1. 使用冒泡排序，实现如下的数组从小到大排序。

int[] arr = new int[]{34,5,22,-98,6,-76,0,-3};

2. 如何反转上面的数组。请代码实现

3. 写出二维数组初始化的两种方式

4. 代码实现二维数组的遍历.

int[][] arr = new int[4][3];

5. 二维数组元素的默认初始化值的理解

```
1. 使用冒泡排序，实现如下的数组从小到大排序。
 int[] arr = new int[]{34,5,22,-98,6,-76,0,-3};
for(int i=0;i<arr.length-1;i++){
    for(int j=0;j<arr.length-1-i;j++){
        if(a[j]>a[j+1]){

            int temp = a[j];
            a[j] = a[j+1];
            a[j+1]=temp;

        }
        }
    }

2. 如何反转上面的数组。请代码实现
for(int i =0,j = arr.length-1;i<arr.length/2;i++,j--){
            int temp = a[j];
            a[j] = a[i];
            a[i]=temp;

}
```

```java
3. 写出二维数组初始化的两种方式
int[][] arr = new int[][]{{1,2},{3,4}};

int[][] arr1 = new int[2][3];
int[][] arr2 = new int[3][];


4. 代码实现二维数组的遍历.
int[][] arr = new int[4][3];

for (int i = 0;i < arr.length ;i++ ){
    for (int j = 0;j < arr[i].length;j++ ){
        System.out.print(arr[i][j]);
    }
}

5. 二维数组元素的默认初始化值的理解
int[][] arr1 = new int[2][3];

```

1.数组的说明：

```
    1. 定义：是多个相同类型数据一定顺序排列的集合，并使用一个名字命名，并通过编号的方式对这些数据进行统一管理。
 * 2. 相关概念：
 *     ① 元素
 *     ② 下标、角标、索引
 *     ③ 数组名
 *     ④ 数组的长度
 * 3. 数组，属于引用数据类型。数组的元素，可以是基本数据类型，也可以是引用数据类型
 * 4. 数组的分类：
 *       ①按照维度：一维数组、二维数组、三维数组、...
 *       ②按照元素的类型：基本数据类型变量的数组、引用数据类型变量的数组
 *
 * 5.数组一旦初始化完成，则数组的长度就确定了。长度一旦确定，就不可更改。
 *
 * 6.数组中的元素是在内存空间中连续存储的。
```

2.一维数组的声明与初始化

正确的方式：

```
        int[] arr1;//声明
        //静态初始化:数组的初始化和数组元素的赋值同时进行。
        arr1 = new int[]{1,2,3,4};
        //合并声明和初始化
        int[] arr2 = new int[]{4,5,6,7};
        //或：int arr2[] = new int[]{4,5,6,7};
//      或：int[] arr2 = {4,5,6,7}; //类型推断

        //动态初始化：数组的初始化和数组元素的赋值分开进行。
        String[] arr3 = new String[3];
```

错误的声明方式：

```
//      int[] arr4 = new int[4]{1,2,3,4};
//      int[4] arr4 = new int[]{1,2,3,4};
```

排序算法

排序：假设含有n个记录的序列为{R1，R2，...,Rn},其相应的关键字序列为{K1，K2，...,Kn}。将这些记录重新排序为{Ri1,Ri2,...,Rin},使得相应的关键字值满足条Ki1<=Ki2<=...<=Kin,这样的一种操作称为排序。

- 通常来说，排序的目的是快速查找。

衡量排序算法的优劣：

1.时间复杂度：分析关键字的比较次数和记录的移动次数

2.空间复杂度：分析排序算法中需要多少辅助内存

3.稳定性：若两个记录A和B的关键字值相等，但排序后A、B的先后次序保持不变，则称这种排序算法是稳定的。

3.5 Arrays工具类的使用

尚硅谷

java.util.Arrays类即为操作数组的工具类，包含了用来操作数组（比如排序和搜索）的各种方法。

1	**boolean equals(int[] a,int[] b)**	判断两个数组是否相等。一系列重载的方法。
2	**String toString(int[] a)**	输出数组信息。一系列重载的方法。
3	**void fill(int[] a,int val)**	将指定值填充到数组之中。一系列重载的方法。
4	**void sort(int[] a)**	对数组进行排序。一系列重载的方法。
5	**int binarySearch(int[] a,int key)**	对排序后的数组进行二分法检索指定的值。
6	**copyOf(原数组,新数组长度)**	复制数组，从下标0开始复制，复制指定参数的个数
7	**copyOfRange(原数组,from,to)**	复制数组任意部分，从from到to（不包含）的元素

```
/*
 * Arrays:
 * 1. java.util包下
 * 2. Arrays:操作数组的工具类，提供了一系列操作数组的方法
 * 3. 常见方法：
 * boolean equals(int[] a,int[] b)
 * String toString(int[] a)
 * void fill(int[] a,int val)
 * void sort(int[] a)
 * int binarySearch(int[] a,int key)
 * copyOf(原数组,新数组长度)
 * copyOfRange(原数组,from,to)
```

学习面向对象内容的三条主线

1.java类及类的成员

2.面向对象的三大特征

3.其它关键字

```
/*
 * 一、面向对象学习的三条主线：
 * 1. java中类及类的成员：成员变量（或属性）、方法（函数）、构造器（或构造方法）；代码块（或初始化块）、内部类
 * 2. 面向对象的特征：封装性、继承性、多态性、(抽象性)
 * 3. 其它关键字的使用：this、super、abstract、interface、package、import、。。。
 *
 *  “大处着眼，小处着手”
 *
 */
```

何谓“面向对象”的编程思想？

首先解释一下“思想”。
先问你个问题：你想做个怎样的人？
可能你会回答：我想做个好人，孝敬父母，尊重长辈，关爱亲朋……
你看，这就是思想。这是你做人的思想，或者说，是你做人的原则。
做人有做人的原则，编程也有编程的原则。这些编程的原则呢，就是编程思想。

顿悟？ OR 渐悟？

让天下没有难学的技术

名称	修改日期	类型	大小
Java核心技术	2018/11/29 星期...	文件夹	
Java语言程序设计-基础篇(原书第8版)	2018/11/29 星期...	文件夹	
《Effective Java中文版 第2版》.(Joshua Bloch).[PDF]&cko...	2018/12/2 星期...	Adobe Acrobat ...	59,588 KB
Java编程那些事儿.pdf	2012/9/14 星期...	Adobe Acrobat ...	6,447 KB
Java编程思想第4版.pdf	2018/11/29 星期...	Adobe Acrobat ...	2,104 KB
Java程序员的基本修养.pdf	2014/11/30 星期...	Adobe Acrobat ...	72,785 KB
Java程序员修炼之道.pdf	2017/6/18 星期...	Adobe Acrobat ...	62,189 KB
编写高质量代码：改善Java程序的151个建议.pdf	2012/11/24 星期...	Adobe Acrobat ...	13,052 KB
数学之美与浪潮之巅合集.pdf	2012/8/8 星期三 ...	Adobe Acrobat ...	9,095 KB

4.1 面向过程与面向对象

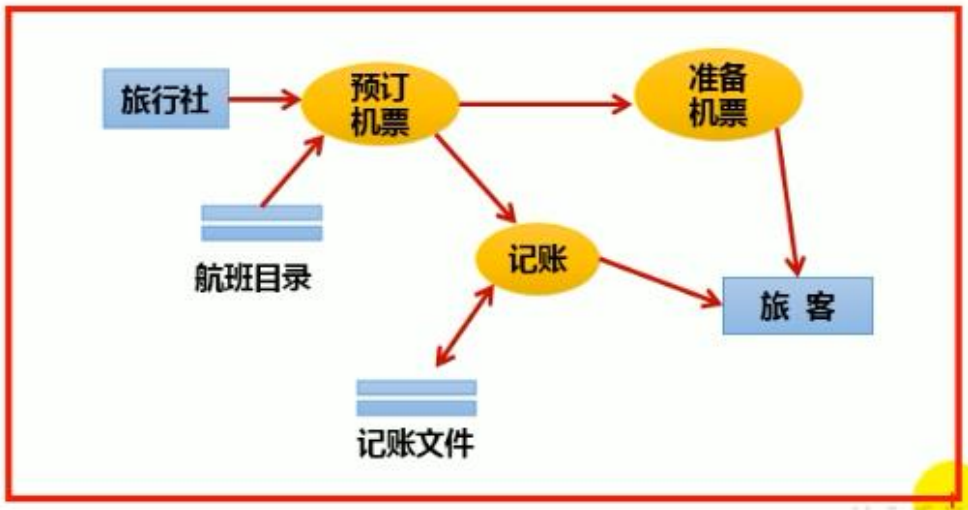
- **面向过程(POP) 与 面向对象(OOP)**
  - 二者都是一种思想，面向对象是相对于面向过程而言的。面向过程，强调的是功能行为，以函数为最小单位，考虑怎么做。面向对象，将功能封装进对象，强调具备了功能的对象，以类/对象为最小单位，考虑谁来做。
  - 面向对象更加强调运用人类在日常的思维逻辑中采用的思想方法与原则，如抽象、分类、继承、聚合、多态等。
- **面向对象的三大特征**
  - 封装 (Encapsulation)
  - 继承 (Inheritance)
  - 多态 (Polymorphism)

面向对象：Object Oriented Programming
面向过程：Procedure Oriented Programming

面向对象的思想概述

- 程序员从面向过程的执行者转化成了面向对象的指挥者
- 面向对象分析方法分析问题的思路和步骤
  - 根据问题需要，选择问题所针对的现实世界中的实体。
  - 从实体中寻找解决问题相关的属性和功能，这些属性和功能就形成了概念世界中的类。
  - 把抽象的实体用计算机语言进行描述，形成计算机世界中类的定义。即借助某种程序语言，把类构造成计算机能够识别和处理的数据结构。
  - 将类实例化成计算机世界中的对象。对象是计算机世界中解决问题的最终工具。

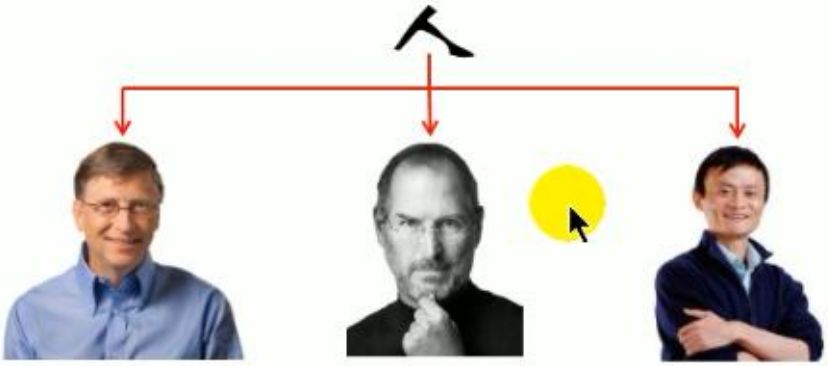
3. 抽象出下面系统中的“类”及其关系。



面向对象的思想概述

- 类(class)和对象(object)是面向对象的核心概念。
  - 类是对一类事物的描述，是抽象的、概念上的定义
  - 对象是实际存在的该类事物的每个个体，因而也称为实例(instance)。
- “万事万物皆对象”

面向对象的思想概述



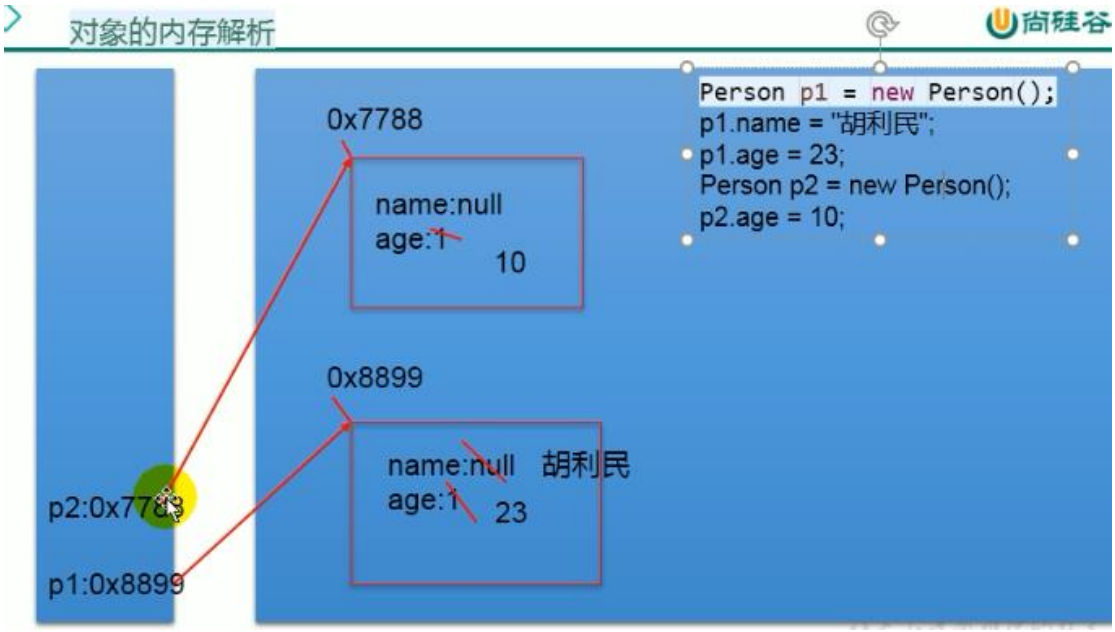
- 可以理解为：类 = 抽象概念的人；对象 = 实实在在的某个人
- 面向对象程序设计的重点是类的设计
- 定义类其实是定义类中的成员(成员变量和成员方法)

Java类及类的成员

- 现实世界的生物体，大到鲸鱼，小到蚂蚁，都是由最基本的细胞构成的。同理，Java代码世界是由诸多个不同功能的类构成的。
- 现实生物世界中的细胞又是由什么构成的呢？细胞核、细胞质、... 那么，Java中用类class来描述事物也是如此
  - 属 性：对应类中的成员变量
  - 行 为：对应类中的成员方法

Field = 属性 = 成员变量，Method = (成员)方法 = 函数

```java
/*
 * 一、类的设计:创建类，并提供类中的结构：属性、方法
 *
 * 属性 - 成员变量 - field - （域 - 字段）
 * 方法 - 成员方法 - 函数 - method
 *
 * 创建类的对象 - 实例化类 - 类的实例化
 *
 * 二、类的实例化及功能的调用（面向对象的落地的实现一）：
 * 1. 创建类及提供类的成员：属性、方法
 * 2. 类的实例化（创建类的对象）
 * 3. 调用对象的相关结构:"对象.属性" 或 "对象.方法"
 *
 *
 * 三、说明
 * 1. 如果创建一个类的多个对象，则每个对象独立拥有一份类中声明的属性。如果修改对象a中的属性值，不会影响对象b中
 * 相同属性的值。
 */
```



```
3  * 类中属性(或成员变量)的声明和使用
4  *
5  * 成员变量 vs 局部变量
6  *
7  * 相同点：
8  * 1. 声明变量的结构：数据类型 变量名 = 变量值。
9  * 2. 先声明，后使用
10 * 3. 都有对应的作用域
11 *
12 * 不同点：
13 * 1. 类中声明的位置不同。
14 * 成员变量：直接声明在类中
15 *
16 *
17 *
18 *
19 *
20 * 回顾：分类一：按照数据类型分：基本数据类型（8种） vs 引用数据类型（数组、类、接口）
21 *      分类二：按照类中声明的位置： 成员变量 vs 局部变量
22 *
```

```
12 * 不同点：
13 * 1. 类中声明的位置不同。
14 * 成员变量：直接声明在类中
15 * 局部变量：方法的形参、方法的内部、构造器的内部、构造器的形参、代码块的内部、...
16 *
17 *
18 *
19 *
20 * 回顾：分类一：按照数据类型分：基本数据类型（8种） vs 引用数据类型（数组、类、接口）
21 *      分类二：按照类中声明的位置： 成员变量 vs 局部变量
22 *
23 */
```

```
12 * 不同点：
13 * 1. 类中声明的位置不同。
14 * 成员变量：直接声明在类中
15 * 局部变量：方法的形参、方法的内部、构造器的内部、构造器的形参、代码块的内部、...
16 *
17 * 2. 权限修饰符的使用
18 * 成员变量声明前，可以使用权限修饰符进行修饰。而局部变量不可以使用。
19 * 常见的权限修饰符有：public \ private \ 缺省
20 *
21 *
```

对象属性的默认初始化赋值

当一个对象被创建时，会对其中各种类型的成员变量自动进行初始化赋值。除了基本数据类型之外的变量类型都是引用类型，如上面的Person及前面讲过的数组。

成员变量类型	初始值
byte	0
short	0
int	0
long	0L
float	0.0F
double	0.0
char	0 或写为:'\u0000'(表现为空)
boolean	false
引用类型	null


```
22  * 成员变量有默认初始化值。
23  *       整型（byte \ short \ int \ long ）：0
24  *       浮点型（float \ double）：0.0
25  *       字符型（char）：'\u0000'
26  *       布尔型（boolean）：false
27  *       引用数据类型：null
28  *
29  *
30  * 而局部变量没有默认初始化值，在调用之前，必须显式初始化
31  *
32  * 4. 内存中加载的位置不同
33  *   局部变量，声明在栈空间中
34  *   成员变量，声明在堆空间中
35  *
```

```
2  /*
3   * 类中方法的声明和使用
4   *
5   * public void show(){}
6   *  int getAge(){}
7   * private void showNation(String nation){}
8   * public String eat(String food1,String food2,String food3){}
9   *
10  * 1. 方法声明的格式：
11  * 权限修饰符  返回值类型 方法名(形参列表){
12  *       //方法体
13  * }
14  *
15  * 说明：我们定义方法时，都需要考虑如上方法声明的五部分结构！！
16  *
17  */
```

```
17  * 2. 细节说明：
18  * 2.1  权限修饰符：指明所修饰的结构可被调用的范围的大小。
19  *       技巧：暂时大家声明方法时，都使用public.
20  *
21  * 2.2 返回值类型：  具体的数据类型 或 void（没有返回值）
22  *       ① 我们定义方法时，根据方法的实际需要，来决定是否需要有返回值。如果需要，则指明返回值对应的数据类型。
23  *          如果不需要，则使用void来声明
24  *       ② 如果指明了具体的数据类型，则必须在方法执行最后，返回所要求的数据类型的变量或常量，使用return
25  *       ③ 方法体中一旦执行到return，则结束此方法。
26  *       ④ 声明为void的方法中，也可以使用return。
27  *
28  * 2.3 方法名，属于标识符，声明时满足标识符的命名规则和规范。
29  *     起名时，要“见名知意”
30  *     Arrays:binarySearch() / sort() /...
31  *     Math:random() / round()
```

```
16  * 2. 细节说明：
17  * 2.1  权限修饰符：指明所修饰的结构可被调用的范围的大小。
18  *       技巧：暂时大家声明方法时，都使用public.
19  *
20  * 2.2 返回值类型：  具体的数据类型 或 void（没有返回值）
21  *       ① 我们定义方法时，根据方法的实际需要，来决定是否需要有返回值。如果需要，则指明返回值对应的数据类型。
22  *          如果不需要，则使用void来声明
23  *       ② 如果指明了具体的数据类型，则必须在方法执行最后，返回所要求的数据类型的变量或常量，使用return
24  *       ③ 方法体中一旦执行到return，则结束此方法。
25  *       ④ 声明为void的方法中，也可以使用return。
26  *
27  * 2.3 方法名，属于标识符，声明时满足标识符的命名规则和规范。
28  *     起名时，要“见名知意”
29  *     Arrays:binarySearch() / sort() /...
30  *     Math:random() / round()
31  * 2.4 形参列表：可以根据需要，声明方法时，提供形参列表。
32  *     如果需要声明多个形参，使用逗号,隔开。
33  */
```

4.4 类的成员之二：方 法

什么是方法(method、函数):

- 方法是类或对象行为特征的抽象，用来完成某个功能操作。在某些语言中也称为函数或过程。
- Java里的方法不能独立存在，所有的方法必须定义在类里。

语法格式：

```
修饰符 返回值类型 方法名 ( 参数列表) {
        方法体语句;
}
```

举例：

```
public class Person{
   private int age;
   public int getAge() { return age; } //声明方法getAge
   public void setAge(int i) {        //声明方法setAge
            age = i;       //将参数i的值赋给类的成员变量age
   }
}
```