Arrays (Chapter 6)

Hsuan-Tien Lin

Department of CSIE, NTU

OOP Class, March 18, 2013

Primitive Array (1/2)

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

base type

ohohoh

hahaha

arr.setElement(0, 1);

int[] arr;

- 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)

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

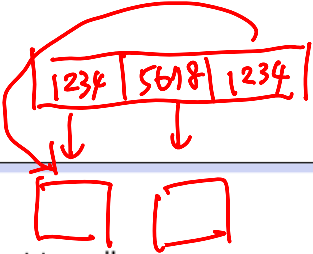
- construct an array instance (with automatic length calculation), then assign its address to the reference variable

Primitive Array: Key Point

primitive array: reference to “a batch of values”

Reference Array (1/1)

```java
class Record{ String name; int score; }
public class ArrayDemo{
  public static void main(String[] arg){
    Record[] arr = new Record[3];
    System.out.println(arr[0]);   null
    arr[0] = new Record();
    arr[1] = new Record();
    arr[2] = arr[0];
    arr[1] = null;
    arr = null;
  }
}
```



- 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)

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

int[] ←

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

Multidimensional Array (2/3)

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

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

Multidimensional Array (3/3)

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

- 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)

```
class Tool{
  void swap(int[] both){
    int tmp = both[0];
    both[0] = both[1];
    both[1] = tmp;
  }
}
public class Demo{
  public static void main(String[] arg){
    Tool t = new Tool();
    int[] arr = new int[2];
    arr[0] = 3; arr[1] = 5;
    t.swap(arr);
    System.out.println(arr[0]);
    System.out.println(arr[1]);
  }
}
```

- 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)

```java
class Util{
  public static double avg(double[] arr){
    double res = 0.0;
    int i;
    for(i=0;i<arr.length;i++) res += arr[i];
    return res / arr.length;
  }
  public static double cool_avg(double[] arr){
    double res = 0.0;
    for(double element: arr) res += element;
    return res / arr.length;
  }
}
```

- 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 toolbox

Variable-Length Argument List (1/1)

```java
class Util{
  public static double cool_avg(double[] arr){
    double res = 0.0;
    for(double element: arr) res += element;
    return res / arr.length;
  }
  public static double even_cooler_avg(double... arr){
    double res = 0.0;
    for(double element: arr) res += element;
    return res / arr.length;
  }
}

System.out.println(Util.cool_avg(new double[] {1, 5, 3, 2}));
System.out.println(Util.cool_avg(new double[] {1, 2, 3}));
System.out.println(Util.even_cooler_avg(1, 5, 3, 2));
System.out.println(Util.even_cooler_avg(1, 2, 3));
double[] a = {1, 2, 4};
System.out.println(Util.cool_avg(a));
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

A method or constructor may take at most one variable length argument, and this must always be the last declared argument in the signature.