Course 2 Java intermediate

Module 1 类与对象

- 对象是实体，需要被创建，可以为我们做事情
  - 对象（这只猫）
    - 表达东西或事件
    - 运行时相应消息（提供服务）
  - 数据：属性或状态
  - 操作：函数
  - 对象内部的数据是被外部的操作紧密包围的 -》封装
    - 由操作去保护内部的数据，而数据是不对外公开的
- 类是规范，根据类的定义来创建对象
  - 类（猫）
    - 定义所有猫的属性
    - 就是Java中的类型
    - 可以用来定义变量

定义类

类的组成

表示对象有什么的成员变量

表示对象能做什么的成员函数

一旦定义了类，我们就可以创建这个类的多个对象，这些对象都会做那个类所定义的动作（函数），但是各自具有不同的数据。

This

是成员函数的一个固有的特殊的本地变量，它表达了调用这个函数的那个对象

成员变量定义初始化

- 成员变量在定义的地方就可以给出初始值
- 没有给出初始值的成员变量会自动获得0值
    - 对象变量的0值表示没有管理任何对象，也可以主动给null值
- 定义初始化可以调用函数，甚至可以使用已经定义的成员变量

构造函数

- 如果有一个成员函数的名字和类的名字完全相同，则在创建这个类的每一个对象的时候会自动调用这个函数 -》构造函数
- 这个函数不能有返回类型

函数重载

- 一个类可以有多个构造函数，只要他们的参数表不同
- 创建对象的时候给出不同的参数值，就会自动调用不同的构造函数
- 通过this()还可以调用其他构造函数
- 一个类里的同名但参数表不同的函数构成了重载关系

Module 2 对象交互

对象和对象之间的联系紧密程度叫做耦合。对象和对象的耦合程度越紧，表现在源代码上，就是它们的代码是互相依赖、互相牵制的。我们理想的模型，是对象和对象之间的耦合要尽可能的松，平行的对象要尽量减少直接联系，让更高层次的对象来提供通信服务。

访问属性

封装，就是把数据和对这些数据的操作放在一起，并且用这些操作把数据掩盖起来，是面向对象的基本概念之一，也是最核心的概念。

1. 所有的成员变量必须是private的，这样就避免别人任意使用你的内部数据；
    a. private只能用于成员变量和成员函数，不能用在函数里面（本地变量不能是private）

b. private只能在这个类内部使用（只有自己能访问）

   i. 类内部指类的成员函数和定义初始化

   ii. 这个限制是对类的而不是对对象的

2. 所有public的函数，只是用来实现这个类的对象或类自己要提供的服务的，而不是用来直接访问数据的。除非对数据的访问就是这个类及对象的服务。简单地说，给每个成员变量提供一对用于读写的get/set函数也是不合适的设计。

   a. public是任何人都可以访问的

      i. 任何人指的是在任何类的函数或定义初始化中可以使用

      ii. 使用指的是调用、访问或定义变量

如果函数或变量前没有定义private或者public，那么意味着friendly，即和它位于同一个包package的其他类都可以访问它

如果一个类是public的，任何人都可以用这个类的定义去定义变量

并且有一个要求，这个类必须处于源代码文件里，而这个源代码的文件名必须和这个类的名字相同

编译单元

一个.java文件，一个源代码文件是一个编译单元

编译单元是编译的时候一次对这一个编译单元做编译的动作

一个编译单元可以有很多个类，但是只有一个可以是public

包 - package

当你的程序越来越大的时候，你就会需要有一个机制帮助你管理一个工程中众多的类了。包就是Java的类库管理机制，它借助文件系统的目录来管理类库，一个包就是一个目录，一个包内的所有的类必须放在一个目录下，那个目录的名字必须是包的名字。

import别的包的时候一般不全import进来（import display.*），为了避免有重名而引起的冲突，一般import具体的东西

像java.util.Scanner，包的名字中间的那个点代表的是文件系统里文件夹的层次

类变量

类是描述，对象是实体。在类里所描述的成员变量，是位于这个类的每一个对象中的。

Static

而如果某个成员有static关键字做修饰，它就不再属于每一个对象，而是属于整个类的了。

通过一个对象去修改static成员的时候，别的对象里那个static成员也会被修改

通过每个对象都可以访问到这些类变量和类函数，但是**也可以通过类的名字来访问它们**。类函数由于不属于任何对象，因此也没有办法建立与调用它们的对象的关系，就不能访问任何非static的成员变量和成员函数了。

在一个static函数里不能访问non-static变量或函数，只能访问static函数和成员变量

Module 3 对象容器

顺序容器

容器 - Collection、Container

所谓容器，就是“放东西的东西”。数组可以看作是一种容器，但是数组的元素个数一旦确定就无法改变，这在实际使用中是很大的不足。一般意义上的容器，是指具有自动增长容量能力的存放数据的一种数据结构。在面向对象语言中，这种数据结构本身表达为一个对象。所以才有“放东西的东西”的说法。

Java具有丰富的容器，Java的容器具有丰富的功能和良好的性能。

我们首先学习的是顺序容器，即放进容器中的对象是按照指定的顺序（放的顺序）排列起来的，而且允许具有相同值的多个对象存在。

容器类

- `ArrayList<String> notes = new ArrayList<String>();`
- 容器类有两个类型
  - 容器的类型 → ArrayList
  - 元素的类型 → String

对象数组

int数组里每一个放的是整数

string数组里的每一个不是字符串本身，而是指向一个别的字符串的管理者

当数组的元素的类型是类的时候，数组的每一个元素其实只是对象的管理者而不是对象本身。因此，仅仅创建数组并没有创建其中的每一个对象！

其中每一个对象一开始都是null

for-each循环

对于int数组来说，k++不会起到任何作业，因为在循环的每一轮，拿出来的每一个k都只是对这个元素的一个复制品

```
int[] ia = new int[10];
for (int= i=0; i<ia.length; i++) {
  ia[i]=i;
}
for (int k:ia) {
  k++
}
```

而对于对象数组来说，不一样，第二遍会输出0

因为数组a里，每一个单元都指向了外面的v，所以v=a[0]对对象变量来说代表了v这个变量指向了a[0]所管理的那个对象。而v.set(0)是让v所指的那个对象里面的值变成0，所以会改变数组里的值

```
class Value {
  private int i;
  public void set(int i) {this.i=i;}
  public int get() {return i;}
}

public class NoteBook {
  public static void main(String[] args) {
    Value[] a = new Value[10];
    for (int i = 0; i<a.length; i++) {
      a[i] = new Value();
      a[i].set(i);
    }
    for ( Value v:a) {
      System.out.println(v.get());
      v.set(0);
    }
    for ( Value v:a) {
      System.out.println(v.get());
    }
```

对容器类来说，for-each循环也是可以用的

集合容器 set

集合就是数学中的集合的概念：所有的元素都具有唯一的值，元素在其中没有顺序。

```
HashSet<String> s = new HashSet<String>();
```

散列表 Hash table

传统意义上的Hash表，是能以int做值，将数据存放起来的数据结构。Java的Hash表可以以任何实现了hash()函数的类的对象做值来存放对象。

```
HashMap<Integer, String> coinnames = new HashMap<Integer, String>();
```

> 这是一个面向对象的世界，在这些容器里面，所有的类型都必须是对象（Integer，String），而不能是基本元素（int，double，float）

key是唯一的，如果多次放入同一个key和不同的value，key等于最后放入的那个value

也可以用for loop

```
for (Integer k : coinnames.keySet() ) { // k is every key
  String s = coinnames.get(k); // get the value of key k
  System.out.println(s);
}
```

1. java的整型分为4种：byte、short、int、long。其中byte占1个字节、short占2个字节、int占4个字节、long占8个字节
2. 当知道循环具体次数的时候使用for loop，当希望loop在具体条件下停下的时候使用while loop
3. 从初始条件开始，代入到循环体里运行，完成一次运行后，根据迭代条件在迭代中使用的对象取下一个可迭代的值，检查是否达到判断条件，如果没有，把取出的值代入循环体里运行，如果达到判断条件，则循环终止
4. 在内存里数组是连续的，链表是离散的
5. 二维数组即当一维数组的元素，是一个个一维数组时构成
6. 在java里用s1.equals(s2)去判断两个字符串是否相等，因为字符串变量不是字符串的所有者，而是字符串的管理者

Module 4 继承

面向对象程序设计语言有三大特性：封装、继承和多态性。继承是面向对象语言的重要特征之一，没有继承的语言只能被称作“使用对象的语言”。继承是非常简单而强大的设计思想，它提供了我们代码重用和程序组织的有力工具。

类是规则，用来制造对象的规则。我们不断地定义类，用定义的类制造一些对象。类定义了对象的属性和行为，就像图纸决定了房子要盖成什么样子。

一张图纸可以盖很多房子，它们都是相同的房子，但是坐落在不同的地方，会有不同的人住在里面。假如现在我们想盖一座新房子，和以前盖的房子很相似，但是稍微有点不同。任何一个建筑师都会拿以前盖的房子的图纸来，稍加修改，成为一张新图纸，然后盖这座新房子。所以一旦我们有了一张设计良好的图纸，我们就可以基于这张图纸设计出很多相似但不完全相同的房子的图纸来。

基于已有的设计创造新的设计，就是面向对象程序设计中的继承。在继承中，新的类不是凭空产生的，而是基于一个已经存在的类而定义出来的。通过继承，新的类自动获得了基础类中所有的成员，包括成员变量和方法，包括各种访问属性的成员，无论是public还是private。当然，在这之后，程序员还可以加入自己的新的成员，包括变量和方法。显然，通过继承来定义新的类，远比从头开始写一个新的类要简单快捷和方便。继承是支持代码重用的重要手段之一。

类这个词有分类的意思，具有相似特性的东西可以归为一类。比如所有的鸟都有一些共同的特性：有翅膀、下蛋等等。鸟的一个子类，比如鸡，具有鸟的所有的特性，同时又有它自己的特性，比如飞不太高等等；而另外一种鸟类，比如鸵鸟，同样也具有鸟类的全部特性，但是又有它自己的明显不同于鸡的特性。

如果我们用程序设计的语言来描述这个鸡和鸵鸟的关系问题，首先有一个类叫做“鸟”，它具有一些成员变量和方法，从而阐述了鸟所应该具有的特征和行为。然后一个“鸡”类可以从这个“鸟”类派生出来，它同样也具有“鸟”类所有的成员变量和方法，然后再加上自己特有的成员变量和方法。无论是从“鸟”那里继承来的变量和方法，还是它自己加上的，都是它的变量和方法。