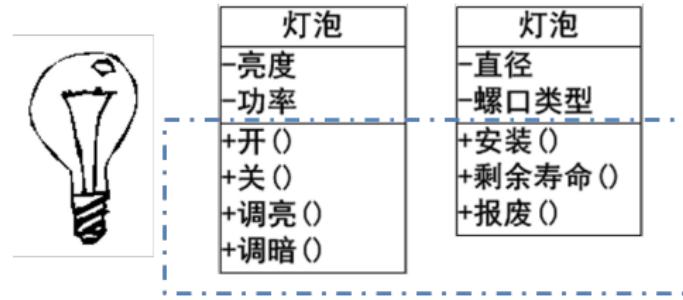
An interface is a description of the actions that an object can do for the outside world.

接口

An interface is a description of the actions that an object can do for the outside world.

接口如何产生？

- 确定可能向对象发出的请求（在屏幕上画图，打开开关…）
- 由方法声明定义这些请求
- 聚合这些方法声明



接口

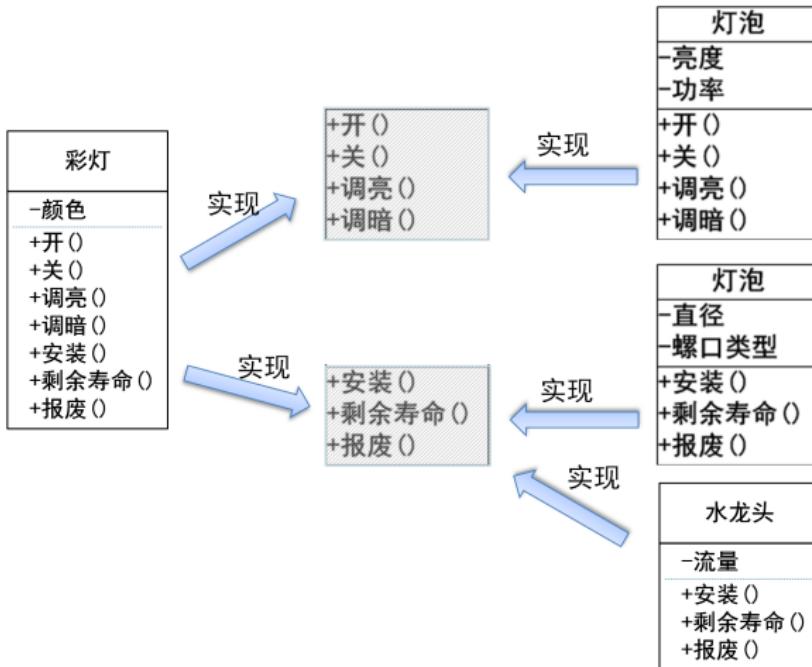
An interface is a description of the actions that an object can do for the outside world.

接口如何产生？

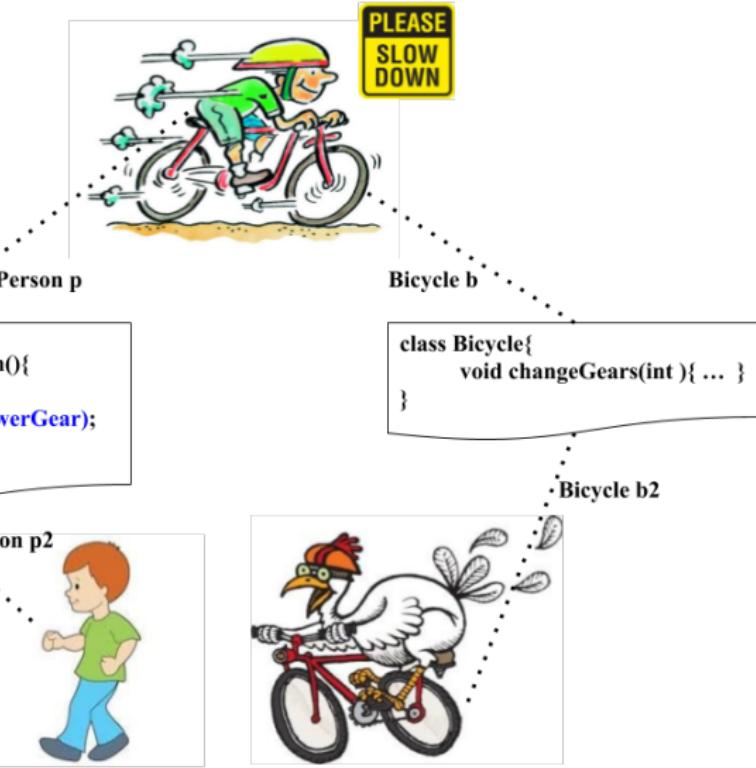
- 确定可能向对象发出的请求（在屏幕上画图，打开开关…）
 - 由方法声明定义这些请求
 - 聚合这些方法声明
-
- 接口确定了对特定对象所能发出的请求，或者对象接收消息的方式

接口

- 接口与类的关系是实现关系
 - 一个类可以实现多个接口
 - 一个接口可以被多个类实现



类的实例化



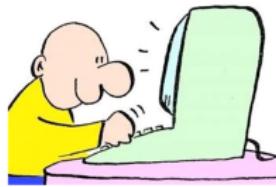
提要

1 面向对象程序设计的定义

2 对象和类

3 封装，继承和多态

封装

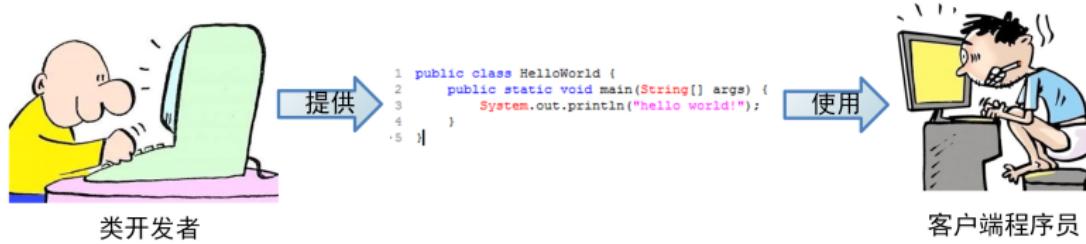


类开发者

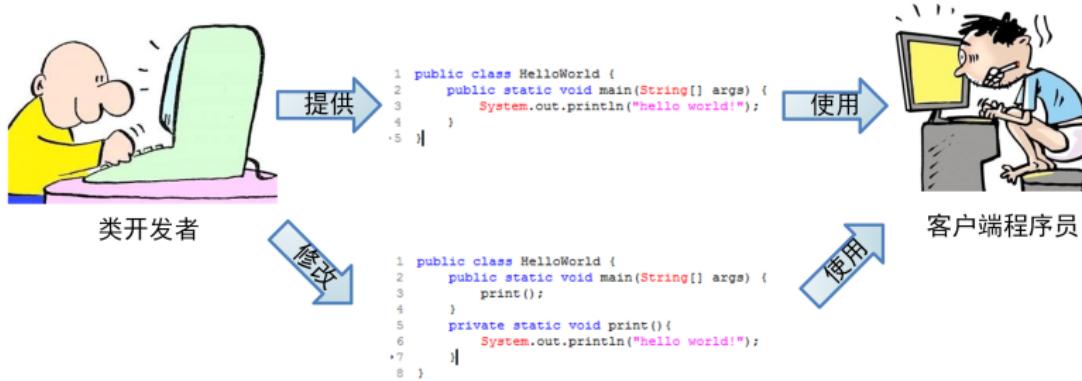


客户端程序员

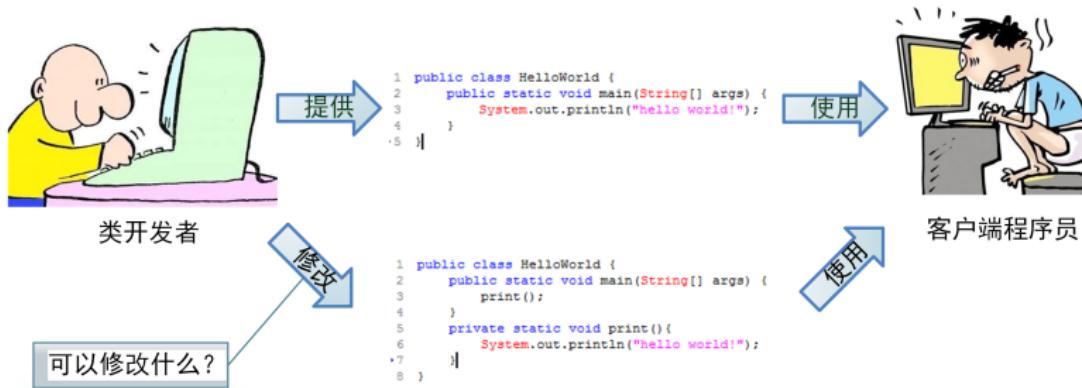
封装



封装



封装



封装

定义

把数据和方法包装进类中，并对具体实现进行隐藏

- 类是基本的封装单元，一个类的所有对象实例有相同的封装特性
- “具体实现”包括：对象的数据域、方法实现以及非接口方法的定义等
- 对数据的完全隐藏是对象的理想结构，现实中使用public、protected、private关键字实现4种隐藏能力（访问权限）

封装

意义

- 模块化：不同对象间仅通过必要的消息进行交互
- 保证数据对象的一致性：通过隐藏对象变量和方法实现，防止绕过接口更改变量
- 易于维护：对私有变量和私有方法的更改，不会影响到调用对象接口的其他程序，提高了程序的可移植性

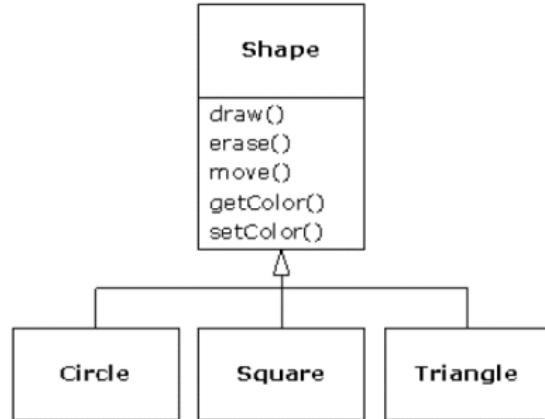
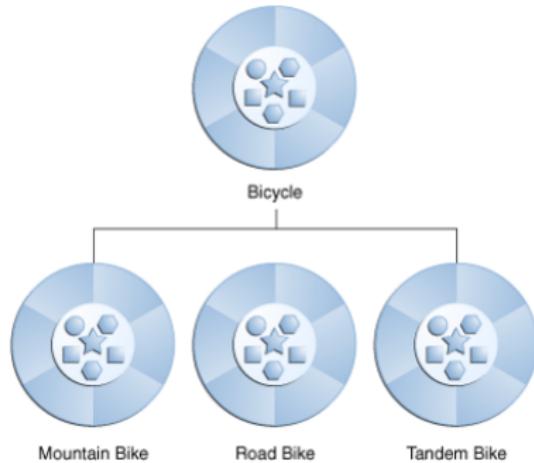
继承

- 现实世界中的对象间关系

- 包含 (has a) : 对象A是对象B的组成部分
- 关联: 对象A中保存对象B的一个引用而非对象B本身
- 继承 (is a) : 对象A是对象B的特例, 抽象 → 具体

继承

- 现实世界中的对象间关系
 - 包含 (has a) : 对象A是对象B的组成部分
 - 关联: 对象A中保存对象B的一个引用而非对象B本身
 - 继承 (is a) : 对象A是对象B的特例, 抽象 → 具体



继承

含义

描述子类与父类之间的关系。子类是父类的特例，继承、修改和细化父类的状态和行为

- 子类：从某个特定类派生出来的类
- 父类：派生出某个特定类的类

原则

- 子类继承父类的变量和方法
- 子类可以增加新的变量和方法
- 子类可以重写（Override）继承来的方法
- 继承关系可以有多层，子类要继承它所有父类的方法与状态

继承

意义

- 定义对象之间的层次关系，下层对象继承了上层对象的特性，实现对父类中代码的重用
- 将一般行为与特殊行为分离
 - 一般行为包含在父类中
 - 从父类继承得到的特定子类，实现特殊行为的具体细节（新变量、新方法、重写父类方法）

多态

定义

泛指能够通过同一名称或接口表示多种形式的方法和多种类型的对象的能力

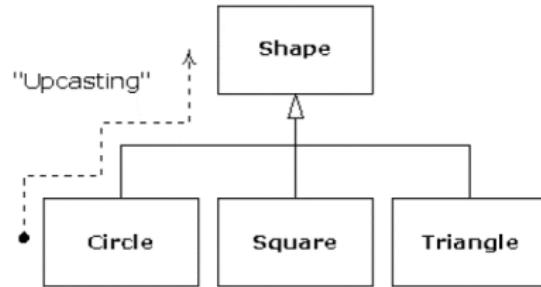
多态的形式

编译时多态：一个类中定义多个名称相同但参数不同的方法
通过方法重载（overloading）实现

多态的形式

运行时多态：通过相同的接口访问不同类型的对象的能力

- 这些对象所对应的每一种类型都提供了对该接口的独特实现
- 通过方法重写、向上转型和动态绑定，达到“对外一个接口，内部多种实现”
 - 以继承为基础
 - 改变从父类继承的行为（方法重写， overriding）
 - 将子类对象当作父类对象看待（向上转型， upcasting）
 - 在运行时确定接收消息的对象的类型及其行为（动态绑定， dynamic binding）



多态

意义

使得在程序运行时，对象的一种接口可以提供多种不同的操作，提高程序的灵活性