数组的定义与使用

组在java程序中，基本上每个程序当中都会使用数组，二基本的数组出现的记录不高，并且不会涉及到太多复杂的操作。

1.数组的基本概念
如果说要定义100个整型变量。比如int i1,i2……
这种方式可以定义，但是这种模式出来的变量就不适合程序维护，因为没有任何的参考规律。
数组的本质在于一组相关变量的集合，但是在java里面，将数组定义为了引用数据类型，所以数组会牵扯到内存分配。那么首先会想到关键字newe来处理

数组的定义格式
数组的动态初始化（初始化之后所有的元素保存内容为其对应内容的默认值）；
声明并初始化数组：
数据类型 数组名称 [] = new 数据类型 [长度] ;
数组的静态初始化（在数组定义的时候就为其设置好了里面的内容）;
简化格式： 数据类型 数组名称 = {数据1,数据2,数据3} [] ;
完整格式：[] = new 数据类型 {数据1,数据2,数据3} [] ;

当初创建了一个数组之后就可以按照如下方式进行使用；

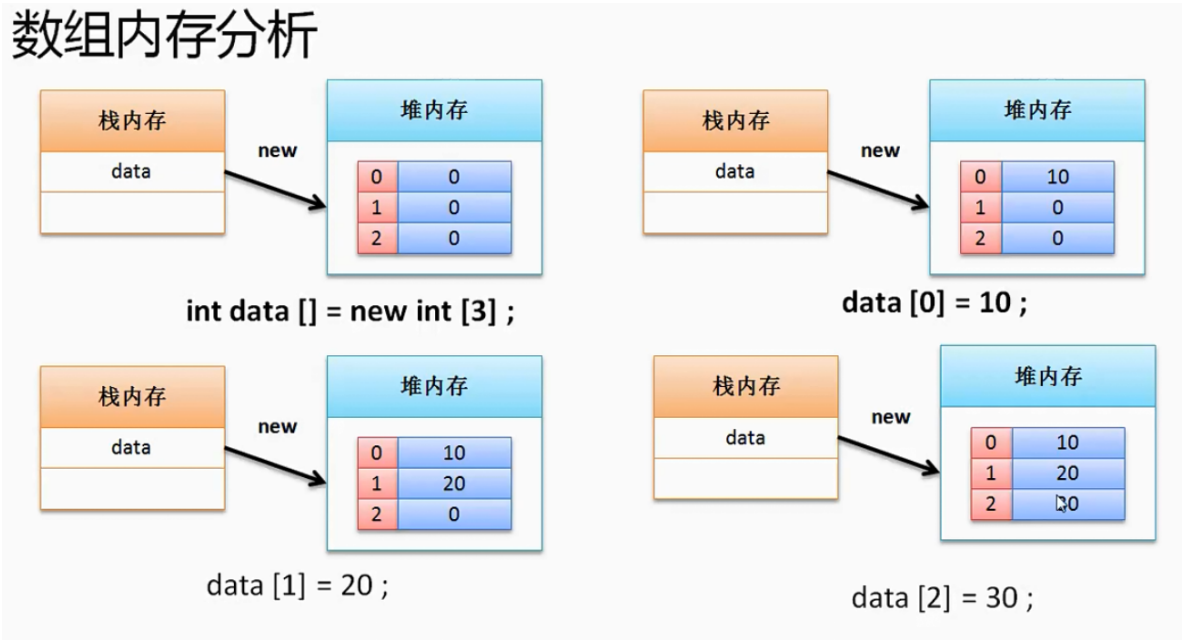
数组里面可以通过脚标进行每一个元素的访问，脚标从0开始定义，所以可以使用的脚标范围：“0~数组长度-1”
同事如果超过了脚标的范围，则会出现

“ArrayIndexOutOfBoundsException（数组越界）” 异常
使用数组是为了方便变量进行管理，所以进行数组操作的时候会往往利用for循环来完成

对于数组的长度也可以使用“数组名称.length”属性进行获得。

范例：定义数组

```
public class JavaStudy {
   public static void main(String args[]) {
      //使用数组的动态初始化实现了数组的定义
      int data [] = new int [3] ;
      data [0] = 11 ; //为数组设置内容
      data [1] = 12 ; //为数组设置内容
      data [2] = 13 ; //为数组设置内容
      for (int x = 0 ; x < data.length ; x ++){
         System.out.println(data[x]) ;
      }
   }
}
```



在以后的开发过程中，见到的代码最多的就是 “数组的循环处理”

```
for (int x = 0 ; x < data.length ; x ++){
        System.out.println(data[x]) ;
    }
```

数组本身分为动态初始化和静态初始化，上面使用的是动态初始化，会发现数组之中的每一个元素内容，都对应数组的默认值。

如果不想这么复杂或者不希望这么做，想让数组在定义的时候就已经可以提供内容，可以采用静态初始化完成。

范例：使用静态初始化进行定义

```
public class JavaStudy {
   public static void main(String args[]) {
      //使用数组的静态初始化
      int data [] = new int [] {11,12,13} ;
      for (int x = 0; x < data.length ; x ++ ) {
         System.out.println(data[x]) ;
      }
   }
}
```

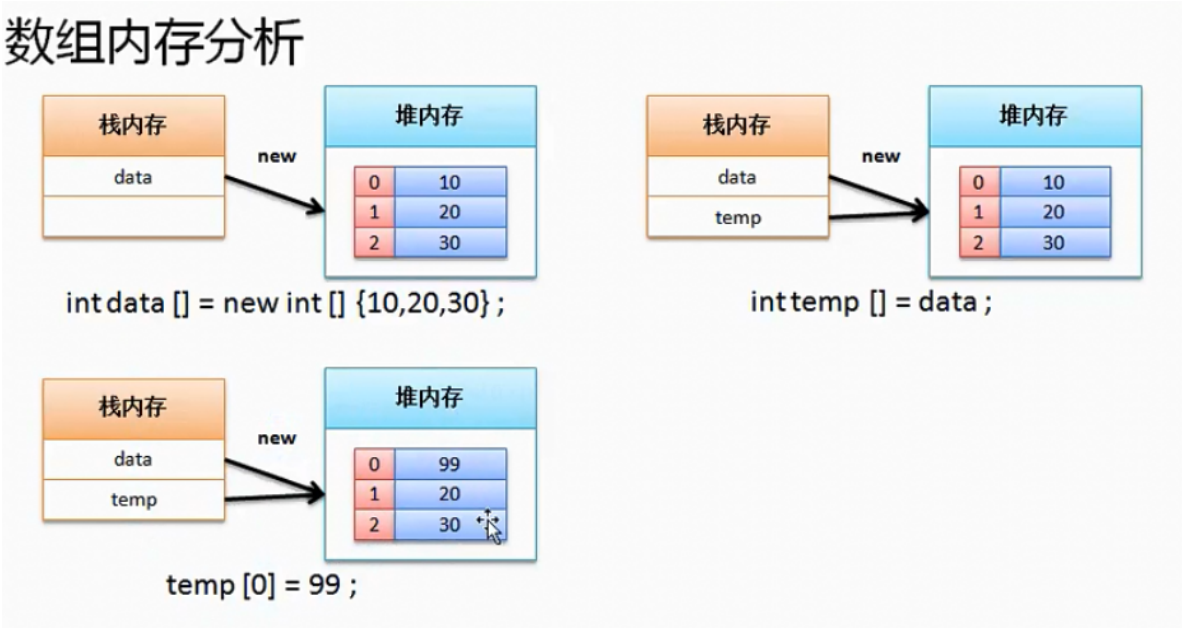
对于数组而言，基本上都是拿到数据后循环控制

2.数组引用传递分析

通过数组的基本定义可以发现，在数组使用过程总依然需要关键字new进行开辟，那么一定会定义内存的关系匹配

```
public class JavaStudy {
   public static void main(String args[]) {
      //使用数组的静态初始化
      int data [] = new int [3] ;
      data [0] = 10 ;   //为数组设置内容
      data [1] = 11 ;
      data [2] = 12 ;
      for (int x = 0; x < data.length ; x ++) {
         System.out.println(data[x]) ;
      }
   }
```

}



但是数组本身毕竟是属于引用数据类型，那么既然是引用数据类型，就一定会发生引用传递。

引用传递：一个堆内存可以被多个占内存所指向

范例：观察数组引用

```
public class JavaStudy {
   public static void main(String args[]) {
      //使用数组的静态初始化
      int data [] = new int [3] ;
      data [0] = 10 ;   //为数组设置内容
      data [1] = 11 ;
      data [2] = 12 ;
      for (int x = 0; x < data.length ; x ++) {
         System.out.println(data[x]) ;
      }
   }
}
```

由于数组属于引用类型，所以一定为开辟堆内存空间之后才可以使用，如果现在使用了未开辟的内存会出现 “NullPointerException” 异样

必须提供有实例化对象才可以使用下表的形式进行数组的操作，否则会出现数据异样的操作