零基础学 Java

1

类（ Class ）

- 用变量表示商品
- 用类来表示商品

用变量表示商品

- 商品有标识，名字，数量，价格着几个属性
- 多个商品怎么办？重复，又是重复！

用类来表示商品

重新认识类（class）真正的用途

- 是用来描述同一类事物的
- 可以在内部定义任意数量的、不同类型的变量，作为这一类事物的属性。这种属性叫做成员变量（member variable）。
- 有类名，类名必须和文件名一样
- 就好像文件路径+文件名不能重复一样，一个Java程序中相同名字的类只能有一个

看例程：定类的语法

2

初探类和对象

- 如何创建类的实例/对象（ Instance/Object ）
- 通过点操作符操作对象的属性

如何创建类的实例/对象（ Instance/Object ）

- 从数据类型的角度来看，类就是自己创建了一种新的数据类型。类也叫做“自定义类型”。一个Java程序中不允许类同名。

- 看例程，学习类和对象的使用

通过点操作符操作对象的属性

认识点操作符

- 点操作符是用来访问/操作前面实体的属性的，类似于“的”
- merchandise.name可以读作merchandise的name。

3

认识引用类型

- 引用（reference）数据类型
- 引用数据类型和基本数据类型
- Java有一个大大的布告板，放着所有实例

引用（reference）数据类型

- Java 中的数据类型分为基本数据类型和引用数据类型
- 看例程，理解引用

引用数据类型和基本数据类型

引用数据类型和基本数据类型的相同点

- 都可以用来创建变量，可以赋值和使用其值
- 本身都是一个地址

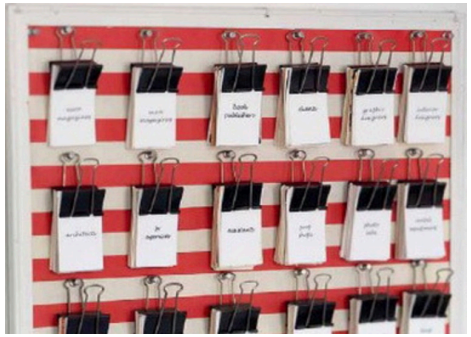
引用数据类型和基本数据类型的不同点

- 基本类型变量的值，就是地址对应的值。引用数据类型的值还是一个地址，需要通过“二级跳”找到实例
- 引用数据类型是Java的一种内部类型，是对所有自定义类型和数组引用的统称，并非特指某种类型

尝试输出一下引用类型的值到控制台，看看是什么

Java有一个的大大的布告板，放着所有实例

Merchandise m1 = new Merchandise();



- 使用 new 操作符可以创建某个类的一个实例。在 Java 程序运行的时候，所有这些创建出来的实例都被Java放在内存里一个叫做堆（heap）的、类似公告板的地方
- 创建一个实例，就是根据类的定义，点出需要的“纸”，订成一个本子，挂在布告板上。实例本身，可以认为是一个小本子
- 引用里存放的，相当于某个本子所在的布告板的地址

4

类，对象和引用的关系

类，对象和引用的关系

类和对象的关系

- 类是对象的模版，对象是类的一个实例
- 一个 Java 程序中类名相同的类只能有一个，也就是类型不会重名
- 一个类可以有很多对象
- 一个对象只能根据一个类来创建

引用和类以及对象的关系

- 引用必须是、只能是一个类的引用
- 引用只能指向其所属的类型的类的对象
- 相同类型的引用之间可以赋值
- 只能通过指向一个对象的引用，来操作一个对象，比如访问某个成员变量

5

认识数组类型

认识数组类型

数组是一种特殊的类

- 数组的类名就是类型带上中括号
- 同一类型的数组，每个数组对象的大小可以不一样。也就是每个数组对象占用的内存可以不一样，这点和类的对象不同。
- 可以用引用指向类型相同大小不同的数组，因为它们属于同一种类型

引用的数组

- 可以把类名当成自定义类型，定义引用的数组，甚至多维数组

6

引用的缺省值——null

引用的缺省值——null

引用也有缺省值——null

- null是引用类型的缺省值
- null代表空，不存在。可以读作空
- 引用类型的数组创建出来，初始值都是空

null带来的问题

- 大名鼎鼎的 NullPointerException （ NPE ）
- 如果不确定，使用前要先判断引用是不是空

通过null理解引用的“二级跳”

7

像自定义类型一样使用类

像自定义类型一样使用类

类就是一种自定义类型

- 在类定义中可以使用类，创建类的引用
- 在类定义中，甚至可以使用类自己的类创建引用
- 引用类型的缺省值是null。一个类定义中如果有引用，创建出来的实例，其缺省值是null

8

Java中的包和访问修饰符

- 类多太混乱？用 package 管理
- 类使用太繁琐怎么办？用 import
- 属性访问修饰符：public
- 类的全限定名

类多太混乱？用 package 管理

- 为了避免类在一起混乱，可以把类放在文件夹里。这时就需要用 package 语句告诉 Java 这个类在哪个 package 里。 package 语句要和源文件的目录完全对应，大小写要一致
- package 读作包。一般来说，类都会在包里，而不会直接放在根目录
- 不同的包里可以有相同名字的类
- 一个类只能有一个 package 语句，如果有 package 语句，则必须是类的第一行有效代码

类使用太繁琐怎么办？用 import

- 当使用另一个包里的类时候，需要带上包名
- 每次使用都带包名很繁琐，可以在使用的类的上面使用 import 语句，一次性解决问题，就可以直接使用类了。就好像我们之前用过的 Scanner 类
- import 语句可以有多个
- 如果需要import一个包中的很多类，可以使用 * 通配符

属性访问修饰符：public

- 被 public 修饰的属性，可以被任意包中的类访问
- 没有访问修饰符的属性，称作缺省的访问修饰符，可以被本包内的其他类和自己的对象
- 访问修饰符是一种限制或者允许属性访问的修饰符

类的全限定名

- 包名 + 类名 = 类的全限定名。也可以简称为类的全名
- 同一个 Java 程序中全限定名字不可重复

9

打造一个小超市

- 小超市例程
- Java 的世界是一个类和对象的世界

小超市例程

Java 的世界是一个类和对象的世界

Java 就是使用类来描述世界，用类的实例（对象）让世界运转起来

各种操作数据的代码太乱了怎么办？

10

Intellij 调试程序初探

- debug：不让程序不要一闪而过
- 用断点调试程序

debug：不让程序不要一闪而过

设置断点，debug调试模式运行程序

断点（ breakpoint ）：可以让程序在调试模式停在某一行

用断点调试程序

Frame 视图和 Variable 视图。视图的作用，隐藏和显示

程序调试标准动作

- 查看变量的值，展开实例看内部成员变量的值
- 程序继续执行之 Step Over：执行一行
- 程序继续执行之 Resume：继续执行直到遇到下一个断点或者程序结束（视频中有口误，Step Out是执行到方法结束，我们会在讲到方法的时候讲解这个功能。在这里因为我们只有一个方法，Step Out和Resume是一样的效果）
- 执行任意代码之 Evaluate Expression：在对话框输入代码，直接执行看结果值
- 条件断点：给断点设置条件，只有满足条件时，程序才会在该断点停住

使用快捷键和调试更配哦

11

方法：让 Merchandise 对象有行为

- 如果要描述一个商品怎么办？
- 方法——让可以商品描述自己
- 详解describe方法

如果要描述一个商品怎么办？

- 看例程
- 每次需要描述商品，都要重复这些代码？一个成熟的商品应该学会自己描述自己

方法（Method）——让可以商品描述自己

方法英文名叫做method，又称作function。看例程，学习方法的语法

详解describe方法

方法的调用

- 通过引用的点操作符，可以调用对象的方法
- 方法调用要有括号，即使没有参数

方法可以使用的数据：对象的成员变量（member variable ）

12

返回值：让 Merchandise 计算利润

让商品自己计算利润

- 看例程： 让商品自己计算利润的方法
- 看例程：让超市找出利润最高的商品的方法
- 看例程：商品的其它方法
- 方法调用调试之 Step Into：进入被调用的方法内部继续调试

13

参数：让 Merchandise 计算多件商品的总价

- 第二件半价哦！
- 参数：告诉商品实例要购买多少个

第二件半价哦！

极客时间

看例程

这种方式有哪些问题？

- 每个用到这个逻辑的地方，都要重复这段代码
- 如果告诉商品要买多少个，商品帮我算多少钱，就不用重复代码了

参数：告诉商品实例要购买多少个

极客时间

看MerchandiseV2中的新方法，学习方法参数相关语法

极客时间

14

参数和返回值是怎么传递的

参数和返回值是怎么传递的

极客时间

- 参数和方法里的局部变量可以认为是一样的东西。只是在方法调用之前，会用实参给参数的形参赋值
- 发生在代码块里的，就让它留在代码块里。方法执行完毕，参数和方法的局部变量的数据就会被删除回收。就好像演草纸，作用是计算一个值，算好之后，演草纸就可以扔了
- 调用一个有返回值的方法时，就好像访问一个成员变量
- 看代码，学习参数传递
- 看代码，学习返回值的传递

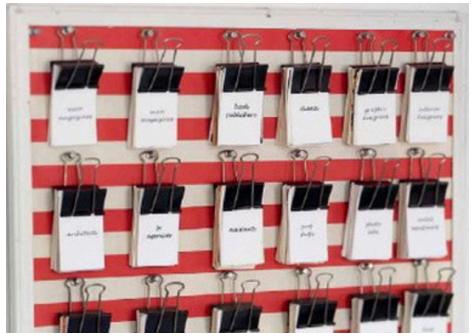
15

分清参数、局部变量和实例的地盘

分清参数、局部变量和实例的地盘

极客时间

- 局部变量就是我们之前说的变量，是在方法体里创建的变量
- 参数和局部变量都是演草纸，方法执行完就清除了
- 对象是实体/实例，不是变量。对象创建出来后，被堆在一起，放在类似公告板的地方。方法里创建的对象是不会随着方法结束被清除的。所以对象的地盘不受限制，只要有引用指向一个对象，这个对象的数据就可以通过这个引用来访问



- 看例程：理解局部变量，参数和实例

16

隐藏的 this 自引用

- 参数和局部变量重名会怎样
- 使用this：访问局部变量的完整形态

参数和局部变量重名会怎样

使用this：访问局部变量的完整形态

- 看例程，理解this自引用的用法和意义

极客时间

17

理解方法：一种特殊的代码块

- 方法是什么
- 方法的特殊之处

方法是什么

- 类中如果不定义方法，只定义成员变量，那么一个类就没有了功能，只是简单的数据的封装。创建一个对象之后，所有对这些数据的操作，都要在每个用到这些数据的地方写代码

- 类通过成员变量和方法描述世界。成员变量是描述一类事物的属性，是数据；方法是描述一类事物的行为和功能，是对数据的操作。比如最开始我们写的 describe 方法

- 方法中的代码可以通过操作一个对象的成员变量，完成一个功能

方法是什么（续）

- 方法是 Java 中代码执行的单元，是代码的载体。所有的代码，都必须属于某一个方法
- 方法就是一串语句，加上数据输入 this 自引用和参数，执行后得到一个返回值。所以使用一个对象调用一个方法，可以叫做调用对象的方法，也可以叫做“在这个对象上调用方法（ invoke a method on an object ）”
- 方法不是对象的一部分，它是类的一部分。每个对象可以给成员变量赋不同的值，但是无法让方法有不同的行为。同理，无论在一个类中定义多少方法，都不会让影响创建一个对象所占用的内存

方法的特殊之处

极客时间

- 和代码块比，方法特殊在：

  - 有名字
  - 有返回值
  - 有参数
  - 有 this 自引用
  - 明确的属于某一个类
  - 可以（也只能）通过对象引用来调用

18

理解方法的调用：代码的一种特殊跳转

理解方法的调用：代码的一种特殊跳转

极客时间

- 看例程：模拟this自引用传递
- 看例程：对象可以在自己的方法里调用自己类的别的的方法，甚至可以调用自己的方法。
- 方法在类中定义的先后，没有任何影响。
- 方法是一种根据名字做跳转的代码流程控制方式。它的特殊之处在于有this自引用，有参数，有返回值。

19

给类和方法加 Java 注释

给类和方法加 Java 注释

极客时间

- 多行注释
- 给类添加注释（Javadoc）
- 给方法添加注释（Javadoc）
- Oracle 官方的 Javadoc 地址

https://docs.oracle.com/en/java/javase/11/docs/api/index.html

20

成熟的类的对象要自己做事情

- 放着我来！
- 把操作成员变量的代码放在类里

放着我来！

- 看看我们之前的代码，还有哪些复制的到处都是
- 想一下到处复制代码的话，一旦有变化，就得到处改
- 解决这个问题的关键是什么？成熟的类会说：放着我来！成熟的类的对象要自己做事情，自己操作自己的成员变量

把操作成员变量的代码放在类里

- 初始化成员变量
- 简单访问和设置成员变量的值（ Java Bean ）
- 专有的一些计算逻辑
- 用类定义成员变量，并把操作成员变量的代码都放在类里，就是封装：
  - 可以集中管控，自己的成员变量别人不可以乱来，避免出现非法的状态，比如库存为负数
  - 代码逻辑可以公用，避免代码重复，修改的时候只需改一处
  - 封装的好，可以更好的抽象一类事物

21

方法的签名和重载

- 不同的场景， 不同的购买方法
- 方法重载

不同的场景，不同的购买方法

极客时间

- 看代码：不同的购买方法
- 有重复，但是又明显不一样，怎么办？

方法重载（overload）

极客时间

- 方法签名：方法名+依次参数类型。注意，返回值不属于方法签名。方法签名是一个方法在一个类中的唯一标识
- 同一个类中方法可以重名，但是签名不可以重复。一个类中如果定义了名字相同，签名不同的方法，就叫做方法的重载
- 看代码：重写我们的购买方法，理解方法签名

22

重载的参数匹配规则

重载的参数匹配规则

极客时间

- 看例程：有了论斤卖的商品，数量变成 double 类型。有论整个卖的，有散装称重卖的，事情开始复杂起来
- 看例程：方法调用的时候，参数就不必完全类型一样，对于数字类型的参数，实参数可以自动类型转换成形参类型即可
- 看例程：重载时如果参数类型不完全匹配怎么办

极客时间

23

构造方法：构造实例的方法

构造方法：构造实例的方法

- 如果 init 方法能在对象创建的时候就自动被调用多方便
- 自动调用的唯一的问题：调用哪个方法呢？
- 看例程：学习构造方法（constructor）的语法

极客时间

24

构造方法的重载和互相调用

构造方法的重载和互相调用

极客时间

- 看例程：构造方法的重载和互相调用的语法
- 看例程：局部变量赋初始值和构造方法殊途同归

25

静态变量

静态变量

- 尽量不要使用 Magic Number
- VIP 的折扣作为一个成员变量， hin浪费啊
- 看例程：学习静态变量（也叫做类变量）

26

静态方法

静态方法

极客时间

- 静态方法（也叫类方法）的特点：只能使用参数和静态变量。换言之，就是没有 this 自引用的方法
- 看例程：学习静态方法的定义和静态方法的调用

27

静态方法的重载

静态方法的重载

- 静态方法的重载和成员方法（实例方法）一样
- 看例程：理解方法的重载
- 顺带学一个三元操作符

28

static 代码块和 static 变量初始化

static 代码块和 static 变量初始化

- 看例程：学习静态代码块和执行顺序

29

方法和属性的可见性修饰符

方法和属性的可见性修饰符

极客时间

- 可见性修饰符用在类、成员方法、构造方法、静态方法和属性上，其可见性的范围是一样的
- 看代码，学习可见性修饰符：
  - public：全局可见
  - 缺省：当前包可以见
  - private：当前类可以见
- 理解访问修饰符：不只是为了限制不让人用，更为了有规矩才成方圆。成员变量应该是 private 的，不需要让外部使用的方法应该都是 private 的

30

重新认识老朋友：Math 和 Scanner

重新认识老朋友：Math

极客时间

- 查看Math类的源代码
- 看例程：学习Math中的常用方法
- Math类的文档

https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/lang/Math.html

重新认识老朋友：Scanner

极客时间

- Scanner 的作用
- 查看 Scanner 的源代码和 since，理解 public 带来的现实中的约束。
- 看源代码是学习的一种好办法

31

最熟悉的陌生人：String

最熟悉的陌生人：String

极客时间

- String 对象最重要的特点：不可变（immutable）不可变不可变，重要的事情说三遍。String 用来存储字符的数据是 private 的，而且不提供任何修改内容的方法，所以String 对象一旦生成，其内容就是完全不可能被修改的
- 看例程：学习 String 中常用的方法
- 看例程：写一个人工不智能聊天程序

极客时间

32

重新认识老朋友：main方法和System类

- main 方法使用指南
- System 类不简单

main 方法使用指南

极客时间

- main 方法也只是一个静态的，有 String[] 做参数的，没有返回值的方法而已。它的特殊性在于 Java 可以把 main 方作为程序入口
- 给 main 方法传递参数
- 自己试着调用 main 方法

System 类不简单

- System 类中有很多和系统相关的方法。我们用的最多的就是 in 和 out 来读取和输出数据
- 看例程：System 里另一个最常用的，无可替代的方法，取当前时间
- System的文档

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/System.html

33

String 类的好兄弟

String 类的好兄弟

- StringBuilder 是一个非常方便的用来拼接和处理字符串的类，和 String 不同的是，它是可变的
- 看例程：学习 StringBuilder 的使用

34

继承：方便让商品增加新的类别

- 手机是商品，但不只是商品
- 继承：问题迎刃而解

手机是商品，但不只是商品

- 手机类需要商品类所有的属性和方法，因为手机是商品，有价格，库存数量，进价售价等
- 但是手机也不只是商品，除了商品通用的属性和操作之外，手机还有自己这一类商品共有的属性，比如 CPU，内存，存储，品牌，操作系统，屏幕大小等描述信息
- 把商品类里的属性和方法都复制过来，然后再增加上手机自己的属性和方法。又是熟悉的问题！

继承：问题迎刃而解

极客时间

看例程：继承的语法和作用

- 子类继承了父类的方法和属性
- 使用子类的引用可以调用父类的共有方法
- 使用子类的引用可以访问父类的共有属性
- 就好像子类的引用可以一物二用，既可以当作父类的引用使用，又可以当作子类的引用使用。

35

子类对象里藏着一个父类对象

- 另一种解决问题的思路：组合
- 继承和组合的区别

另一种解决问题的思路：组合

极客时间

- 看例程：学习使用组合如何解决问题

继承和组合的区别

极客时间

- 看例程：初步了解组合和继承的区别

36

覆盖：子类想要一点不一样

覆盖：子类想要一点不一样

- 覆盖才是继承的精髓和终极奥义
- 看例程：学习覆盖
- 从覆盖的角度理解为什么使用方法读写属性，优于直接访问属性
- 问题：为了避免代码重复，在子类里使用父类的 buy 和 describe 方法。如何在子类里，调用父类的方法？

极客时间

37

super：和父类对象沟通的桥梁

super：和父类对象沟通的桥梁

极客时间

- 子类对象里可以认为有一个特殊的父类的对象，这个父类对象和子类对象之间通过 super 关键字来沟通
- 看例程：学习 super 的用法
- 理解 super 是什么，不是什么

38

super：调用父类的构造方法

super：调用父类的构造方法

极客时间

- 看例程：学习 super 的使用方法

39

父类和子类的引用赋值关系

父类和子类的引用赋值关系

极客时间

- 看例程：学习子类引用和父类引用的关系
- 父类引用可以指向子类对象，子类引用不可以指向父类的对象
- 可以进行强制类型转换，如果类型不对，会出错
- 可以调用的方法，是受引用类型决定的

40

多态：到底调用的哪个方法？

多态：到底调用的哪个方法？

极客时间

- 可以调用哪些方法，取决于引用类型。具体调用哪个方法，取决于实例所属的类是什么
- 看例程：学习覆盖的奥义
- 覆盖是多态里最重要的一种形式
- 从 this 自引用的角度，理解覆盖。

41

多态里更多的语法点

多态里更多的语法点

- 静态多态：重载（Overload）
- 动态多态：覆盖（Override）
- 看例程：学习重载和覆盖的花式操作
- 勿忘初心：程序的执行就是找到要执行的代码，并且知道执行的代码能访问哪些数据，数据从哪里来。多态的核心问题就是：要调用哪个类的哪个方法，这个方法用到的数据（this 引用）是谁

42

instanceof 操作符

instanceof 操作符

极客时间

- 看例程：学习 instanceof 操作符

43

继承专属的访问控制：protected

继承专属的访问控制：protected

极客时间

- 看例程：学习 protected 可见性 = default + 对子类可见
- 看例程：覆盖可以，但是不能让可见性更低

44

final 修饰符

final 修饰符最见不得变化

极客时间

- final 修饰类：不可被继承
- final 修饰方法：不可被子类覆盖
- final 修饰变量：不可被赋值。这个最难理解

45

继承里的静态方法

继承里的静态方法

极客时间

- 看例程：继承里的静态方法

46

插曲：for 循环的另一种写法

插曲：for 循环的另一种写法

极客时间

- 看例程：学习 for 循环的另一种写法

47

万类之祖：Object 类

万类之祖：Object 类

- 所有的类，都间接或者直接的继承自 Object 类
- Object 类中的方法
- Object 类引用

48

hashCode 和 equals 方法

- hashCode 和 equals 方法初探
- String 类的 equals 方法

hashCode 和 equals 方法初探

极客时间

- hashCode 可以翻译为哈希码，或者散列码。应该是一个表示对象的特征值的 int 整数
- equals 方法应该用来判断两个对象从逻辑上是否相等
- 看例程：hashCode 和 equals方法怎么写
- 看例程：使用equals 用来判断对象是否相等，而非 ==

String 类的 equals 方法

极客时间

- 看例程：String 类的 equals 方法和 == 判断

49

toString 方法

toString 方法

- 看例程：自动生成 toString 方法
- 看例程：调用 toString 方法的地方有很多

50

初探 Class 类

初探 Class 类

- Class 类是代表类的类。每个Class类的实例，都代表了一个类
- 看例程：看看 Class 类里有什么

51

初探反射

初探反射

极客时间

- 使用反射（ reflection ） 访问属性
- 使用反射访问方法
- 使用反射访问静态方法和属性
- 使用反射访问 private 的方法和属性
- 反射是什么

52

面向对象三要素：封装、继承和多态

面向对象三要素：封装、继承和多态

极客时间

- 封装解决了什么问题，带来了什么问题
- 继承解决了什么问题，带来了什么问题
- 多态解决了什么问题，带来了什么问题

53

枚举：定义商品的门类

枚举：定义商品的门类

极客时间

- 看例程：使用枚举定义商品门类
- 看例程：枚举的使用

54

接口：让商品类型更丰富

接口：让商品类型更丰富

极客时间

- 看例程：学习接口的定义
- 看例程：学习接口的实现和接口引用的使用
- 看例程：学习接口的继承

55

抽象类：接口和类的混合体

抽象类：接口和类的混合体

极客时间

- 看例程：学习抽象类的定义和使用
- 看例程：学习抽象类的引用的使用
- 在 IDE 里看接口所有的实现类

56

有方法代码的接口

有方法代码的接口

- 在 Java 8 中，接口中允许有缺省实现的抽象方法
- 看例程：不用抽象类，而是使用缺省方法的接口

57

接口内的代码的更多内容

接口内的代码的更多内容

- 在 Java 8 中，接口中允许有静态方法，私有方法和带有缺省实现的抽象方法
- 看例程：接口中有方法，不代表是 Java 有多继承
- 看例程：理解接口方法里的 this 自引用
- 看例程：一个类不可以从两个接口中获得相同的缺省方法
- 看例程：接口中的静态方法

58

静态内部类

静态内部类

- 看例程：静态内部类的语法，了解其和外部类互相的可访问性
- 看例程：静态内部类在定义它的外部使用
- 对比记忆：静态内部类可以被什么修饰，可以在哪里使用，可以对比静态变量
- 看 Math 类源代码，学习静态内部类的一种曲线用途：实现单例模式

59

成员内部类

成员内部类

极客时间

- 看例程：学习成员内部类语法
- 看例程：学习在外部创建成员内部类的对象的语法
- 对比记忆：成员内部类可以被什么修饰，可以在哪里使用，可以对比成员变量

60

局部内部类

局部内部类

极客时间

- 看例程：学习局部内部类
- 对比记忆：局部内部类可以被什么修饰，可以在哪里使用，可以对比局部变量

61

匿名类

匿名类

- 匿名类是用来创建接口或者抽象类的实例的
- 匿名类可以出现在任何有代码的地方
- 看例程：学习匿名类的使用

62

特殊类的总结

- 枚举
- 非公有类
- 内部类
- 匿名类
- 类就一个

枚举

- 枚举就是有固定个数实例的类
- 枚举的父类是Enum

非公有类

极客时间

- 最不特殊的类，可以认为就是被缺省访问控制符修饰的类。也就是说，和 public class 的区别仅仅是可以被访问的范围不一样
- 如果一个文件只有非公有类，那么类名和文件名可以不一样。当然文件后缀必须是 java 。

内部类

- 内部类的特殊之处在于可见性和可以访问的数据以及方法。内部类会被认为是类本身的代码，所以外部类的 private 成员对其可见

- 类里面可以有静态变量，成员变量和局部变量，对比着看，内部类也分为这三种。这些内部类的访问修饰符，可以访问的数据以及可见性都可以对比着记忆
  - 静态内部类：可以有访问修饰符，可以在类外部访问（对比静态变量）
  - 成员内部类：可有访问修饰符，有外部类对象的 this 自引用（对比成员方法），可以在外部使用，但是创建对象语法需要指明外部对象
  - 局部内部类：没有访问修饰符（对比局部变量），有外部类的引用，访问参数和局部变量，必须是 final 的

- 内部类可以有父类，可以实现接口。

匿名类

极客时间

- 匿名类是一种创建接口和抽象类对象的语法，任何可以 new 一个对象的地方，都可以使用匿名类
- 匿名类只能实现/继承一个接口/抽象类，本身没有名字
- 如果是在成员方法或者给成员方法赋值时创建匿名类，那么会有对外部对象的this自引用
- 匿名类也可以访问外部类的 private 属性

类就一个

- 无论是内部类还是匿名类，类都是只有一个，对象可以有多个。不会在每次执行到内部类声明的地方，就创建一个新的类

63

让我们的超市运转起来：设计篇

设计自己的超市

极客时间

- 每个人心中都有自己的超市，自己创造的超市最棒！

- 我设计的一个超市：
  - 主角：超市，消费者，商品，购物车和服务员
  - 配角：优惠策略，卡
  - 互动模式：超市，消费者和商品封装各自的功能。服务员主导整个过程，覆盖帮助消费者选择商品，结算，处理会员积分等

- 拿到一个功能，先思考和抽象出功能的本质，理清脉络。想清楚这个功能需要哪些角色。然后根据角色和角色之间的互动，设计出接口，最后再去实现。面向接口编程，而非实现。

64

让我们的超市运转起来：代码篇

让我们的超市运转起来：代码篇

极客时间

- 梳理一下类的关系和关键实现
- 让程序跑起来看看
- 有兴趣可以试着做做我留的思考题