1. 编程语言概述

编译：把整个程序源代码翻译成另外⼀种代码，然后等待被执⾏，发⽣在运⾏之前，产物是另⼀份代码

解释：把程序源代码⼀⾏⼀⾏的读懂然后执⾏，发⽣在运⾏时，产物是运⾏结果

编译→存储→执行

词法分析，将符号分组→语法分析，得到语法树→语义分析，检查并输出，执行

1. 可计算性

希尔伯特判定问题：试图找到清楚明白的验算程序，只要用所谓的一阶逻辑的符号系统写出来的某些前提和所提出的结果给定，那么通过这些程序就是可以判定的。

停机问题：不存在这样一个程序（算法），它能够计算任何程序（算法）在给定输入上是否会在有限时间内结束（停机）

哥德尔不完备性问题：任何相容的形式系统，只要蕴涵皮亚诺算术公理，它就不能用于证明它本身的相容性。

1. Lambda演算

Lambda项的定义：原子（单个变量），应用（多个lambda项连接）和抽象（lambda+变量.lambda项M）

自由变量：对原子：变量的集合 对应用：多个lambda项的变量的并 对抽象：M中去掉lambda后面的变量的集合

Alpha变换：把λx.M的x换成其他任何不是自由变量的变量

Beta规约：(λx.M)N -> [N/x]M 用N代替x 成为β缩减项

1. 程序结构

算法建模的三种机制：基本表达式、分解、抽象

1. 函数式编程思想
2. 结构式编程思想

结构化思想：自顶向下逐步求精，算法+数据结构

首先根据行为来分解，然后设计数据来配合行为

数据流图的世界观：输入 -> 计算系统 -> 输出

所有的计算系统都是信息的处理和转换。

过程：对数据的处理，接收输⼊，进⾏数据转换，输出结果

将系统看成过程的集合

可能需要和软件系统外的实体尤其是⼈进⾏交互

通过系统的数据流总是从输⼊被转换为输出

过程是对数据的处理，必须有输⼊，也必须有输出，输⼊数据集应该

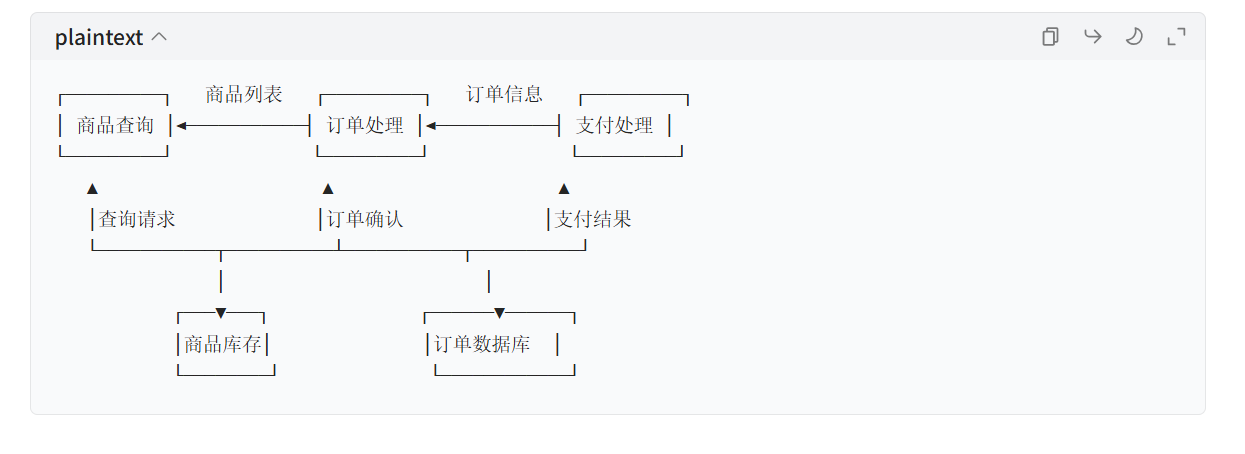
和输出数据集存在差异

数据流是必须和过程产⽣关联的，它要么是过程的数据输⼊，要么是

过程的数据输出，所有对象都应该有⼀个可以唯⼀标示⾃⼰的名称

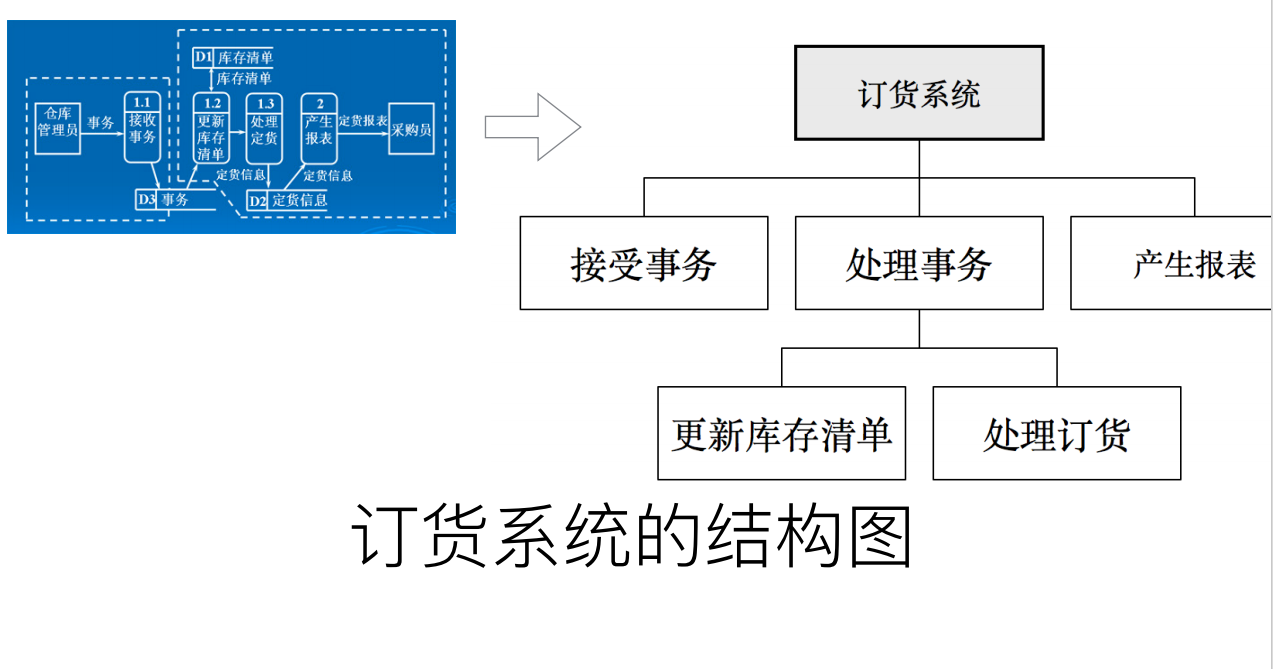
理解数据流图：处理并不一定是程序，数据存储也不等同于一个文件，他代表的是静态的数据，数据流是动态的数据

数据流图举例：



结构图：展示系统的模块划分，层次关系和调用依赖

通过模块的嵌套或排列位置暗示层级关系



流程图：直观描述流程、步骤或算法

变量：局部变量：stack区

成员变量：heap区

Static变量：code区

方法：指令块

Final方法永远不能被覆盖

声明：public/private (abstract) type name (){

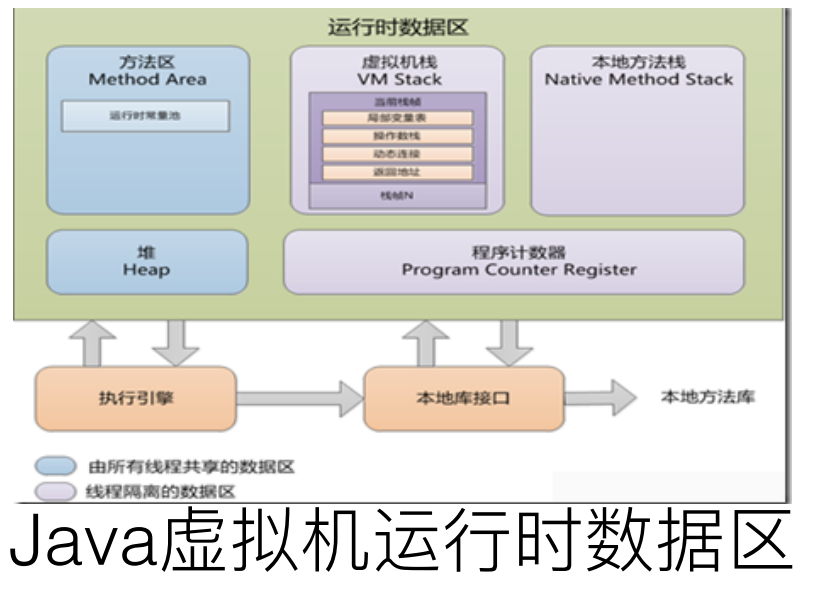
}

三种语句：顺序、选择（if-else）、循环

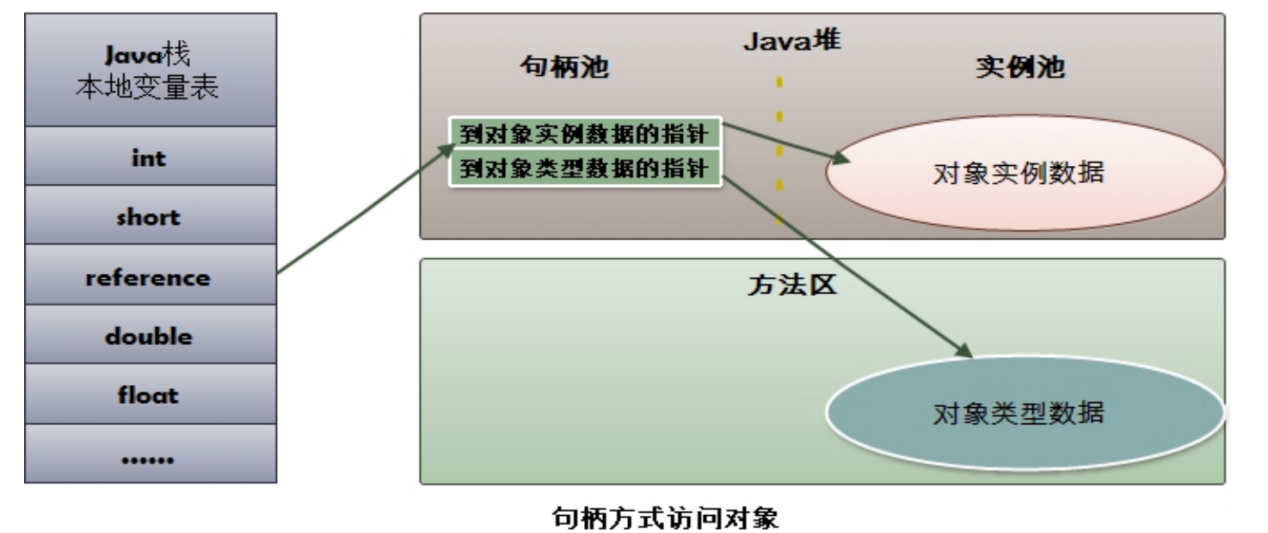
方法重载：改变参数个数或者类型，方法名不变

不可以改变返回值，也不能单纯改变参数的名字，要改变类型

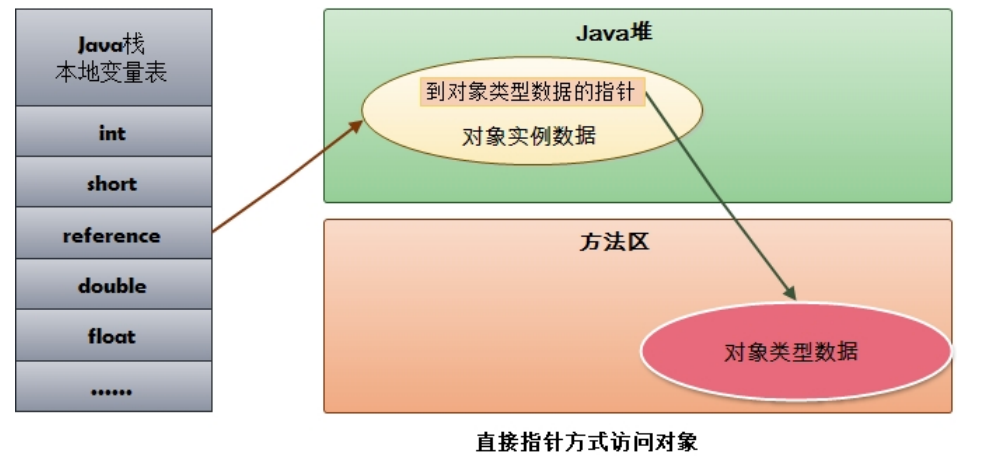
Java运行时数据区（内存）：



两种访问方式：



当用句柄方式访问对象时，Java 代码里的对象引用（比如 Object obj = new Object(); 中的 obj ），存储的是对象在句柄池中的句柄地址 。而句柄本身包含两部分关键信息的指针，到对象实例数据的和到对象类型数据的



1. 面向对象I

类：职责的抽象，每个类应当只有单⼀职责

对象：职责的实现 抽象与具体的关系

每个对象都保存着描述当前特征的信息，对象状态的改变必须通过调⽤⽅法实现。

1. 软件工程建模

软件开发生命周期的核心阶段：需求，设计，实现，测试，部署

数据建模：描述数据的存储、流转和关系

算法建模：用数学算法、模型结构提取问题，建立模型

1. JVM和字节码

JVM

三个部分：类加载器，运行时数据区和执行引擎

将编译后的字节码文件加载到其内部的运行时数据区域中进行执行，涉及类加载机制

JVM 定义了 Java 程序运行期间需要使用到的内存区域，简单

来说，这块内存区域存放了字节码信息

以及程序执行过程的数据

执行引擎充当了将高级语言翻译为机器语言的译者

运行时数据区包括：

•本地方法栈：运行本地方法

•虚拟机栈（栈）：存储方法调用和局部变量。

•堆：存储Java运行时所有的对象实例。

•方法区：存储类信息、常量、静态变量

Java类文件结构

1. 魔数，是 JVM 识别 class 文件的标志
2. 版本号，决定了该类文件可以被哪个版本的Java虚拟机（JVM）加载和执行
3. 常量池，它包含了类、接口、字段和方法的符号引用，以及字符串字面量和数值常量。这些信息在编译时被创建，并在运行时被Java虚拟机（JVM）使用
4. 访问标记，用于识别类或接口的访问信息
5. 类索引、父类索引和接口索引
6. 字段表，一个类中定义的字段会被存储在字段表，方法表和字段表类似，区别是用来存储方法的信息

对于运行时数据区：

1. 程序计数器 为了线程切换后能恢复到正确的执行位置，每个线程都需要有一个独立的程计数器，并且不能互相干扰，否则就会影响到程序的正常执行次序。
2. 虚拟机栈 是一个先入后出栈，线程调用某个对象的方法时，jvm会创造一个栈帧放在虚拟机栈中，线程运行过程中只有一个栈帧处于活跃状态。
3. 虚拟机栈操作的基本元素就是栈帧，栈帧主要包含了局部变量表、操作数栈、动态连接以及方法返回地址。栈帧是一个先进后出的数据结构，每个方法从调用到执行完成都会对应一个栈帧在虚拟机栈中入栈和出栈。
4. 动态链接 实现Java多态特性的基础机制
5. 常量池：存放于字节码里，字节码文件的资源仓库。运行时，JVM 会将字节码文件中的常量池加载到内存中，存放在运行时常量池中。字符串常量池，用于存放字符串常量的运行时的对象的引用，在堆中

JVM将内存分为年轻代和老年代

字节码指令集

1. 加载和储存指令 用于将数据从栈帧的局部变量表和操作数栈之间来回传输
2. 算术指令 用于对两个操作数栈上的值进行某种特定运算，并把结果重新存入到操作栈顶
3. 类型转换指令 将两种Java虚拟机数值类型相互转换
4. 对象创建和访问指令
5. 操作数栈管理指令。。。

类的生命周期：加载、验证、准备、解析、初始化、使用和卸

载

加载：将字节码从不同的数据源转化为二进制字节流加载到内存中，并生成一个代表该类的 java.lang.Class 对象

验证：对二进制字节流进行校验

准备：为静态变量分配内存并初始化

解析：将常量池中的符号引用转化为直接引用

初始化：静态变量被初始化成想要的值

1. 面向对象II-封装

封装规则：把数据和对数据操作的行为放在同一个类中，用职责来驱动数据和行为的组合方式，职责应当是完整的

数据职责：对象的本质特征，计算

行为职责：对象的行为特征，二者在一起

Static 静态方法：允许一个方法在没有类实例的情况下运行，不依赖于实例变量

静态变量在类被加载时被初始化，在创造类的任何对象、运行任何静态方法之前就被初始化

被标记为final的，一旦初始化，就不能被改变

变量在所有方法运行之前被初始化，甚至在构造方法之前，先初始化静态变量，然后非静态

1. III-协作

对象：一个或多个角色的实现

一组对象共同协作履行责任，职责的分与聚

B有全部数据，A要计算，要直接调用B的计算方法，不能调用B的数据再在A中计算，确保单一职责

如果AB各有部分计算所需的数据，看谁拥有职责更合适，面向对象偏向于调用B计算方法然后把结果作为参数传回给B

类之间的关系：

1. 依赖 物理关系，被依赖的对象只作为工具在使用，比如车和轮胎
2. 连接 relationship
3. 关联 逻辑关系

①普通关联

②可导航关联 比如老师 -> 学生

③聚合 sb. Owns sth.

④组合 sth. Is a part of sb.

1. 继承
2. 实现
3. IV-继承与多态

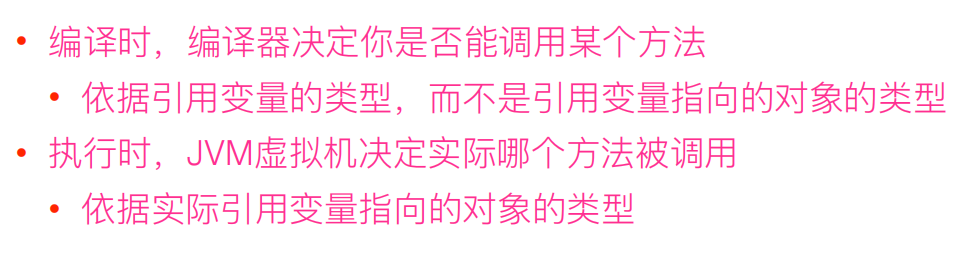
子类继承父类所有成员变量和方法，不可以覆盖成员变量，父类的私有变量（使用private修饰的字段）会被子类继承，但子类无法直接访问这些变量，可以用父类中的get方法间接访问

抽象类本身没有用，除非他被继承，其子类可以被实例化

抽象类里面可以有非抽象的方法，非抽象类不可以有抽象方法

抽象方法存在就是为了多态，子类必须实现父类的所有抽象方法

Object类，所有类的父类



编译时多态（重载相关）和运行时多态（重写相关），编译时：“看引用类型，检查能不能调方法”比如

Animal animal = new Dog();

animal.bark(); // 编译时看 animal 的类型是 Animal，Animal类中有bark方法才能调用

执行时：“看实际对象类型，决定调哪个方法”

运行时 animal 实际指向的是 Dog 对象，JVM 会根据真正的对象类型（Dog） ，去调用 Dog 里的 bark 方法。这就是为啥子类重写父类方法后，运行时会“动态”选子类实现。

一个对象包含了从父类继承的所有东西，意味着他也是父类的类型的对象



变量不存在覆盖的说法，因为是两个不同的变量，访问时候只看类型，比如Father obj = new Son(); //obj的类型是Father

System.out.println(obj.name); // 用父类引用，访问的是父类的name

1. V-可修改性

需求的变更

组合：通过在类中持有其他类的实例，形成 “has-a” 关系（如Car has a Engine）

继承：通过extends关键字，子类获得父类的属性和方法，形成 “is-a” 关系（如Dog is a Animal）

类的初始化包括三部分：

1. 加载，由类加载器执行，查找字节码，创造一个.class对象
2. 链接：验证字节码，为静态域分配空间
3. 初始化：首先执行初始化块static，初始化静态变量，执行静态方法

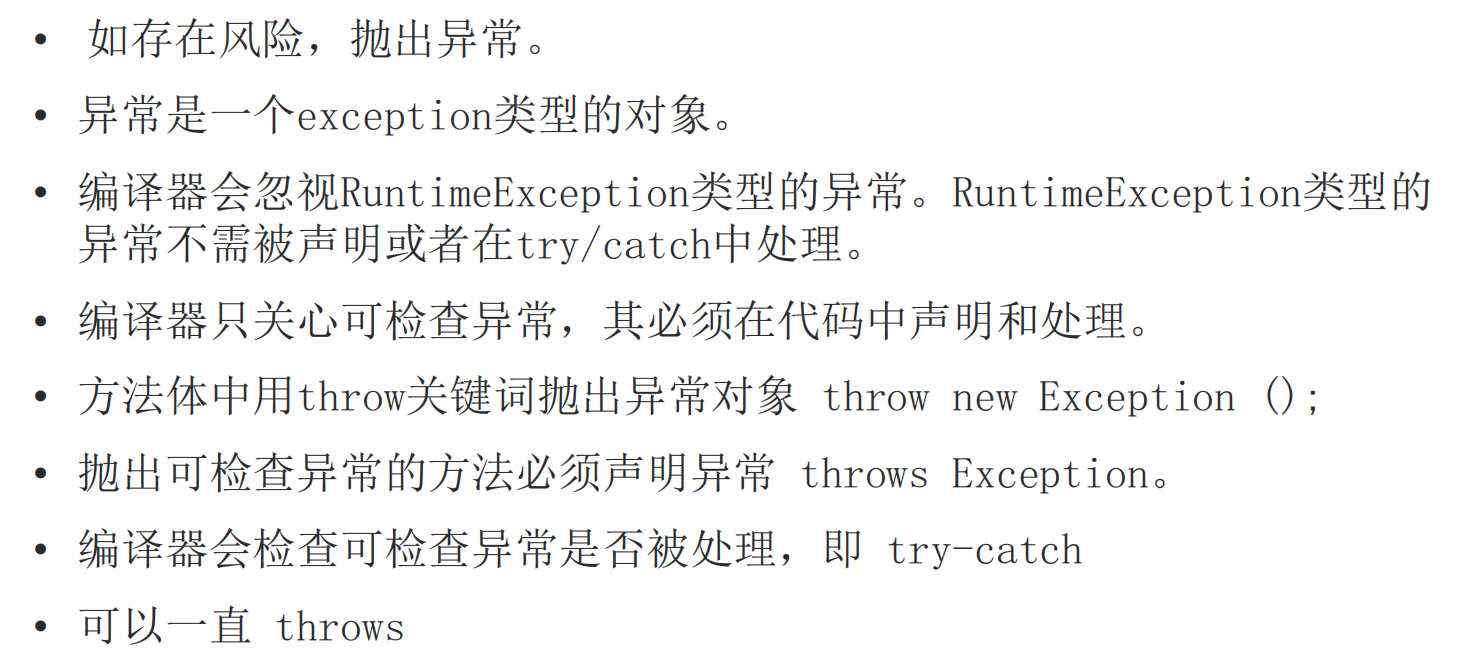
常量不会触发类的初始化，B extends A只会触发A的初始化，用数组比如创造一个Class的数组的时候不会触发Class的初始化

1. VI-接口

默认方法是接口中可以包含具体实现的方法，使用default关键字修饰。实现该接口的类不必强制重写这些方法，但可以选择性地覆盖。

如果不覆盖，那就是用接口里面定义好的方法体。

1. 异常



早抛出，晚捕获，一旦检测到错误就抛出异常