课前：

请利用插入排序升序排列数组：arr1={2,13,5,8,3,21,1}

**int**[] arr1 = {2,13,5,8,3,21,1}; 创建数组

**for**(**int** x = 0;x<=arr.length-1;x++){ 外循环规定哪个数作为插入数

**for**(**int** y = x+1;y > 0 && y <= arr.length-1; y--){ 内循环规定判断次数

**if**(arr[y]<arr[y-1]){ 判断需不需要插入

**int** temp; 互换，进行插入

temp = arr[y];

arr[y] = arr[y-1];

arr[y-1] = temp;

}

}

}

**for**(**int** i = 0;i<=arr.length-1;i++){

System.***out***.println(arr[i]); 遍历打印

}

选择排序降序arr1

**for**(**int** i=0;i<=arr.length-1;i++){ 每一个位置都要和后面所有的值进行一次选择

**int** max = arr[i]; 定义当前位置的数为最大值

**int** maxPo = i; 定义当前位置为最大值

**for**(**int** j=i+1;j<=arr.length-1;j++){ 遍历之后的所有数

**if**(max<a[j]){ 判断，找到最大值

max = a[j];

maxPo = j;

}

}

int temp;

temp = arr[i]; 与找到的最大值互换，保证最前面的一定是最大的

arr[i] = arr[maxpo];

arr[maxpo] = temp;

}

二分查找排序号的arr1中的13在哪个位置: 注意必须是已经排好序的数组

int n = 0; 定义初位置

int m = arr.length-1; 定义末位置

int p = (m+n)/2; 定义二分位置

int i= 13; 定义要查找的数

while(i!=arr[p]){ 判断是否查找到

if(i<arr[p]){

n = p+1; 重设初位置

}

If(i>arr[p]){

m = p-1; 重设末位置

}

P = (m+n)/2; 重设二分位置

}

System.out.println(p);

方法与类

方法的创建：

1. 方法就是一段可重复调用的代码段

2. 定义格式：

访问修饰符 返回值类型 方法名（）{

方法主体

}

方法我将它理解为一种操作，当我们使用一种方法，我们是在解决一个问题。那么解决问题的方法是什么？

一种操作需要有输出，没有输出就没有意义；除非这个方法本身就定义为不需要输出，因为输出以其他形式

例：

**public** **void** tell(){//两个单词的命名，第二个单词首字母大写

System.***out***.println("Hello Person");

}

**public** **static** **int** giveAge(){

**return** 17;

}

**public** **static** String print(){

**return** "hello";

}

**public** **static** String giveName(String n){

**return** n;

}

**public** **static** String giveAge(**int** i,String n){

**return** n+i;

}

方法的调用：

1. 对象.方法名（）；

2. 方法名（参数）

方法的重载：

方法名称相同，但是参数的类型和个数不同，通过传递参数的个数和类型的不同来完成不同的功能。

调用方法的办法对数组进行插入排序

**public** **class** ArrayTest {

**public** **static** **void** main(String[] args) {

**int**[] arr1 = {2,13,5,8,3,21,1};

*insertionSort*(arr1);

}

**public** **static** **void** insertionSort(**int**[] arr){

**for**(**int** i = 0;i<=arr.length-1;i++){

**for**(**int** j = i+1;j>0&&j<=arr.length-1;j--){

**if**(arr[j]<arr[j-1]){

**int** temp;

temp = arr[j];

arr[j] = arr[j-1];

arr[j-1] = temp;

}

}

}

**for**(**int** x = 0;x<=arr.length-1;x++){

System.***out***.println(arr[x]);

}

}

}

作业：用调用方法的办法，对数组进行选择排序

java对象和类：

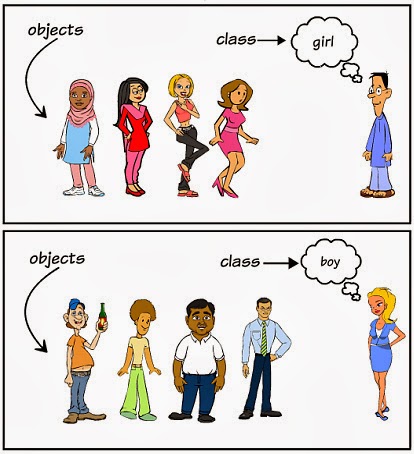
Java作为一种面向对象语言。支持以下基本概念：

* 多态
* 继承
* 封装
* 抽象
* 类
* 对象
* 实例
* 方法
* 重载

**什么是对象和类？**

* **类**：类是一个模板，它描述一类对象的行为和状态。
* **对象**：对象是类的一个实例（**对象不是找个女朋友**），有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。

下图中男孩女孩为类，而具体的每个人为该类的对象：



**Java中的对象**

现在让我们深入了解什么是对象。看看周围真实的世界，会发现身边有很多对象，车，狗，人等等。所有这些对象都有自己的状态和行为。

拿一条狗来举例，它的状态有：名字、品种、颜色，行为有：叫、摇尾巴和跑。

对比现实对象和软件对象，它们之间十分相似。

软件对象也有状态和行为。软件对象的状态就是属性，行为通过方法体现。

在软件开发中，方法操作对象内部状态的改变，对象的相互调用也是通过方法来完成。

**Java中的类**

类可以看成是创建Java对象的模板。

1. class类名称{

属性

方法

}

2. 声明一个类需要通过一个关键字class

通过下面一个简单的类来理解下Java中类的定义：

public class Person{

String name;

int age;

String occupation;

void working(){ }

void eating(){ }

void relaxing(){ } }

一个类可以包含以下类型变量：

* **局部变量**：在方法、构造方法或者语句块中定义的变量被称为局部变量。变量声明和初始化都是在方法中，方法结束后，变量就会自动销毁。
* **成员变量**：成员变量是定义在类中，方法体之外的变量。这种变量在创建对象的时候实例化。成员变量可以被类中方法、构造方法和特定类的语句块访问。
* **类变量**：类变量也声明在类中，方法体之外，但必须声明为static类型。

一个类可以拥有多个方法，在上面的例子中：working()、eating()和relaxing()都是Person类的方法。

**构造方法**

每个类都有构造方法。如果没有显式地为类定义构造方法，Java编译器将会为该类提供一个默认构造方法。

在创建一个对象的时候，至少要调用一个构造方法。构造方法的名称必须与类同名，一个类可以有多个构造方法。

下面是一个构造方法示例：

**class** Person{

String name;

**int** age;

String occupation;

**public** Person(){};

**public** Person(String n, **int** i){

name = n;

age = i;

};

**public** Person(String n, **int** i, String j){

name = n;

age = i;

occupation = j;

}

**public** **void** showPerson(){

System.***out***.println("姓名："+name+""+"年龄："+age);

}

}

**创建对象**



对象是根据类创建的。在Java中，使用关键字new来创建一个新的对象。创建对象需要以下三步：

* **声明**：声明一个对象，包括对象名称和对象类型。
* **实例化**：使用关键字new来创建一个对象。
* **初始化**：使用new创建对象时，会调用构造方法初始化对象。

下面是一个创建对象的例子：

public static void main(String []args){ // 下面的语句将创建一个Person对象

Person p1 = **new** Person("tony",17);

内存分析：

**访问实例变量和方法**

通过已创建的对象来访问成员变量和成员方法，如下所示：

/\* 实例化对象 \*/ ObjectReference = new Constructor();

/\* 访问其中的变量 \*/ ObjectReference.variableName;

/\* 访问类中的方法 \*/ ObjectReference.MethodName();

例子：

Person p2 = **new** Person();

p2.name = "Peter";

p2.age = 17;

p2.showPerson();

**实例**

**class** Person{

String name;

**int** age;

String occupation;

**public** Person(){};

**public** Person(String n, **int** i){

name = n;

age = i;

};

**public** Person(String n, **int** i, String j){

name = n;

age = i;

occupation = j;

}

**public** **void** showPerson(){

System.***out***.println("姓名："+name+""+"年龄："+age);

}

}

**public** **class** classTest {

**public** **static** **void** main(String[] args) {

Person p1 = **new** Person("tony",17);

Person p2 = **new** Person();

p1.showPerson();

p2.name = "Peter";

p2.age = 17;

p2.showPerson();

//方法介绍

**int** i = 17;

**int** j = 19;

String n = "Tony";

System.***out***.println(*giveAge*());

System.***out***.println(*giveAge*(i,n));//静态调用必须调用静态方法

System.***out***.println(*giveAge*(i,j,10));

}

**public** **static** **void** greet(){//两个单词的命名，第二个单词首字母大写

System.***out***.println("Hello everyone");

}

**public** **static** **int** giveAge(){

**return** 17;

}

**public** **static** String print(){

**return** "hello";

}

**public** **static** String giveName(String n){

**return** n;

}

**public** **static** String giveAge(**int** i,String n){

**return** n+i;

}

**public** **static** **int** giveAge(**int** i, **int** j, **int** n){

**return** i+j+n;

}

}