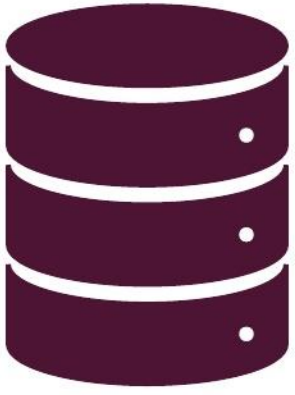


数据结构

WENYE 李
CUHK-SZ

大纲

计算机内存基础知识

部分超出了目前的范围。

几周后再回来详细讨论。 数组

实现

例子

记忆的结构

记忆的基本单位叫位，位是 0 或 1。

在大多数现代体系结构中，硬件运行的最小单位是一个由 8 个连续比特组成的序列，称为字节。

二进制(可执行)文件

0 1 2 3

01011001100001001001111011000011…

…

数字和指令存储在更大的单位中，大多是常见的一个单词。因为机器有不同的架构，一个单词的字节数和字节顺序因机器而异。

数字、基数和转换

(21)10
=(10101)2 (0.65625)10 =
(0.10101)2

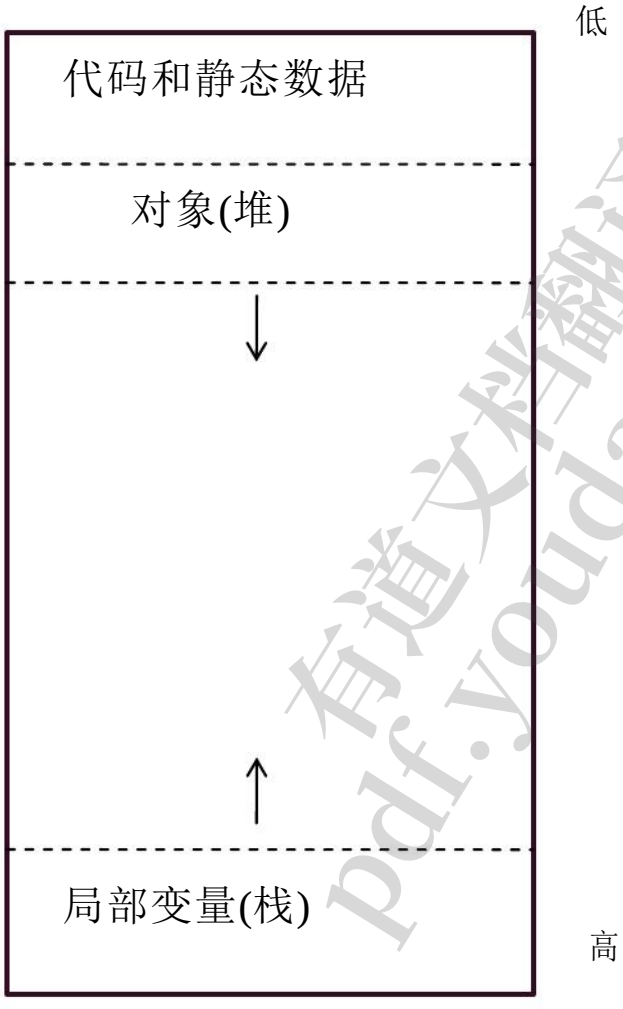
八进制
(0,1,2,3,4,5,6,7)

$(10101)_2 = (010101)_2 = (25)_8$
$(0.10101)_2 = (0.101010)_2 = (0.52)_8$

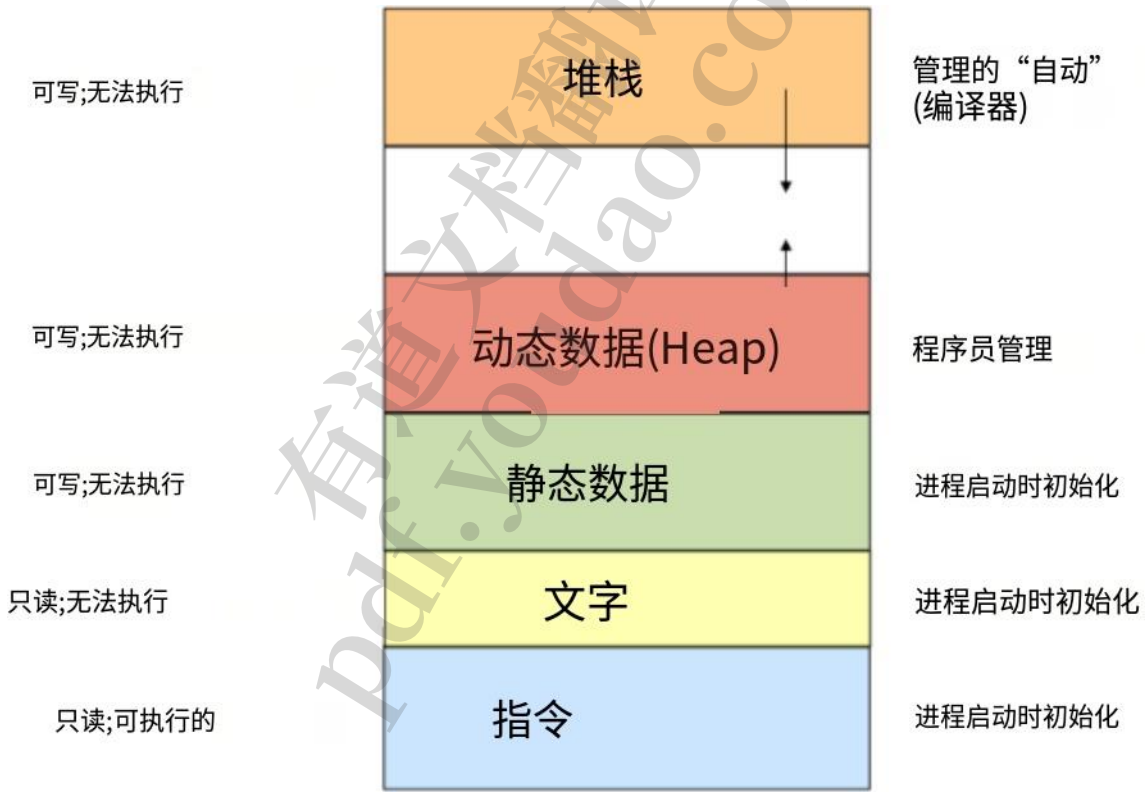
十六进制(0,1,2,3,4,5,6,7,8,9,A, B, C, D, E,
F) (10101) 2 = (00010101) 2 = (15) 16
(0.10101) 2 = (0.10101000) 2 = (0. a8) 16

有用的数字
(10000000000)2 =(1024)10(约 1K)

内存分配



如果你喜欢另一个方向



堆栈 VS 堆

参数	堆栈	堆
基本	内存在一个连续的块中分配。	内存以任意随机顺序分配。
分配和释放	由编译器指令自动完成。	手动由程序员完成。
成本	少	更多的
实现	容易	硬
访问时间	快	慢
主要问题	内存不足	记忆的碎片
参考的局部性	优秀的	足够的
安全	线程安全，存储的数据只能由所有者访问	不是线程安全的，存储的数据对所有线程可见
灵活性	固定大小	可以调整尺寸
数据类型结构	线性	分层

稍后将详细讨论。

给变量分配内存

内存的一个区域是为静态数据保留的。 在程序运行时从不创建或销毁，例如命名常量。

当创建一个新对象时，Java 会从堆中分配空间。

当一个方法被调用时，Java 会分配一个新的内存块，称为栈帧来保存它的本地变量。

当一个方法返回时，它的栈帧被擦除。栈帧来自栈。

对象引用

Java 通过对象在内存中的地址来标识对象。这个地址称为引用(reference)。

如。，当 Java 执行时

理性 a =新理性(1,2);它为新的 Rational 对象分配堆空间。对于这个例子，假设对象被分配在地址 1000 处。

局部变量 a 在当前栈帧中被分配，并被分配值(地址)，该值标识了对象。

```
公共类
{
    /**创建一个初始化为0的新Rational public Rational()
    {this(8);
    }
    /**
     * 从整数参数中创建一个新的Rational。@param n初始值
     */
    public Rational(int n)
    {this(n, 1);
    }
    /**
     * 创建一个值为x / y的新Rational @param x有理数的分子
     * @param y有理数的分母
     */
    public有理数(int x, int y) {
        int g = ged(Math.abs(×)，Math.abs(y));
        Num = x / g;
        den = Math.abs(y) / g;
        if (y<θ) num = -num;
    }
    /**
     * 将有理数r与此相加，返回求和。@param r要加的有理数
     * @return当前数与r的和
     */
    public Rational add(Rational r) {
        return new Rational(此。Num * r.den + r.num * this.den,
                            this.den * r.den);
    }
    /**
     * 从这个中减去有理数r，返回差值。
     *
     * @param r要减去的有理数@return当前数减去r的结果
     */
    public有理相减(有理r) {
        返回新的Rational(此。Num * r.den - r.num * this.den,
                            This.den * r.den);
    }
    /**
     * 将这个数乘以有理数r @param r用作乘数的有理数
     * @return当前数乘以r的结果
     */
    公有理乘(有理r) {
        return new Rational(this。Num * r.num, this.den * r.den);
    }
    /**
     * 这个数除以非零有理数r. @param r用作除数的非零有理数
     * @return当前数除以r的结果
     */
    public有理数除(有理数r) {
        返回new Rational(this。Num * r.den, this.den * r.m um);
    }
    /**
     * 创建这个有理数的字符串表示。@return该有理数的字符串
     * 表示形式
     */
    public String toString() {
        if (den == 1) {
            返回"          num;
        }其他{
            return num + "/" + den;
        }
    }
    /**
     * 用欧几里得算法计算最大公约数。
     * @param第一个整数
     *
     * @param第二个整数@return x和y的最大公约数
     */
    私有int gcd(int x, int y) {
        Int r = x % y;
        While (r != 8) {
            X = y;
            y = r;
            R = x% y;
        }
        返回y;
    }
    /**私有实例变量*/
    私有int num;                /*这个Rational的分子*/
    私人私人书房;               /*这个Rational的分母*/
} /* 类理性                    */
```

解决模型

1020

1000

FFB4

FFB8

FFBC

FFC0

堆栈

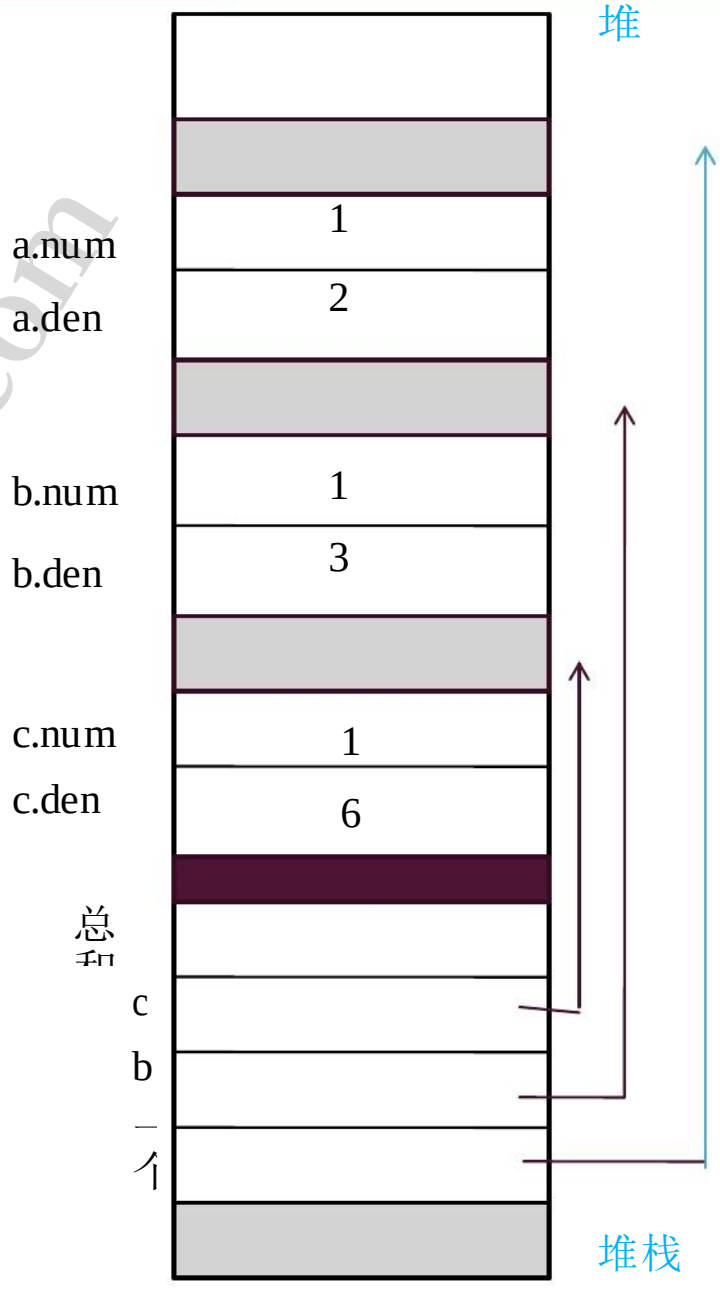
总和

c

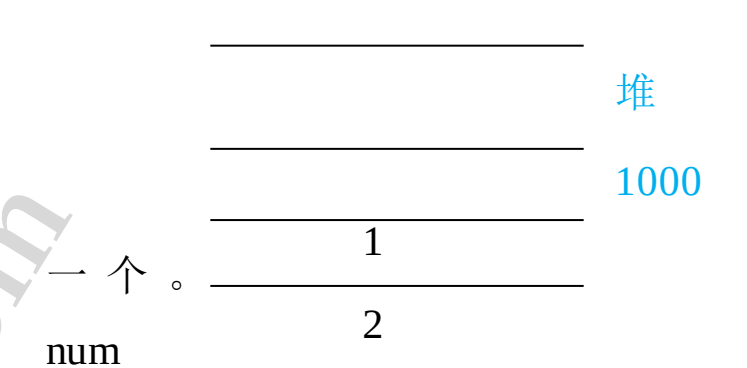
b

指针模型

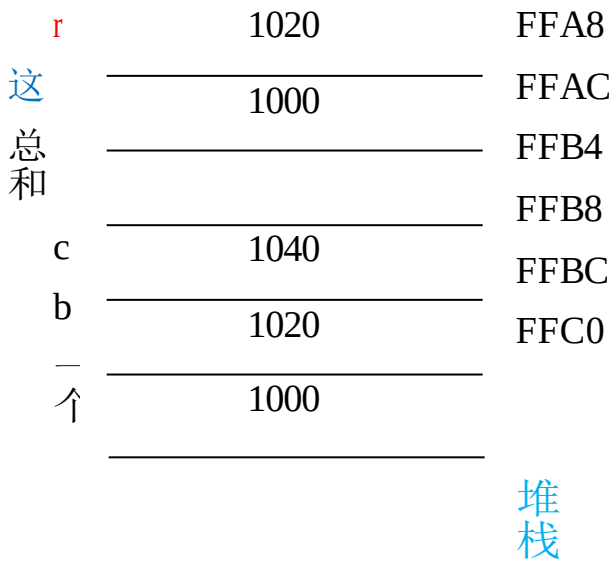
```
公共 void run() {
    理 性 a = 新 理 性 (1,2);
    理 性 b = 新 理 性 (1,3);
    理性 c =新理性(1,6);
    有 理 和 = (a.add(b)).add(c);
}
```



```
公共 void run() {
    理性 a =新理性(1,2);理性 b =新
    理性(1,3);理性 c =新理性(1,6);
    有理和= (a.add(b)).add(c);}
```




```
公共理性添加(理性 r) {
    返回新的理性(this.num*r.den +
                 r.num*this.den,
                 this.den*r.den);
}
```



b。 1

num 3.

b.den 1

6 1020

c。

num

c.den 1040

公共 void run() {

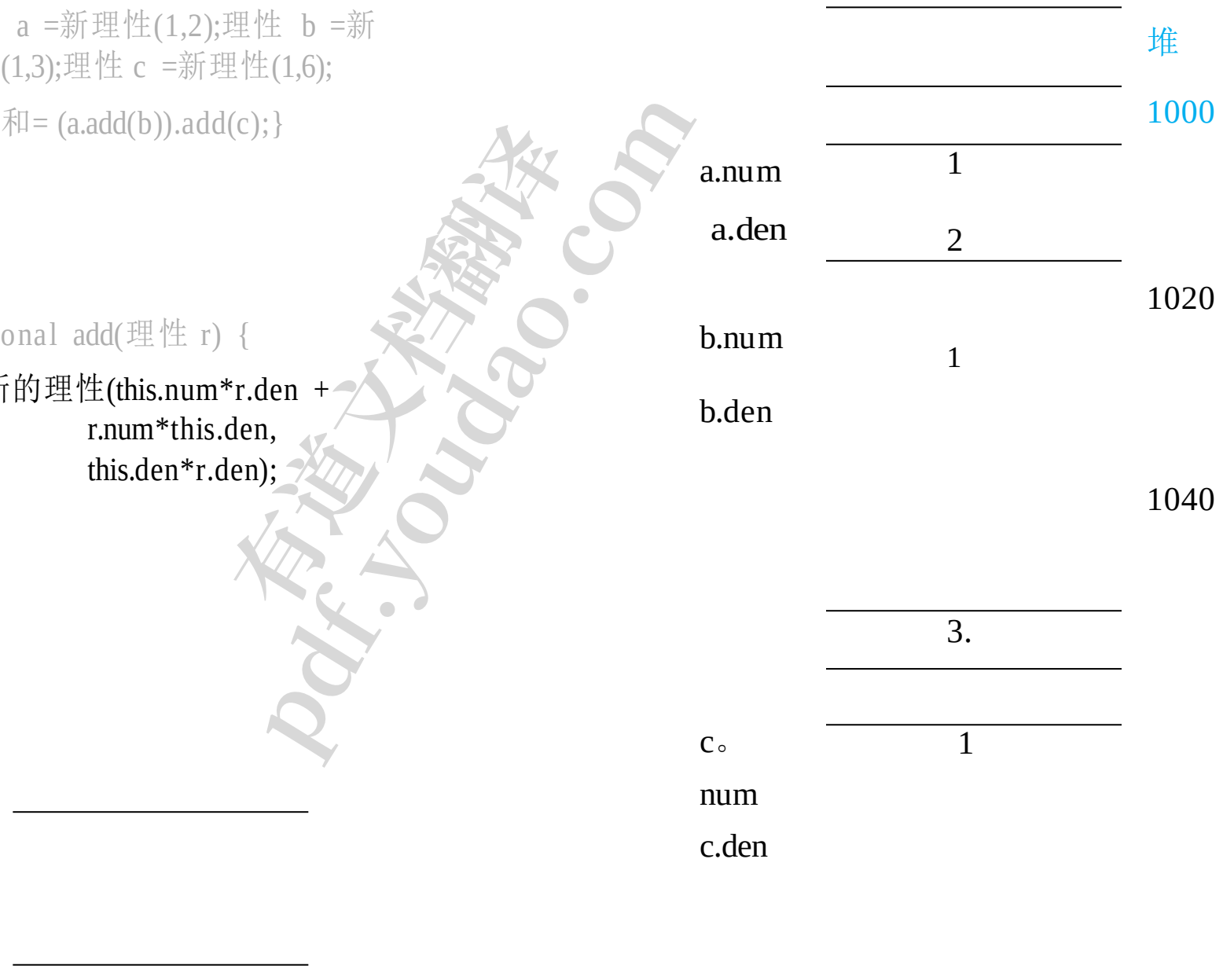
理性 a =新理性(1,2);理性 b =新理性(1,3);理性 c =新理性(1,6);

有理和= (a.add(b)).add(c);}

public Rational add(理性 r) {

返回新的理性(this.num*r.den +
r.num*this.den,
this.den*r.den);

}



1040	FFB4
1020	FFB8
1000	FFBC
	FFC0

(a.add (b))。.den num (a.add (b))

6
5
6

堆栈

总和

1060

c

b

公共 void run() {

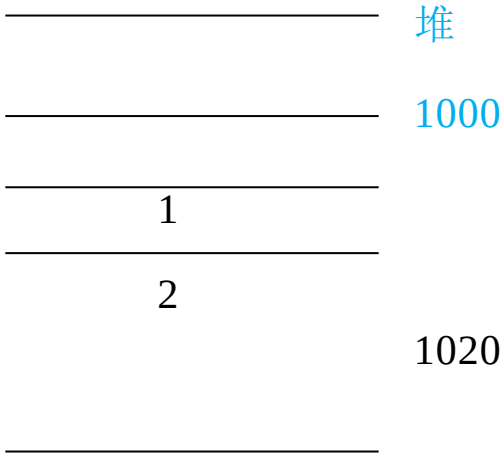
理性 a =新理性(1,2);理性 b =新理性(1,3);理性 c =新理性(1,6);

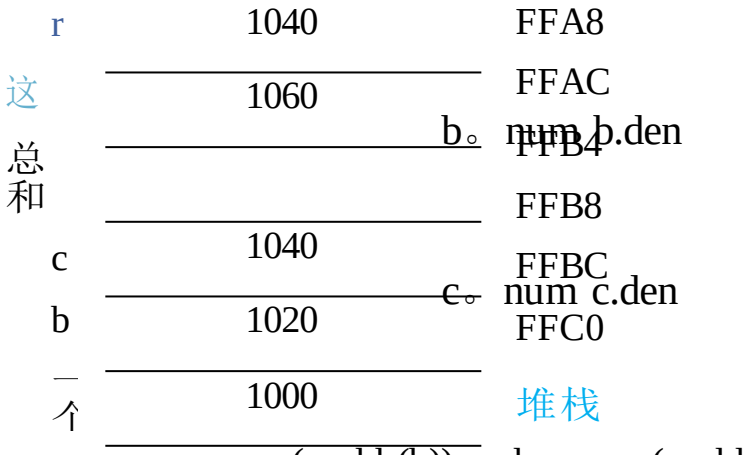
有理和= (a.add(b)).add(c);}

公共理性的添加(理性的 r) {

返回新的理性(this.num*r.den +
r.num*this.den,
this.den*r.den);

}





阀门(a.add (b)) (c)。阀门 num (a.add (b) (c) .den

6 1

1

3.

6

1

1040

1060

5

1080

1

公共 void run() {

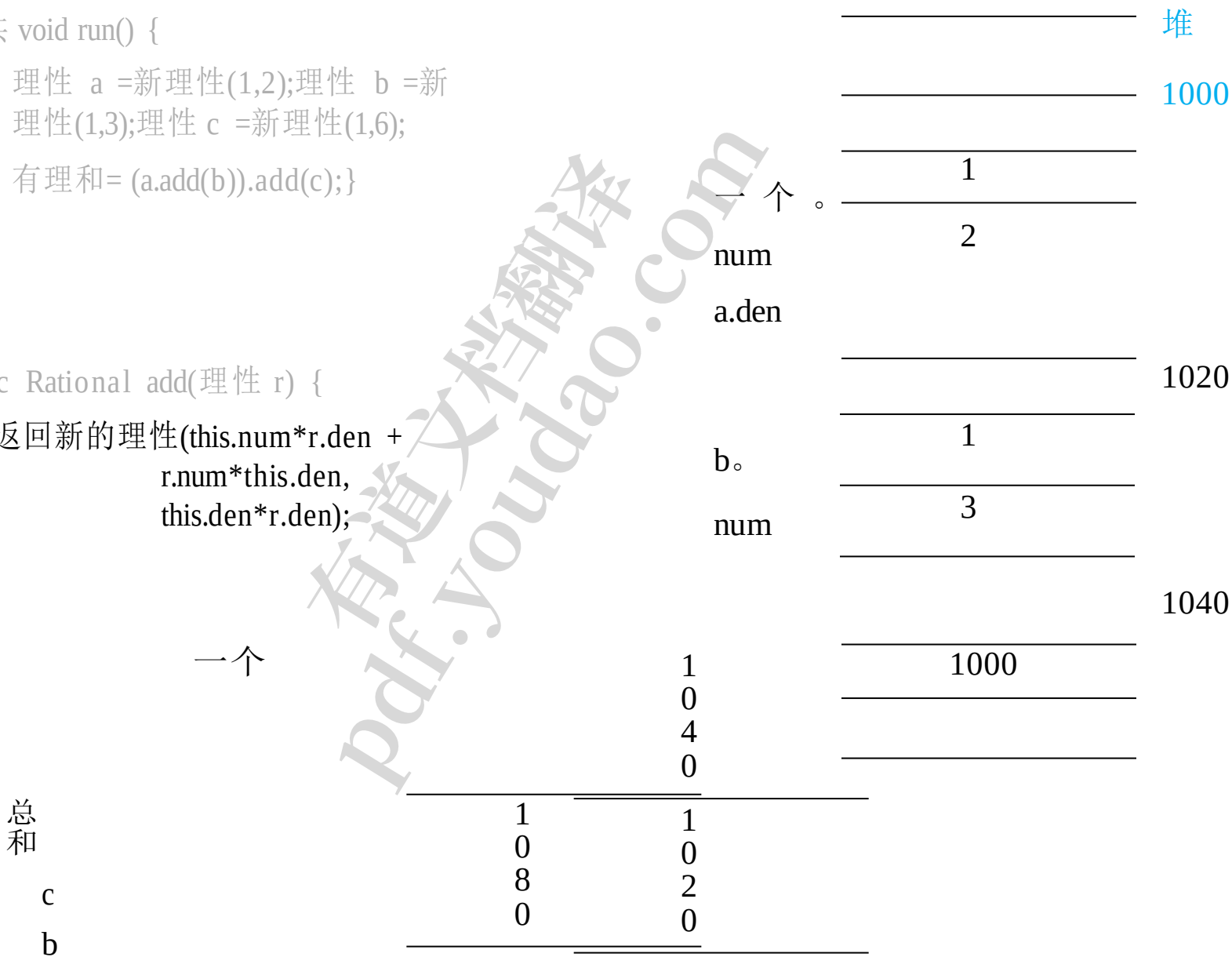
理性 a =新理性(1,2);理性 b =新理性(1,3);理性 c =新理性(1,6);

有理和= (a.add(b)).add(c);}

public Rational add(理性 r) {

返回新的理性(this.num*r.den +
r.num*this.den,
this.den*r.den);

}



	c。num	1
	c.den	6
FFB4		
FFB8	(a.add (b))。.den	5
FFBC	num (a.add (b))	6
FFC0		
堆栈	阀门(a.add (b)) (c) .num	1
	阀门(a.add (b)) (c) .den	1

1060

1080

堆

公共 void run() {

理性 a =新理性(1,2);理性 b =新理性(1,3);理性 c =新理性(1,6);

有理和= (a.add(b)).add(c);}

1000

一个。

1

2

num

a.den

public Rational add(理性 r) {

返回新的理性(this.num*r.den +
r.num*this.den,
this.den*r.den);

}

1020

b。

1

3

num

1040

一个

1000

1040

总和

c

b

1080

1020

c。num 1

c.den 6

FFB4

FFB8 (a.add (b))。.den 5

FFBC num (a.add (b)) 6

FFC0

堆栈 1

阀门(a.add (b)) (c)。阀门 1

num (a.add (b) (c) .den

1060

1080

垃圾收集

在例子中，对象 a.add(b)是在中间步骤创建的，但没有被最终堆栈引用。它现在是垃圾。

当内存不足时，Java 会进行垃圾收集

标记栈上或静态存储中变量引用的对象。

清除堆中的所有对象，回收未标记的对象(垃圾)。

这个过程称为垃圾回收。

练习:堆堆图

```
公共类点{
    公共类点(int x, int y) {cx =
        x;Cy = y;
    }

    Private int cx;
    私人 int cy;
}
```

```
公类行{
    公共类线(p1 点，p2 点){开始= p1;
        终点= p2;
    }

    私点开始;私点结束;
}
```

```
公共 void run() {
    点 p1 =新点(0,0);p2 点=新点
    (200,200);
    点线=新点(p1, p2);}
```

绘制一个堆栈图(指针模型)，显示 run()方法返回之前的内存状态。

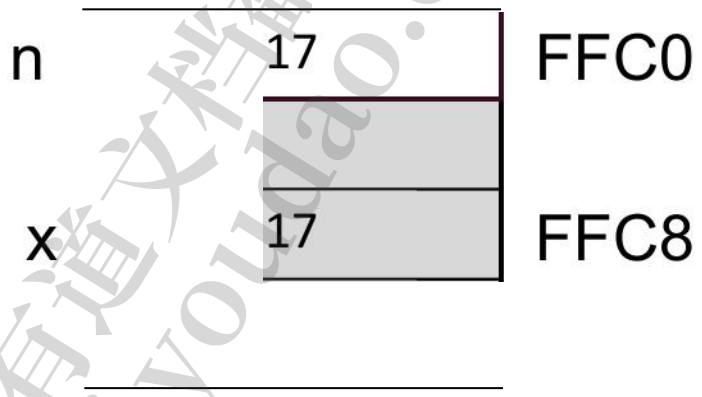
基本类型与对象

原始类型

```
公共 void  run()  {int  x =
      17;增量
      (x);Println(“x  =”  +
      x);
}

私有 void increment(int  n) {n++;
    Println(“n  =”  + n);}

输出  n
= 18 n =
17
```

当您将原始类型的参数传递给方法时，Java 将参数的值复制到参数变量中。因此，对形参变量的修改对实参没有影响。

传递基本类型 int 的 x，值递增(x);



X(一个值)被复制到 n 中

EMBEDDEDINTEGER 类

```
公共类 EnbeddedInteger{公共
    EmbeddedInteger(int  n) {
        值= n;}
    公共 void  setValue(int  n) {
        值= n;}
    public  int getValue() {
        返回值;}
    公共字符串 toString() {
        返回" " +值;}
    私有 int 值;
}
```

对象

```
公共 void run() {
    EmbeddedInteger  x = new  EmbeddedInteger(17);增
    量(x);
    Println(“x  =” + x);
}

私有无效增量(EmbeddedInteger  n) {n. setvalue
   (n. getvalue () + 1);Println(“n  =” + n);
}
```

输出 n
= 18 n
= 18

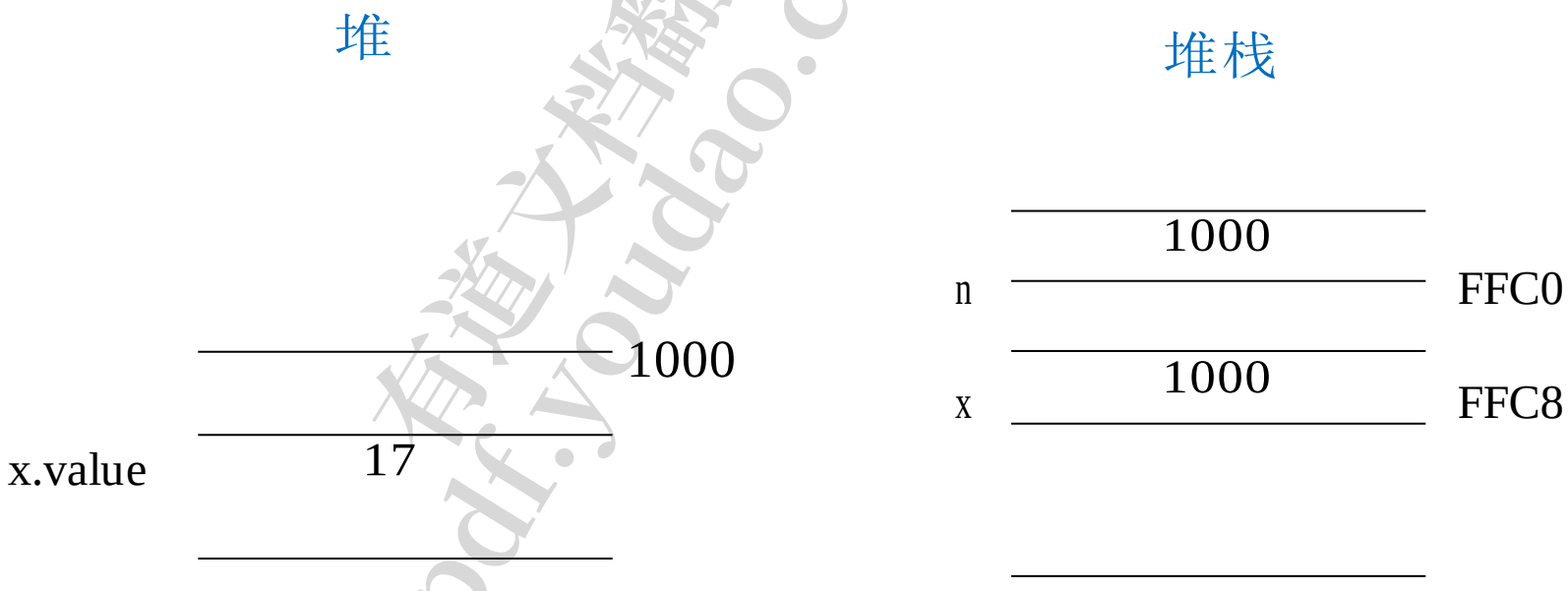
原始的类型和对象

当你传递一个对象作为参数时，似乎有某种形式的共享正在进行。然而，你对对象内部的实例变量所做的任何更改都会对该对象产生永久的影响。

堆-堆图清楚地说明了这种看似不对称的原因。当您将对象传递给方法时，Java 复制的是引用，而不是对象本身。

传递对象 x，是一个引用(地址)

增量(x)



X(一个对象的引用)被复制到 n X 中，n 共享同一个对象

大纲

计算机内存基础　数组

实现　示例

数组

Array:一个的值+ 有序且固定长度

Homogeneous:数组中的每个值都具有相同的类型

数组中的单个值称为元素。

元素的数量称为数组的长度

每个元素由其在数组中的位置标识，称为索引。在 Java 中，索引数字以 0 开始。

插图来自维基百科

数组:由一组元素(值或变量)组成的数据结构　每个元素都由至少一个数组索引或键标识。

每个元素的内存位置可以从它的索引元组计算出来。　最简单的数据结构类型是线性数组，也叫一维数组。

示例:由 10 个 32 位(4 字节)整型变量组成的数组，索引从 0 到 1

9
日
,

可能存储为 10 个单词在内存地址 2000,2004,2008，…，　2036，(十六进制:0x7D0, 0x7D4, 0x7D8，…, 0 x7f4)

索引为 i 的元素地址为 2000 + (i × 4)。

数组声明

一个数组的特征是元素类型

长度

Type [] identifier =新类型[长度];

初始化中的默认值 数字 0 布尔

值 false 对象 null

对象的数组

数组的元素可以是任何 Java 类的对象。

例如:一个 5 的数组 student 类的实例

```
Student []
topStudents  =新生[5];
```

定义长度

使用命名常量来声明数组的长度。

```
private static final int n_裁判= 5;double[]分数=
新的 double[n_裁判];
```

或从用户处读取数组的长度。

选择元素

标识元素数组[index]

Index 可以是一个表达式

循环遍历数组元素 for （int i = 0;I < array.length;我+ +){

涉及第 i 个元素的操作;

}

stack & heap 中会发生什么?

```
Int [] numbers = new Int [10];
```

人类可读的索引值

从 0 开始的索引编号可能会令人困惑。有时，使用以 1 开始的索引是有意义的。

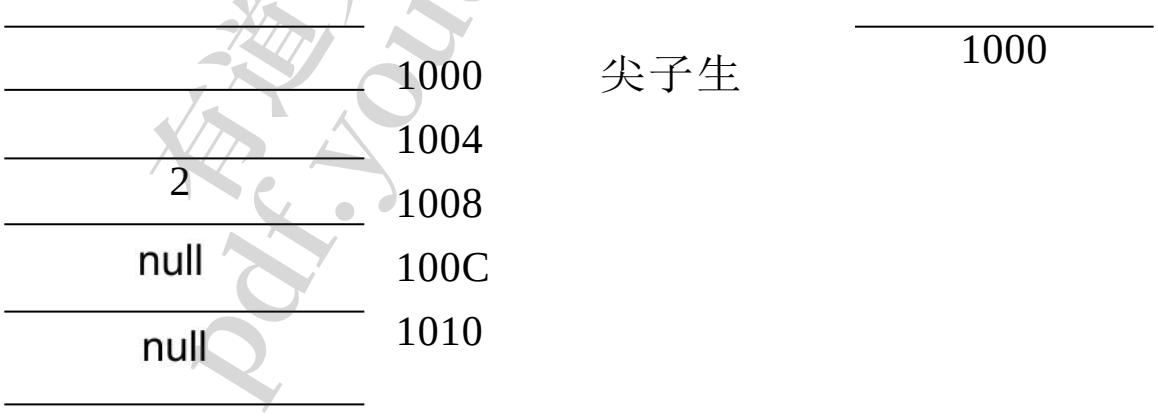
两种标准方式:

在内部使用 Java 的索引号，然后在呈现给用户时添加一个索引号。使用从 1 开始的索引值，忽略每个数组中的第一个(0)元素。

```
/**
 学生班是基础班。
 */
公共班学生
{
    /**
     @param name学生的名字
     学生的id
     */

    public Student(String name, int id)
    {studentName = name;studententid = id;

    }
    /**
     获取学生的姓名
     @返回学生的名字
     */
    getName                    ) {
        返回studentName;
    }
    /**
     获取学生id
     @return学生的id
     */
    public int getId() {
        返回studentId;
    }
    /**
     设置获得的学分数量。
     @param credits获得的新学分数
     */
    公共无效set学分(双学分){
        creditsEarned = credits;
    }
    /**
     获取获得的学分数。
     该学生已获得的学分数
     */
    公共双getCredits() {
        return creditsEarned;
    }
    /**
     设置学生是否被支付。
     @param flag true或false的值，表示已缴费状态
     */
    public void setPaidUp(boolean
     flag) {paidUp = flag;
    }
    /**
     返回该学生是否已付清学费。
     @return该学生是否已缴学费。
     */
    public boolean isPaidUp()
    {return paidUp;
    }
    /**
     创建一个标识该学生的字符串。@return用于
     显示该学生的字符串。
     */
    toString()
        返回studentName    " (#" + studentId+   ")";
    }

    /*公共常量*/公共静态final double
    CREDITS_TO_GRADUATE = 32.0;
    /*私有实例变量*/.
                    private String
                    studentName;private
    private int studententid;
    私人双学分searned;                    /*学生姓名*/.
                                          /*学生id */.
                                          /*已修学分*/ /*学生是否已付清
                                          学费*/
}
```

数组的内部表示

学生[]尖子生=新生[2];尖子生[0]=新生("Abcd"，314159);



堆栈

堆

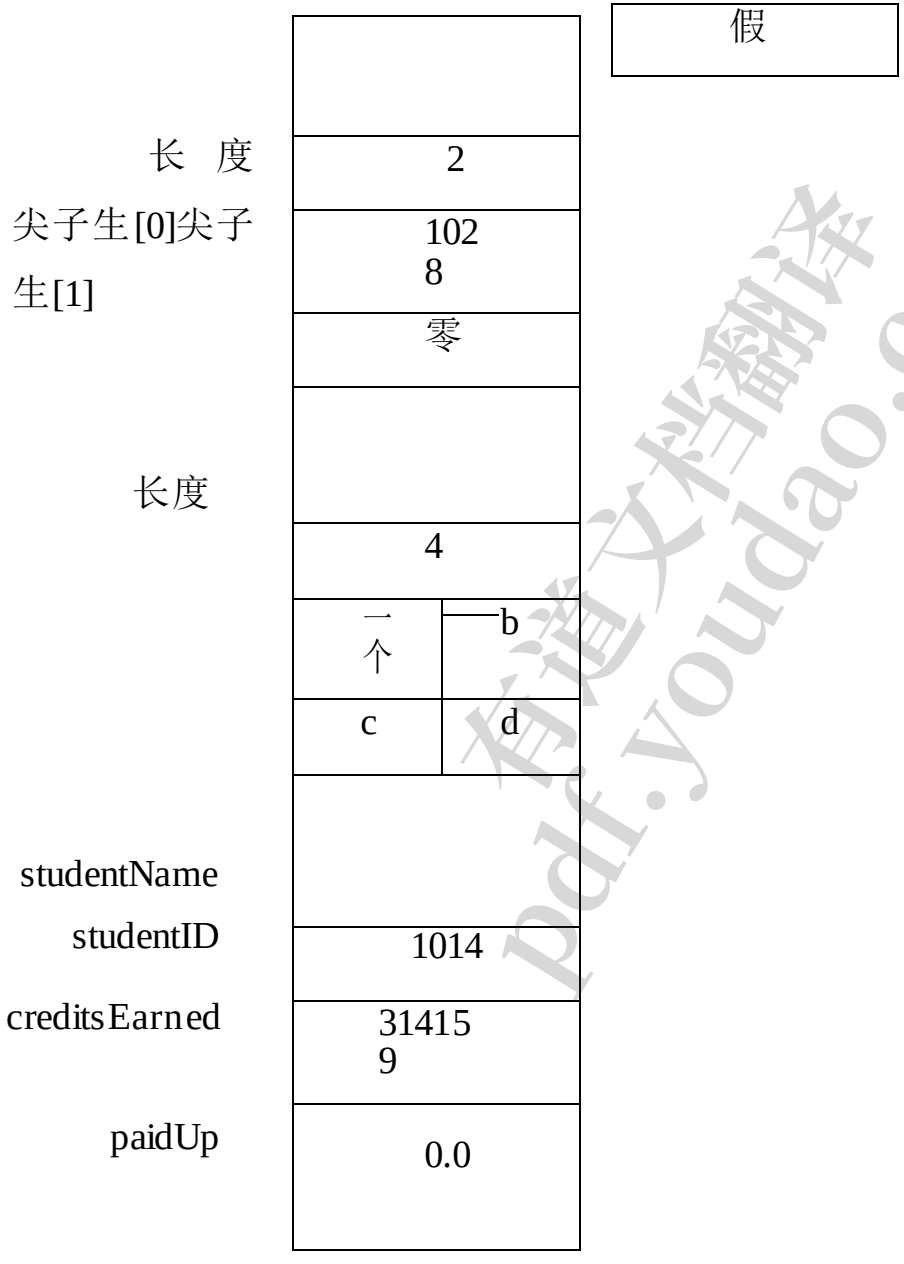
FFB8

FFBC

FFC0

长 度

尖子生[0]尖子生[1]



1
0
0
0

1
0
0
4

1008
Stude
nt[]尖
子生=
新生
[2];

100C 尖子生[0]=
新生(“Abcd”，
314159);1010

1
0
1
4

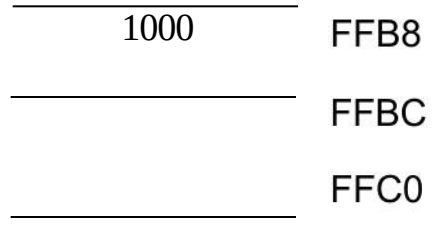
1
0
1
8

1
0
1

c
1020
1024
1028
102 c
1030

1034
1038
103 c
1040

尖子生



将数组作为参数传递

回忆一下:传递对象(引用)与传递基本类型(值)作为参数。

Java 将所有数组定义为对象，这意味着数组的元素在被调用方和调用方之间共享。

swapElements(数组[i]，数组[n - i - 1])(错误)

swapElements(array, i, n - i - 1)

```
private void swapElements(int[] array, int p1, int p2) {int
        tmp = array[p1];
        Array [p1] = Array
        [p2];
        Array [p2] = tmp;
}
```

Java 中的每个数组都有一个长度字段

```
私有 void reverseArray(int[]数组){
        For (int I = 0;I <数组。长度/ 2;我+ +){
                swapElements(array, i, array。Length - I - 1);}
}
```

使用数组

示例:字母频率表

数组:letterCounts[]索引:
距离' A '

index = Character.toUpperCase(ch) - ' A '

letterCounts[0]是' A '或' A '的计数

一种方便的初始化数组的方法:

```
Int [] digits  = {0,1,2,3,4,5,6,7,8,9};


private static final String[] US_CITIES_OVER_ONE_MILLION ={"纽约",
      “洛杉矶”,
      “芝加哥”,
      “休斯
     敦”,
      “费城”,
      “凤凰城”,
      “圣地亚哥”,
      “圣安东尼奥”,
      “达拉
     斯”,
}
```

二维数组

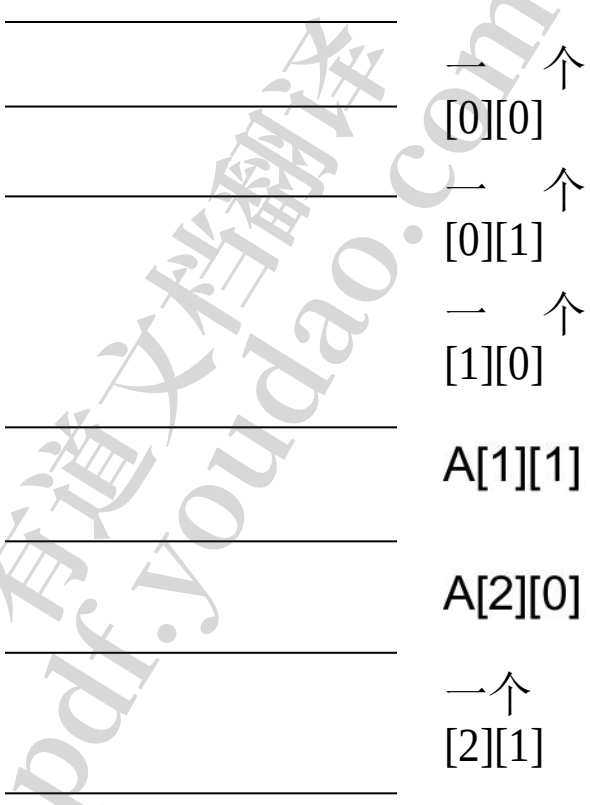
数组的每个元素都是一个数组(维度相同)int[][] A = new int[3][2];

由三个二维数组组成的数组 A[0][0] A[0][1]

一个[1][0]的[1][1]

一个[2][0]的[2][0]

内存分配(行定向)



初始化二维数组

```
static int A[3][2]  = {
    {1,  4},
    {2,  5},
    {3,  6}
};
```

A 3-by-2 matrix

arraylist 类

java。**util** 包包含一个名为数组列表的类

提供了标准的数组行为以及其他有用的操作。

数组列表是一个 Java 类，而不是语言中的特殊形式。arraylist 上的所有操作都使用方法调用来指示。

通过调用数组列表构造函数来创建一个新的数组列表。 通过调用 size 方法获取元素的数量。

使用 get 和 set 方法来选择单个元素。

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

大纲

计算机内存基础　数组

实现　示例

实现

编译并运行以下程序　项目 2.1

maxArray.java　项目 2.2

remaxArray.java　项目 2.3

sortArray.java　项目 2.4

binArray.java　项目 2.5

mergeArrays.java　项目 2.6　noDups.java

大纲

计算机内存基础　数组

实现　示例

数字数组

数组中的每个元素都是数字:int、float、double、…

```
公共类

    public static void main(String[] args)


        //创建一个数组
        Int [] age = {12,4,5,2,5};

        // 访问每个数组元素
        system . out。println(“访问数组元素:”);
        system . out。println(“First Element:” + age[0]);
        system . out。println(“Second Element:” + age[1]);
        system . out。println(“Third Element:” + age[2]);
        system . out。println(“Fourth Element:” + age[3]);system . out。println(
        “Fifth Element:” + age[4]);
    }
}
```

输出

```
访问数组元素:
第一个元素:12
第二元素:4
第三元素:5第四元素:2
第五元素:5
```

```
公共类

    public static void main(String[] args)


        // 创建一个数组
        Int [] age = {12,4,5};

        //
        // 使用for循环遍历数组
        //
        system . out。println( “Using for Loop:” );
        for(int i = 0; i < age.length; i++)
        {
            System.out.println(年龄[我]);
        }
    }
}
```

输出

```
使用for循环:
12
4
5
```

公共类

```
public static void main(String[] args)
{
    //创建数组
    Int [] age = {12,4,5};

    //
    //  使用for循环遍历数组
    system . out。println(“Using for-each Loop:”
    );For (int a: age)
    {
        System.out.println(一个);
    }
}
}
```

输出

```
使用for-each循环:
12
4
5
```

```
公共类
{
    public static void main(String[] args)
    .


        int[]数字={2 9 0、5、12,-25,22日,9日,8日12};Int sum = θ;双平均;


        //访问每个循环使用的所有元素//在sum中添加每个元素
        For (int number: numbers)

            Sum += number;
        }

        //获取元素总数
        int arrayLength = numbers.length;

        //计算平均值
        //将int的平均值转换为double
        average = ((double)sum / (double)arrayLength);

        system . out。println("Sum = " + Sum);system . out。
        println("Average = " + Average);
    }
}
```

输出:

```
Sum = 36
平均= 3.6
```

```
公共类MultidimensionalArray
{
    public static void main(String[] args)
    {

        //创建一个2d数组
        Int [][] a =
        {
            {1, 2, 3},
            {4, 5, 6, 9},{7},
        };

        //计算每一行的长度

        system . out。println(“第一行的长度:”+一个[0].length);system . out。println(“第2行长度:”+一个[1].length);system . out。println(“第3行长度:”+一个[2].length);

    }
}
```

输出:

```
第一行长度:3第2行长度
:4第3行长度:1
```

```
公共类MultidimensionalArray f
    public static void main(String[] args)


        int [] []       =
        {
            {1, -2, 3},
            {-4, -5, 6, 9},{7},
        };

          For (int I = 0;I < a.length;+ + i)

            For (int j = 0;J < a[i].length;+ + j)
            {
                System.out.println([我][5]);
            }
        }
    ~
}
```

输出:

```
 1
 -2
 3
 -4
 -5
9
 9
 7
```

```
公共类MultidimensionalArray
{
        public static void main(String[] args)
    {

        //创建一个2d数组
        int [] []       =
        {
            {1, -2, 3},

            {-4, -5, 6, 9},{7},
        };

        // 第一次……每个循环访问2d数组内的单个数组
        //
        for (int[] innerArray: a)
        {
            //秒为…每个循环访问行内的每个元素
            for(int data: innerArray)
            {
                System.out.println(数据);
            }
        }
    }
}
```

输出:

```
1
 -2
3
 -4
 -5
 6
 9
 7
```

```
公共类ThreeArray
{
    public static void main(String[] args)
    {
        // 创建一个3d数组
        int[][][]测试 =
        {
                {1, -2, 3},{2, 3, 4}
            },
            {-4, -5, 6, 9}, {1} {2, 3}。
            }
        };
        //每个循环遍历3d数组的元素
        for (int[][] array2D: test)
            for (int[] array1D: array2D)
                for(int对象:array1D)
                {
                    System.out.println(项);
                }
            }
        }
    }
}
```

输出:

```
1
2
3
-4
-5
6
9
1
2
3
```

```
公共类ConvertDoubleArrayToFloatArray

    public static void main(字符串args[])
    {
        double[]doubleArray= {3.5,5.0,7.5,11.55};
        float[]floatArray=新的浮动[doubleArray.length];
        For (int I = 0;i <doubleArray.length;我+ +)
        {
            floatArray[i] = (float)doubleArray[i];
        }
        for(int i = 0; i < floatArray.length; i++)
        {
            system . out。println("Element at Index " + i + " is: " + floatArray[i]);
        }
    }
}
```

输出是什么?

```
公共类ConvertFloatArrayToDoubleArray
{
    public static void main(字符串args[])
    {
        float[]floatArray= {2.0f, 1.5f, 8.45f,
        116.77f};double[]doubleArray= new
        double[floatArray.length];For (int I = 0;i <floatArray.length;我+ +)
        {
            doubleArray[i] =(双)floatArray[i];
        }
        for(int i = 0; i < doubleArray.length; i++)
        {
            system . out。println("Element at Index " + i + " is: " + doubleArray[i]);
        }
    }
}
```

输出是什么?

字符串数组

字符串数组是一个保存固定数量的字符串或字符串值的数组。Java 中常用的一种结构。

甚至在 Java 中，“main”函数的参数也是一个字符串数组。

字符串数组是一个对象数组。 字符串是一个对象。

```
公共类

    public static void main(String[] args)
    {

        String [] myarray;//声明无大小的字符串数组string [] strArray = new
        string [5];//带有大小的声明

        / / System.out.println (myarray [0]);//变量myarray可能没有被初始化
        //显示第二个数组的元素
        System.out.print(strArray[0] + " " + strArray[1] + " " + strArray[2] + " " +
                         strArray[3]  + " "  + strArray[4]);
    }
}
```

```
零，零，零，零
```

2 1公共类主要

```
{
    public static void main(String[] args)
    {
        //声明并初始化一个字符串数组
        String[]numArray={"一"，"二"，"三"，"四"，"五"};
        int len = numArray.length;//获取数组的长度
        //显示数组长度
        system . out。println("numArray长度{\"one\"，\"two\"，\"three\"，\"four\"，\"five\"}:" + len);
    }
}
```

```
numArray的长度{"one"，"two"，"three"，"four"，"five"}:5
```

```
公共类
{
    public static void main(String[] args)
    {
        //声明并初始化一个字符串数组
        String [] numArray ={“1”，“2”，“3”，“4”、 “5”};system . out。println(“使用
        for循环显示的字符串数组元素:”);
        // for循环遍历字符串数组
        for(int i = 0; i < numArray.length; i++)
            System.out.print(numArray[i] + " ");

        System.out.println(“\ n”);

        system . out。println(“使用增强的for循环显示的字符串数组元素:”);
        //增强的for循环遍历字符串数组
        for(String val: numArray)
            System.out.print(val + " ");
    }
}
```

```
使用for循环显示的字符串数组元素:

一二三四五


使用增强的for循环显示的字符串数组元素:

一二三四五
```

```
进口java.util。*;
公共类Main
{
    public static void main(String[] args)
    {
        字符串[]colorarray=新字符串[5];
        //数组初始值
        colorarray[θ] = "Red";
        colorsArray[1] = "Green";
        colorsArray[2] = "Blue";
        System.out.println("Original Array:" + Arrays.toString(colorsArray));
        intnumberOfItems = 3;

        //尝试在数组末尾添加新值
        字符串newItem= "黄色";
        colorsArray[numberOfItems++] =newItem;
        system . out。println(“添加一个元素后的数组:”      +
                              Arrays.toString (colorsArray));
    }
}
```

原始数组:[红色，绿色，蓝色，null, null]

添加一个元素后的数组:[Red, Green, Blue, Yellow, null]

```
进口java.util。*;

公共类
{
    public static void main(String[] args)
    {
        //原始数组
        String[]colorarray= {"Red"， "Green"， "Blue"};
        system . out。println("Original Array: " + Arrays.toString(colorarray));√

        //原始数组的长度
        int orig_length = colorsArray.length;
        //要添加到字符串数组中的新元素
        String newElement = "Orange";
        //定义一个长度大于原数组的新数组
        String[] newArray = new String[orig_length + 1];
        //添加原数组的所有元素到新数组
        For (int I = 0;i < colorsArray.length;我+ +)
        。
            newArray[i] = colorsArray [i];
        }
        //添加新元素到新数组的末尾
        纽瓦雷[纽瓦雷]。length - 1] = newElement;
        //创建新数组作为原始数组并打印
        colorsArray= newArray;
        system . out。println("添加新项目后的数组:" + Arrays.toString(colorsArray));
    }
}
```

```
原始阵列:[红，绿，蓝]

添加新物品后的数组:[红，绿，蓝，橙]
```

```
进口java.util。*;

类主要
{

    public static void main(String[] args)
    {
        String[]颜色={“红色”、“绿色”、“蓝”、“白”、“橙色”};system . out
        。println("Original array: " +Arrays.toString(colors));Arrays.sort(颜色);
        system . out。println("Sorted array: " +Arrays.toString(colors));
    }
}
```

```
原始阵列:[红、绿、蓝、白、橙]

排序数组:[蓝、绿、橙、红、白]
```

```
进口java.util。*;
公共类
{
        public static void main(String[] args)

            String[] strArray = {"Book"， "Pencil"， "橡皮"，"Color"， "Pen"};Boolean found = false;
            Int index = 0;
            String searchStr = "钢笔";
            For (int I = 0;i < strArray.length;我+ +)

                    如果(searchStr.equals (strArray[我]))

                        Index = i;
                        Found = true;
                        打破;
                    }
            }
            如果(发现)
                        system . out。println(searchStr +“在索引处找到”+ index);
            其他的
                    system . out。println(searchStr +“在数组中未找到”);

    }
}
```

在索引处找到钢笔

```
进口java.util。*;公共
类Main
{
    public static void main(字符串args[])
    {
        / /字符串数组
        String [] str_Array ={“这”，“是”，“软件”、“测试”，“帮助”};system .
        out。print(“原始字符串数组:”);
        //打印字符串数组
        (String val: str_Array)
            System.out.print(val + " ");

        System.out.println(“\ n”);

        //从给定的字符串数组stringbuilder中构造一个stringbuilder对象
        ;For (int I = 0;i < str_Array.length;我+ +)
        {
            stringBuilder.append(str_Array[i] + " ");
        }
        //打印字符串
        system . out。println("从字符串数组获得的字符串:" + stringBuilder.toString());
    }
}
```

```
原始字符串数组:这是软件测试帮助

这是软件测试帮助
```

```
进口java.util。*;

公共类

    public static void main(String[] args)

        //字符串数组声明
        String [] str_Array ={“10”、“20”、“30”，“40”、“50”};
        //打印字符串数组
        system . out。println(“原始字符串数组:”);for(String
        val: str_Array)
            System.out.print(val + " ");

        system . out。println("\n从字符串数组获得的整数数组:");
        //声明一个int数组
        int [] int_Array = new int [str_Array.length];
        //赋string数组值给int数组
        for(int i = 0; i < str_Array.length; i++)
        {
            int_Array[i] = Integer.parseInt(str_Array[i]);
        }
        //显示int数组
        System.out.println (Arrays.toString (int_Array));
    }
}
```

```
原始字符串数组:

10 20 30 40 50

从string array获取的整数数组:

[10, 20, 30, 40, 50]
```

```
公共类ByteArraySize
{
    public static void main(字符串args[])
    {
        字符串str = "字节数组大小示例";
        byte array[] = str.getBytes();
        system . out。println(“字节数组的大小:”+ Array .length);
    }
}
```


输出是什么?

```
进口java.util.Arrays;
公共类StringToByteArray
{
    public static void main(String[] args)
    {
        String str = " convert String to byte Array in Java ";
        byte[] bytearray = str.getBytes();
        System.out.println(Arrays.toString(bytearray));
    }
}
```

输出是什么?

布尔值数组

- 数组中的每个元素都是一个布尔值(true, false)。

进口 java.util.Arrays;公开类
BooleanArrayTest

公共静态 void main(String[] args)

BooLean[]boolArray new 布尔值[5];iboolArray.length;我+ +

System.out.println(boolArray[i]);

16
11
12 数组。 fil1(boolArray，
Boolean.FALSE);/所有值将为 false
13 for(int I =e;i boolArray.length;我+ +
14
15 boolArray[ij];
16
17 数组。 fil1(boolArray，布尔值;真正
18 的);所有的值都将为 true
19 For (int i=e;iboolArray.length;我+ +
28 System.out.println(boolArray[i]);
21
22
23
24

输出是什么?

总结

懂记忆，懂一切!

数组的行为体现在栈和队列的变化上。

对象数组和基本类型数组有什么区别?