Java Programming

U Hou Lok

Department of Computer Science and Information Engineering,
National Taiwan University

Java 273
22 Aug.–2 Sep., 2016

```
class Lecture5 {

                    "Arrays"

}
```

Arrays

An array stores a large collection of data which is of the same type.

```java
...
        // assume the size variable exists above
        T[] x = new T[size];
        // this creates an array of T type referenced by x
...
```

- **T** can be any data type.
- This statement comprises two parts:
  - Variable declaration
  - Creating an array

Variable Declaration

- In the left-hand side of the assignment operator, it is a declaration for an array variable, which does not allocate real space for the array.
- In reality, this variable occupies only a certain space for the reference to an array.[1]
- If a reference variable does not refer to an array, the value of the variable is null.[2]
- So you cannot assign elements to this array variable unless it has already been created.

[1] Recall the stack and the heap in the memory layout.

[2] Moreover, this holds for any reference variable. For example, the **Scanner** type.

Creating A Real Array

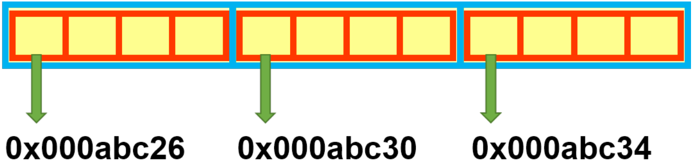
- All arrays of Java are objects.
- So the new operator returns the reference after creating an array object.
- The type of reference variables must be compatible to that of the array object.
- The variable *size* must be a positive integer for the number of elements.
- Note that the size of an array cannot be changed after the array is created.[3]

[3] You can try the **ArrayList** class. See any textbook for data structures.

Arrays and Memory

- Consider this array whose elements are all int type:

```
int[] x = new int[3];
```



- The array is allocated contiguously in the memory.
- Note that the arrays (in Java) are zero-based indexing. (Why?)
- So we have x[0], x[1], and x[2].

Array Initializer

Arrays can be initialized when they are declared.

- When an array is created, the elements are assigned the default value:
  - 0 for the numeric primitive data types
  - \u0000 for char type
  - false for boolean type
- An array also can be initialized by enumerating all the elements in a list.
- For example,

```java
int[] x = {1, 2, 3};
```

- Note that there is no need to use new if enumeration is used.

Processing Arrays

When processing array elements, we often use a for loop.

- Since the size of the array is known, it is natural to use a for loop to manipulate the array.
- For all arrays, they have a field called **length** which records the size of this array.
    - For example, use x.length to get the size of x.

Examples

Initializing arrays with input values

```
...
        // let x be an integer array with a certain size
        for (int i = 0; i < x.length; ++i) {
            x[i] = input.nextInt();
        }
...
```

Initializing arrays with random values

```
...
        for (int i = 0; i < x.length; ++i) {
            x[i] = (int) (Math.random() * 10);
        }
...
```

Displaying arrays

```
...
        for (int i = 0; i < x.length; ++i) {
            System.out.printf("x[%d] = %d\n", i, x[i]);
        }
...
```

Summing all elements

```
...
        int sum = 0;
        for (int i = 0; i < x.length; ++i) {
            sum += x[i];
        }
...
```

Finding the extreme values

```
...
        int max = x[0];
        int min = x[0];
        for (int i = 1; i < x.length; ++i) {
            if (max < x[i]) max = x[i];
            if (min > x[i]) min = x[i];
        }
...
```

- How about the location of the extreme values?

Shuffling

```java
...
        // Assume that x is an integer array.
        for (int i = 0; i < x.length; ++i) {
            int j = (int) (Math.random() * x.length);
            // swap
            int tmp = x[i];
            x[i] = x[j];
            x[j] = tmp;
        }
...
```

- How to swap values of two variables without tmp?
- However, this simple (or naive) algorithm is biased.[4]

[4] See `https://blog.codinghorror.com/the-danger-of-naivete/`.

Exercise[5]

Generate 4 distinct integers randomly ranging from 0 to 9.

- Consider this algorithm.
  - Generate the 1st random integer and the 2nd random integer.
  - Then check if the second one is identical to the first integer.
  - If so, redraw and check again.
  - If not, then generate the next random integer and repeat the previous steps until the 4 distinct integers are set.
- Argue the time complexity for this algorithm.

[5] Thanks to a lively discussion on January 24, 2016.