Array

10/26/2015

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Primitive Array (1/2)

```
1  public class ArrayDemo{
2    public static void main(String[] arg){
3      int[] arr = new int[3];
4      //think: intArray arr = new intArray(3);
5      arr[0] = 1; //think: arr.setElement(0, 1);
6      arr[1] = 3;
7      arr[2] = 5;
8      arr[3] = 9;
9      System.out.println(arr.length);
10     arr.length = 5;
11     arr = null;
12   }
13 }
```

- array is a reference by itself
- `new`, `null` like usual reference instances
- primitive array: `new` initialize element to default
- length: read-only
- index out of bound: run time error

Primitive Array (2/2)

```
1  public class ArrayDemo{
2    public static void main(String[] arg){
3      int[] arr = {1, 3, 5};
4      //compare String s = "HTLin";
5      System.out.println(arr.length);
6    }
7  }
```

- construct an array instance (with automatic length calculation), then assign its address to the reference variable

Primitive Array: Key Point

primitive array: reference to “a batch of values”

Reference Array (1/1)

```
1  class Record{ String name; int score; }
2  public class ArrayDemo{
3    public static void main(String[] arg){
4      Record[] arr = new Record[3];
5      System.out.println(arr[0]);
6      arr[0] = new Record();
7      arr[1] = new Record();
8      arr[2] = arr[0];
9      arr[1] = null;
10     arr = null;
11   }
12 }
```

- array is a reference
- reference array: `new` initialize element to null

Reference Array: Key Point

reference array: reference to “a batch of references”

Multidimensional Array (1/3)

```
1  public class ArrayDemo{
2    public static void main(String[] arg){
3      int[][] arr = new int[3][];
4      //think: intArray[] arr = new intArray[3];
5      arr[0] = new int[5];//think arr[0] = new intArray(5);
6      arr[1] = arr[0];
7      arr[2] = null;
8      System.out.println(arr.length);
9      System.out.println(arr[1].length);
10   }
11 }
```

- multidimensional: array of “array references”
- can be irregular

Multidimensional Array (2/3)

```
1  public class ArrayDemo{
2    public static void main(String[] arg){
3      int[][] arr = new int[3][5];
4      System.out.println(arr.length);
5      System.out.println(arr[1].length);
6    }
7  }
```

- still array of “array references”
- regular, automatic construction

Multidimensional Array (3/3)

```
1  public class ArrayDemo{
2    public static void main(String[] arg){
3      int[][] arr = {null, {0, 1}, {2, 3, 4}};
4      System.out.println(arr.length);
5      System.out.println(arr[1].length);
6    }
7  }
```

- construct an array, and assign its address to reference

Multidimensional Array: Key Point

multidimensional array: a special reference array, reference to “a batch of (multidimensional) arrays”

Array Argument/Parameter (1/1)

```
1  class Tool{
2    void swap(int[] both){
3      int tmp = both[0];
4      both[0] = both[1];
5      both[1] = tmp;
6    }
7  }
8  public class Demo{
9    public static void main(String[] arg){
10     Tool t = new Tool();
11     int[] arr = new int[2];
12     arr[0] = 3; arr[1] = 5;
13     t.swap(arr);
14     System.out.println(arr[0]);
15     System.out.println(arr[1]);
16   }
17 }
```

- array is just special reference, same calling mechanism

Array Argument/Parameter: Key Point

array is like other extended types in return value, parameter passing, assignment (=), reference equal (==)

For Each (1/1)

```
1  class Util{
2    public static double avg(double[] arr){
3      double res = 0.0;
4      int i;
5      for(i=0;i<arr.length;i++) res += arr[i];
6      return res / arr.length;
7    }
8    public static double cool_avg(double[] arr){
9      double res = 0.0;
10     for(double element: arr) res += element;
11     return res / arr.length;
12   }
13 }
```

- special for (called **for each**) that automatically enumerates all the elements within a collection

For Each: Key Point

for each: an elegant tool to be kept in your tool-box

Variable-Length Argument List (1/1)

```
1  class Util{
2    public static double cool_avg(double[] arr){
3      double res = 0.0;
4      for(double element: arr) res += element;
5      return res / arr.length;
6    }
7    public static double even_cooler_avg(double... arr){
8      double res = 0.0;
9      for(double element: arr) res += element;
10     return res / arr.length;
11   }
12 }
13
14 System.out.println(Util.cool_avg(new double[] {1, 5, 3, 2}));
15 System.out.println(Util.cool_avg(new double[] {1, 2, 3}));
16 System.out.println(Util.even_cooler_avg(1, 5, 3, 2));
17 System.out.println(Util.even_cooler_avg(1, 2, 3));
18 double[] a = {1, 2, 4};
19 System.out.println(Util.cool_avg(a));
20 System.out.println(Util.even_cooler_avg(a));
```

- a “syntactic sugar” after Java 5

Variable-Length Argument List: Key Point

variable-length arguments: another good tool that roots from arrays