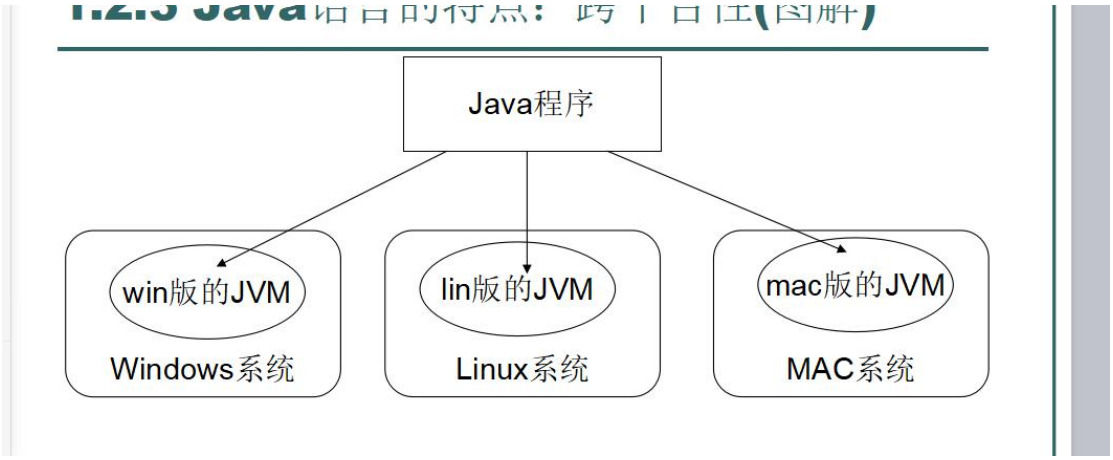
跨平台性

只要在需要运行 java 应用程序的操作系统上，
先安装一个 Java 虚拟机(JVM Java Virtual Machine)即可。
由 JVM 来负责 Java 程序在该系统中的运行。



JRE Java Runtime Environment

JDK Java development kit

使用 JDK 开发完成的 java 程序，交给 JRE 去运行

Javac.exe----------compile

Java 运行

```
D:\javafile>javac Document1.java

D:\javafile>java Demo
错误: 在类 Demo 中找不到 main 方法, 请将 main 方法定义为:
   public static void main(String[] args)
否则 JavaFX 应用程序类必须扩展javafx.application.Application
```

```
C:\Users\yangshanqi>set classpath=D:\javafile

C:\Users\yangshanqi>java Demo
hello java

C:\Users\yangshanqi>set classpath=

C:\Users\yangshanqi>set classpath
环境变量 classpath 没有定义
```

Classpath 设置如何 class 被系统可以找到

先找 classpath，再找当前目录？。。。。（加；的情况下）

```
D:\java0217\day01>set classpath=c:\;

D:\java0217\day01>java Demo
hello java

D:\java0217\day01>java Demo
hello D盘

D:\java0217\day01>set classpath=c:\

D:\java0217\day01>java Demo
Exception in thread "main" java.lang.NoClassDefFoundError: Demo
Caused by: java.lang.ClassNotFoundException: Demo
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:248)
Could not find the main class: Demo.  Program will exit.
```

没加；情况，只找 class path， 不找当前目录（推荐）
Path 现在当前目录底下找，没找到去 path 环境变量下

Java 语言组成

关键字 标识符 注释 变量常量

严格区分大小写
main 不是关键字，被 jvm 所识别的名称

包名：多单词组成时所有字母都小写。

xxxyyyzzz

类名接口名：多单词组成时，所有单词的首字母大写。

XxxYyyZzz

变量名和函数名：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写。

xxxYyyZzz

常量名：所有字母都大写。多单词时每个单词用下划线连接。

XXX_YYY_ZZZ

进制转换

System。Out。Println（0x3c）　前面是零

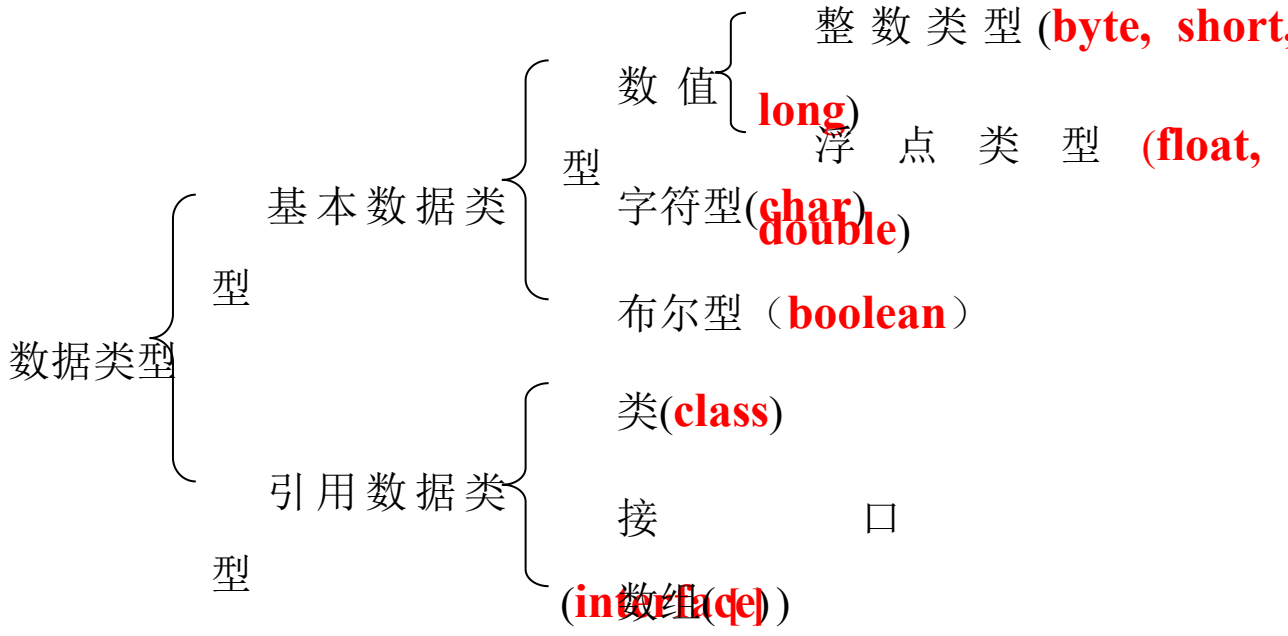
进制转化

```
System.out.println(Integer.toBinaryString(11111111111111111111111111111
11));
```

负数二进制最高位是 0 ？？？？对于固定位数的情况下

变量：内存中放什么，里面数值可以变，但是类型不变

Java：强类型语言，对于类型划分十分细致



```
int x=6;
System.out.println(x);
```

Float 单精度 double 双精度

Float d=2.3f (后面加一个 f 表示是单精度的数值？必须。。。)

变量上要非常注意 Java 的强类型

```
byte b=2;
        b=b+2;。。。。。。。。。。。。会报错
```

可行的？

```
byte b=2;
int c=7;
c=b+2;
System.out.println(c);
```

强类型====什么样的可以赋值到那儿去。。。

强制转化

```
byte b=2;
b=(byte)(b+2);
System.out.println(b);
```

写出完整的表达式

```
int b=7;
        System.out.println("b="+b);
```

转义符号

= , +=, -=, *=, /=, %=

算术运算符

赋值运算符

比较运算符

逻辑运算符

位运算符

三元运算符

“&”和“&&”的区别：

单&时，左边无论真假，右边都进行运算；

双&时，如果左边为真，右边参与运算，如果左边为假，那么右边不参与运算。

“|”和“||”的区别同理，双或时，左边为真，右边不参与运算。

<<	空位补 0，被移除的高位丢弃，空缺位补 0。
>>	被移位的二进制最高位是 0，右移后，空缺位补 0； 最高位是 1，空缺位补 1。
>>>	被移位二进制最高位无论是 0 或者是 1，空缺位都用 0 补。
&	二进制位进行&运算，只有 1&1 时结果是 1，否则是 0;

|	二进制位进行 | 运算，只有 0 | 0 时结果是 0，否则是 1;
^	任何相同二进制位进行 ^ 运算，结果是 0；1^1=0 , 0^0=0 不相同二进制位 ^ 运算结果是 1。1^0=1 , 0^1=1

1.最有效率的方式算出 2 乘以 8 等于几？2<<3;

2.对两个整数变量的值进行互换(不需要第三方变量)

```
int a=7;
int b=8;
a=a+b;
b=a-b;
a=a-b;
System.out.println(a);
System.out.println(b);
```

还有一个异或的方式

三元运算符：格式

(条件表达式)?表达式 1：表达式 2；

如果条件为 true，运算后的结果是表达式 1；

如果条件为 false，运算后的结果是表达式 2；

示例：

获取两个数中大数。

int x=3,y=4,z;

z = (x>y)?x:y;//z 变量存储的就是两个数的大数。

条件语句

```
if(条件表达式)
    {
        执行语句；
    }

if(条件表达式)
    {
        执行语句；
    }
    else
    {
        执行语句；
    }
```

switch 语句

格式：

```
switch(表达式)
{
    case  取值 1:
        执行语句；
        break；
    case  取值 2:
        执行语句；
        break；
    …...
    default:
        执行语句；
        break；
}
```

while 语句格式：

```
while(条件表达式)
{
    执行语句；
}
```

do while 语句格式：

```
do
{
    执行语句；
}while(条件表达式);
```

```
for(初始化表达式；循环条件表达式；循环后的操作表达式)
{
    执行语句；
}
```

关于变量作用范围　　报错

```
  int x=0;
for (int x=5;x<10;x++)
{
System.out.println(x);
}
System.out.println(x);
```

```
class Str
{
        public static void main(String[] args)
        {
          int x=1;
          while (x<101){
                if (x%7==0)
                {System.out.println(x);          }
                x++;
          }
        }

}
```

break(跳出)，　continue(继续，结束本次循环，继续下次循环)

break 语句：应用范围：选择结构和循环结构。

continue 语句：应用于循环结构。

函数

修饰符 返回值类型 函数名(参数类型 形式参数 1，参数类型 形式参数 2，)

{

执行语句;

return 返回值;

}

```
public static void main(String[] args)
{
int a=3;
System.out.println(getResult(a));
}

public static int getResult(int x)
{
return x*3+333;
}
```

重载的概念
在同一个类中，允许存在一个以上的同名函数，只要它们的参数个数或者参数类型不同即可。

```
int a=3;
int b=2;
int result = add(a,b);
double c=3.4;
double result1=add(a,c);

System.out.println(result);
System.out.println(result1);
}

public static int add(int x,int y)
{
        return x+y ;
}

public static double add(double x, double y)
{
        return x+y;
}

 public static int add(int x, int y, int z)
 {
    return x+y+z;
 }
```

重载和返回值没有关系

数组：

Java 程序在运行时，需要在内存中的分配空间。为了提高运算效率，有对空间进行了不同区域的划分，因为每一片区域都有特定内存管理方式。

栈内存

用于存储局部变量，当数据使用完，所占空间会自动释放。

堆内存

数组和对象，通过 new 建立的实例都存放在堆内存中。

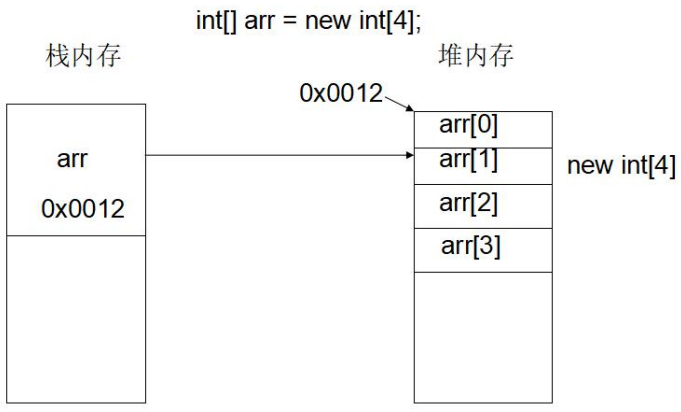
每一个实体都有内存地址值

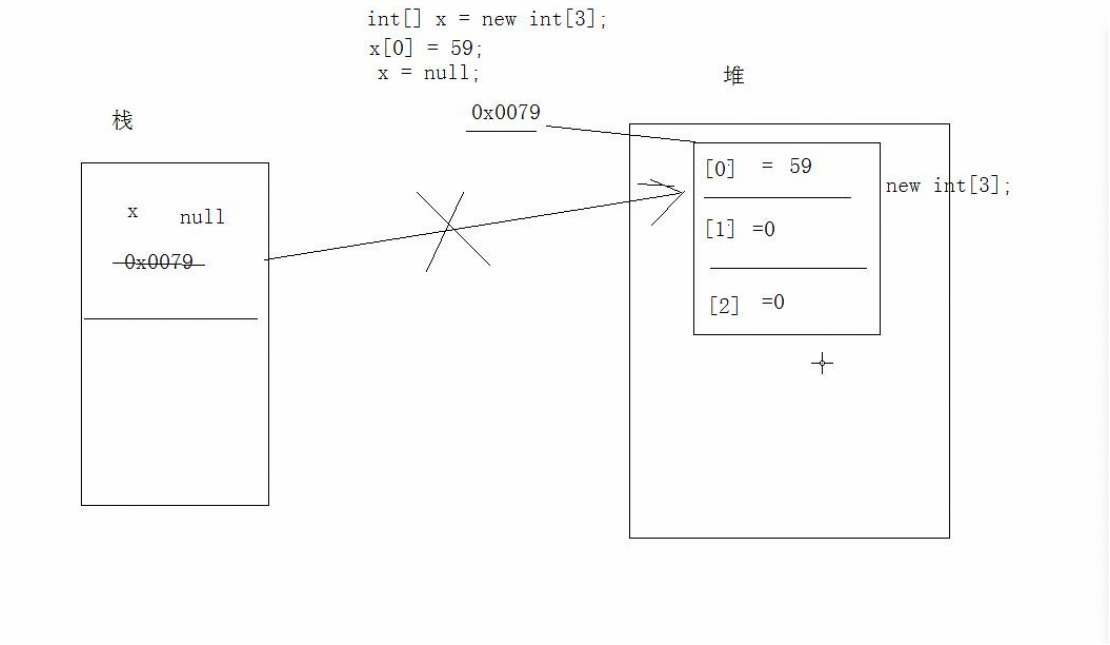
实体中的变量都有默认初始化值

实体不在被使用，会在不确定的时间内被垃圾回收器回收

方法区，本地方法区，寄存器

.8.2 数组内存结构





Java 内存优化 （garbage collection）做的比 c++ 好？。。。。

数组脚标越界异常(ArrayIndexOutOfBoundsException)

`int[] arr = new int[2];`
 	
`System.out.println(arr[3]);`
 	
访问到了数组中的不存在的脚标时发生。

空指针异常(NullPointerException)

`int[] arr = null;`
 	
`System.out.println(arr[0]);`
 	
arr 引用没有指向实体，却在操作实体中的元素时。

获取最值(最大值，最小值)

排序（选择排序，冒泡排序）

折半查找(二分查找)

面向对象

人开门 ---------- 门。开

面向对象是相对面向过程而言

面向对象和面向过程都是一种思想

面向过程

强调的是功能行为

面向对象

将功能封装进对象，强调具备了功能的对象。

面向对象是基于面向过程的。

//面向对象：三个特征：封装，继承，多态。

//以后开发：其实就是找对象使用。没有对象，就创建一个对象。

//找对象，建立对象，使用对象。维护对象的关系。

/*
类和对象的关系。

现实生活中的对象：张三 李四。
想要描述：提取对象中共性内容。对具体的抽象。
描述时：这些对象的共性有：姓名，年龄，性别，学习 java 功能。

映射到 java 中，描述就是 class 定义的类。
具体对象就是对应 java 在堆内存中用 new 建立实体。

类就是：对现实生活中事物的描述。
对象：就是这类事物，实实在在存在个体。

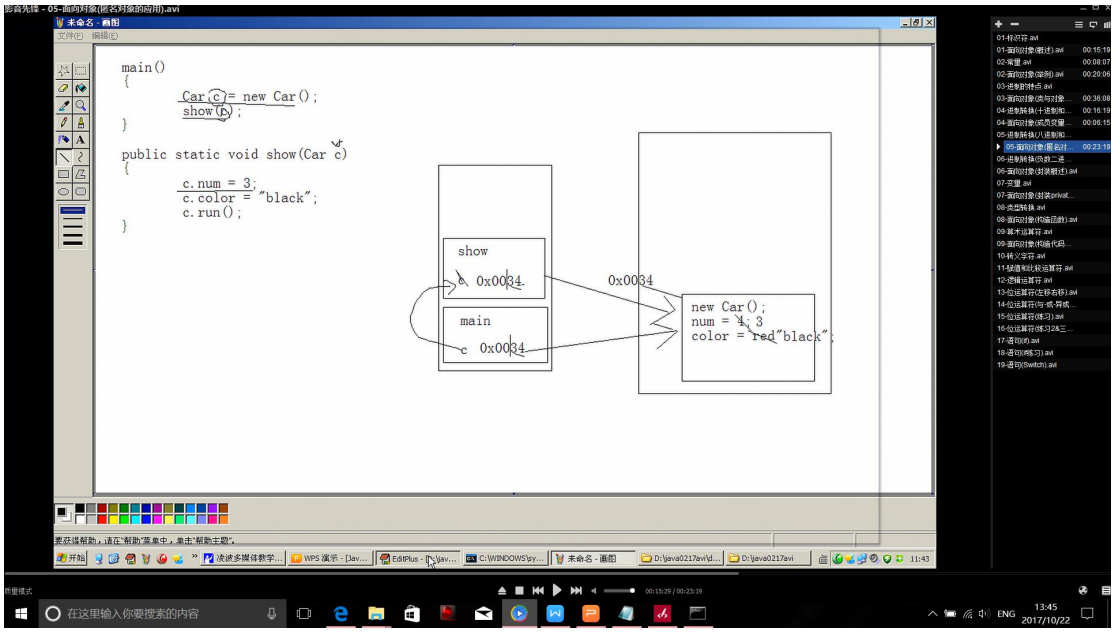
*/

开发的过程：其实就是不断的创建对象，使用对象，指挥对象做事情。
设计的过程：其实就是在管理和维护对象之间的关系。

面向对象的特征：
封装(encapsulation)
继承(inheritance)
多态(polymorphism)

//属性对应是类中变量，行为对应的类中的函数(方法)。
//其实定义类，就是在描述事物，就是在定义属性和行为。属性和行为共同成为类中的成员(成员变量和成员方法)。

匿名对象
匿名对象是对象的简化形式
匿名对象两种使用情况
当对对象方法仅进行一次调用的时
匿名对象可以作为实际参数进行传递



封装(Encapsulation)

封装：是指隐藏对象的属性和实现细节，仅对外提供公共访问方式。

好处：

将变化隔离。

便于使用。

提高重用性。

提高安全性。

封装原则：

将不需要对外提供的内容都隐藏起来。

把属性都隐藏，提供公共方法对其访问。

private 关键字：

是一个权限修饰符。

用于修饰成员(成员变量和成员函数)

被私有化的成员只在本类中有效。

常用之一：

将成员变量私有化，对外提供对应的 set ，get 方法对其进行访问。提高对数据访问的安全性。

构造函数（类似于自己来个初始值？。。。）

特点：

函数名与类名相同

不用定义返回值类型

不可以写 return 语句

作用：

给对象进行初始化。

注意：

默认构造函数的特点。

多个构造函数是以重载的形式存在的。

```
class Person2
{
    private int age;
    private String name;

         Person2()
        {
                System.out.println("name = "+name+" ,age = "+age);
        }

    Person2(String a)
        {
            name = a;
                System.out.println("name = "+name+" ,age = "+age);
        }

}
class demo
{
        public static void main(String[] args)
        {
                Person2 p1 = new Person2();
                Person2 p2 = new Person2("Mike");
        }
}
```

构造代码块

```
/*
    构造代码块。
    作用：给对象进行初始化。
    对象一建立就运行，而且优先于构造函数执行。
    和构造函数的区别：
    构造代码块是给所有对象进行统一初始化，
    而构造函数是给对应的对象初始化。

    构造代码快中定义的是不同对象共性的初始化内容。

    */
```

This 关键字

/*

this:看上去，是用于区分局部变量和成员变量同名情况。

this 为什么可以解决这个问题？

this 到底代表的是什么呢？

this：就代表本类的对象，到底代表哪一个呢？

this 代表它所在函数所属对象的引用。

简单说：哪个对象在调用 this 所在的函数，this 就代表哪个对象。

this 的应用：当定义类中功能时，该函数内部要用到调用该函数的对象时，这时用 this 来表示这个对象。但凡本类功能内部使用了了本类对象，都用 this 表示。

```
*/
class Person3
{
    private int age;
        private String name;
        {
                System.out.println("This program starts.");
        }

         Person3(String name)
         {
                this.name = name;
                System.out.println("name="+name);
         }

         Person3(int age)
         {
                this.age = age;
         }

         public boolean compare (Person3 p)
         {
                return this.age==p.age ;
         }
}
class PersonDemo3
{
        public static void main(String[] args)
        {
                Person3 p1 = new Person3(21);
                Person3 p2 = new Person3(21);
                System.out.println(p1.compare(p2));
        }
}
/*
this 语句 :用于构造函数之间进行互相调用。

this 语句只能定义在构造函数的第一行。因为初始化要先执行。
*/

class Person
{
     private String name;
```

```
	private int age;

	{

		System.out.println("code run");
	}

	Person()
	{
		//this("hah");
		System.out.println("person run");
	}
	Person(String name)
	{
		//this();
		this.name =name;
	}
	Person(String name,int age)
	{
		//this(name);
		//this.name = name;
		this.age = age;

	}

}

class	PersonDemo4
```

Static 用于节约内存空间？。。。。

```
/*
静态：static。
用法：是一个修饰符，用于修饰成员(成员变量，成员函数).
当成员被静态修饰后，就多了一个调用方式，除了可以被对象调用外，
还可以直接被类名调用。类名.静态成员。

static 特点：
1，随着类的加载而加载。
	也就说：静态会随着类的消失而消失。说明它的生命周期最长。
```

2，优先于的对象存在
明确一点：静态是先存在。对象是后存在的。

3，被所有对象所共享
4，可以直接被类名所调用。

实例变量和类变量的区别：
1，存放位置。
　　类变量随着类的加载而存在于方法区中。
　　实例变量随着对象的建立而存在于堆内存中。
2，生命周期：
　　类变量生命周期最长，随着类的消失而消失。
　　实例变量生命周期随着对象的消失而消失。

静态使用注意事项：
1，静态方法只能访问静态成员。
　　非静态方法既可以访问静态也可以访问非静态。
2，静态方法中不可以定义 this，super 关键字。
　　因为静态优先于对象存在。所以静态方法中不可以出现 this。
3，主函数是静态的。

静态有利有弊
利处：对对象的共享数据进行单独空间的存储，节省空间。没有必要每一个对象中都存储一份。
　　可以直接被类名调用。
弊端：生命周期过长。
　　访问出现局限性。(静态虽好，只能访问静态。)
*/

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

主函数：是一个特殊的函数。作为程序的入口，可以被 jvm 调用。

主函数的定义：
public：代表着该函数访问权限是最大的。
static：代表主函数随着类的加载就已经存在了。
void：主函数没有具体的返回值。
main：不是关键字，但是是一个特殊的单词，可以被 jvm 识别。
（String[] arr）:函数的参数，参数类型是一个数组，该数组中的元素是字符串。字符串类型的数组。

主函数是固定格式的：jvm 识别。

jvm 在调用主函数时，传入的是 new String[0];
*/

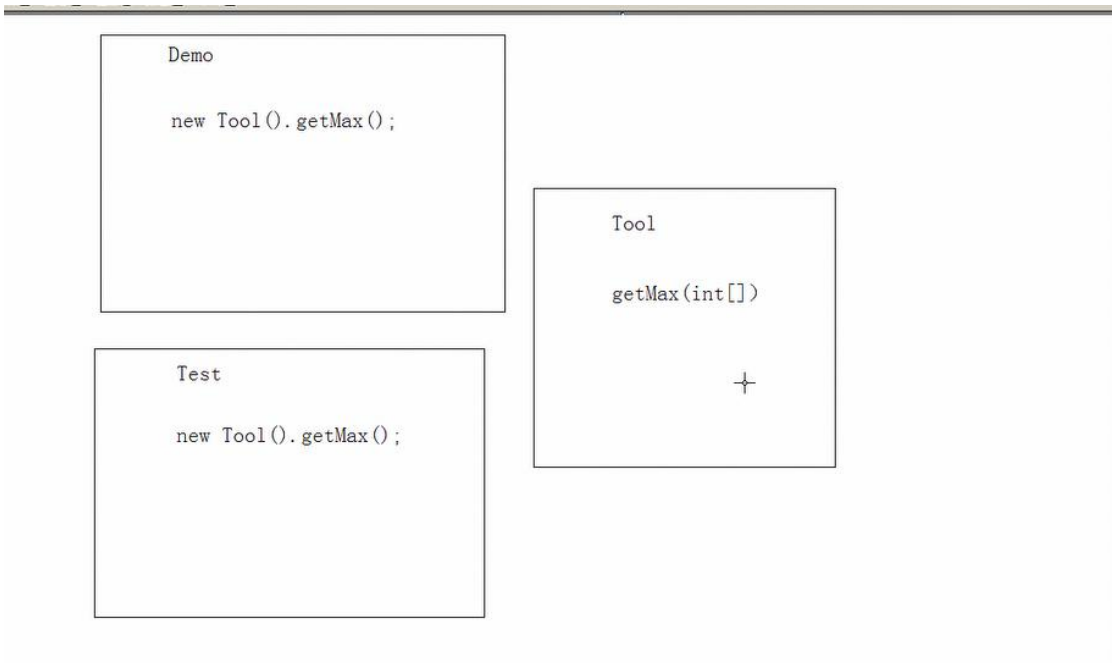
/*
什么使用静态？

要从两方面下手：
因为静态修饰的内容有成员变量和函数。
什么时候定义静态变量(类变量)呢？
当对象中出现共享数据时，该数据被静态所修饰。
对象中的特有数据要定义成非静态存在于堆内存中。

什么时候定义静态函数呢？

当功能内部没有访问到肺静态数据(对象的特有数据)，
那么该功能可以定义成静态的。

*/




```
class ArrayTool
{
   static int getMax(int[] arr)
   {
        int max=0;
        for (int i=1; i<arr.length;i++)
          if (arr[i]>arr[max])
```

```
            max =i;
        return arr[max];
    }

        void sort(int[] arr)
        {
         for(int x=0; x<arr.length; x++)
           { for (int y=x+1; y<arr.length; y++)
                 {if (arr[x] >arr[y])
                 swap (arr,x,y);}
                }
        }

        void arrPrint(int[] arr)
        {
                for (int x=0; x<arr.length; x++)
                {System.out.print(arr[x]+"  ");}
        }

        static void swap(int[] arr,int x, int y)
        {
                int temp = arr[x];
                arr[x] = arr[y];
                arr[y] = temp;
        }
```

*

静态的应用。

每一个应用程序中都有共性的功能，
可以将这些功能进行抽取，独立封装。
以便复用。

虽然可以通过建立 ArrayTool 的对象使用这些工具方法，对数组进行操作。
发现了问题：
1，对象是用于封装数据的，可是 ArrayTool 对象并未封装特有数据。
2，操作数组的每一个方法都没有用到 ArrayTool 对象中的特有数据。

这时就考虑，让程序更严谨，是不需要对象的。
可以将 ArrayTool 中的方法都定义成 static 的。直接通过类名调用即可。

将方法都静态后，可以方便于使用，但是该类还是可以被其他程序建立对象的。
为了更为严谨，强制让该类不能建立对象。
可以通过将构造函数私有化完成。

接下来，将 ArrayTool.class 文件发送给其他人，其他人只要将该文件设置到 classpath 路径下，就可以使用该工具类。

但是，很遗憾，该类中到底定义了多少个方法，对方去不清楚。因为该类并没有使用说明书。

开始制作程序的说明书。java 的说明书通过文档注释来完成。
*/

Javadoc 制作
只提供 public 的部分，不会提供 private 部分

静态代码块：用于对类进行初始化，类进入内存的时候，执行一次
构造代码块：用于对象初始化
构造函数：（相对于构造代码块带了参数）

*********初始化过程 day 6 .07 视频*****************
/*
Person p = new Person("zhangsan",20);

该句话都做了什么事情？
1，因为 new 用到了 Person.class.所以会先找到 Person.class 文件并加载到内存中。
2，执行该类中的 static 代码块，如果有的话，给 Person.class 类进行初始化。
3，在堆内存中开辟空间，分配内存地址。
4，在堆内存中建立对象的特有属性。并进行默认初始化。
5，对属性进行显示初始化。
6，对对象进行构造代码块初始化。
7，对对象进行对应的构造函数初始化。
8，将内存地址付给栈内存中的 p 变量。

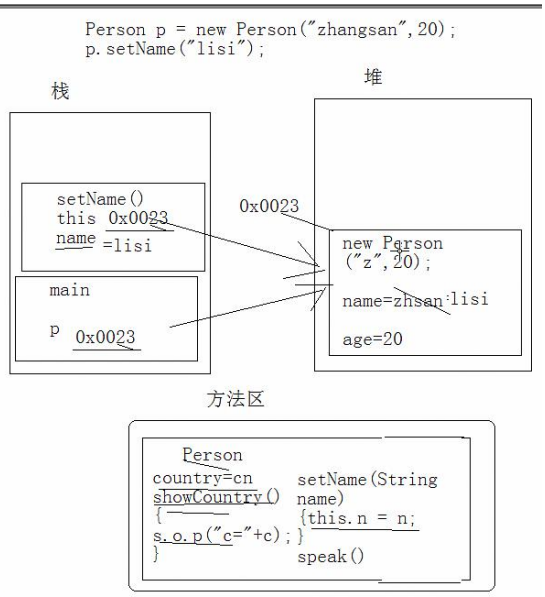
*/

```
class Person
{
  private String name;
  private int age;
  private static  String country = "cn";
  Person(String name,int age)
  {
          this.name = name;
          this.age = age;
  }

  public void setName(String name)
  {
          this.name = name;
  }

  public void speak()
  {
          System.out.println(this.name+"..."+this.age)
  }

  public static void  showCountry()
  {
          System.out.println("country="+country);
  }
}
```



设计模式

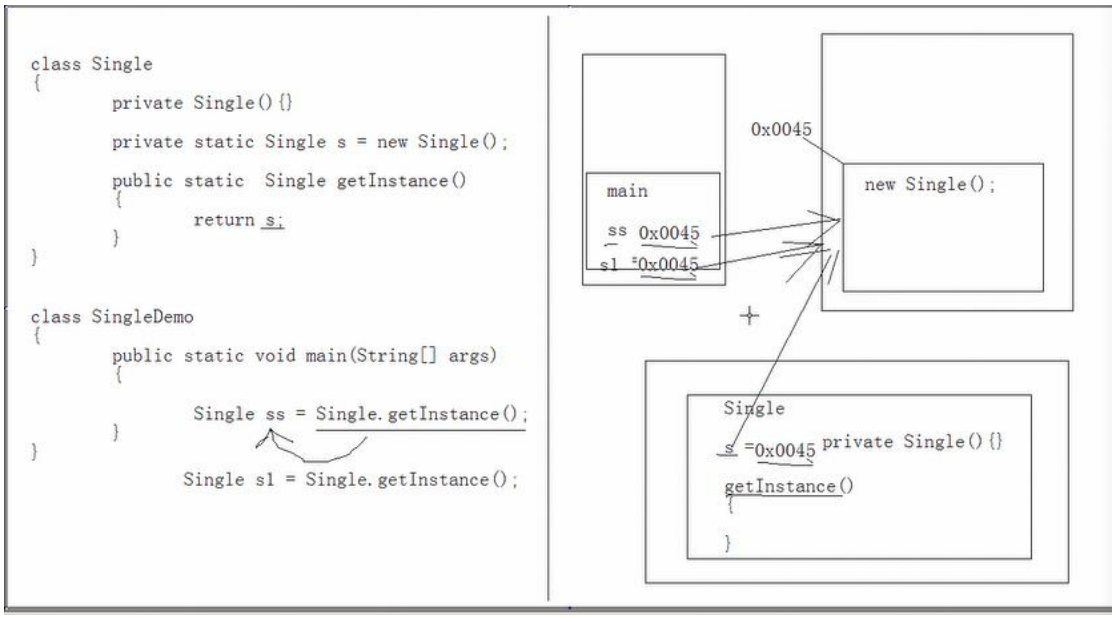
```
/*
设计模式：解决某一类问题最行之有效的方法。
java 中 23 种设计模式：
单例设计模式：解决一个类在内存只存在一个对象。

想要保证对象唯一。
1，为了避免其他程序过多建立该类对象。先禁止其他程序建立该类对象
2，还为了让其他程序可以访问到该类对象，只好在本类中，自定义一个对象。
3，为了方便其他程序对自定义对象的访问，可以对外提供一些访问方式。

这三部怎么用代码体现呢？
1，将构造函数私有化。
2，在类中创建一个本类对象。
3，提供一个方法可以获取到该对象。

对于事物该怎么描述，还怎么描述。
当需要将该事物的对象保证在内存中唯一时，就将以上的三步加上即可。
*/
class Single
{     private    Single(){}
      private static Single s = new Single();
      public static    Single getInstance()
      {
            return s;
```

```
    }
}
class SingleDemo
{
    public static void main(String[] args)
    {
        Single s1 = Single.getInstance();
        Single s2 = Single.getInstance();
```



懒汉式，饿汉式?...day 6 the last video ----unsawn