



Lecture 3: Java Language Foundations II

Instructor: Xin Wang



CONTENTS OF THIS LESSON

- Booleans
 - If...Else
 - Switch
 - While Loop
 - For loop
 - Break/Continue
 - Arrays
-



Java Booleans



Java has a **boolean** data type, which can take the values **true** or **false**.

When we define the **boolean** variables, their values only can be **true** or **false**.

```
boolean isJavaFun = true;
boolean isFishTasty = false;
System.out.println(isJavaFun);    // Outputs true
System.out.println(isFishTasty);  // Outputs false
```

```
int x = 10;
int y = 9;
System.out.println(x > y); // returns true
System.out.println(x == y); // returns false
```



Java If...Else



Java conditional statements:

if – If a specified condition is true, a block of code will be executed.

else – If a specified condition is false, another block of code will be executed.

else if – If the first condition is false, a new condition will be specified and tested.

switch – Many alternative blocks of code can be specified to execute.

Syntax

```
if (condition1) { // block of code to be executed if condition1 is true
} else if (condition2) {
    // block of code to be executed if the condition1 is false and condition2 is true
} else {
    // block of code to be executed if the condition1 is false and condition2 is false
}
```

Q: What will be the result?

```
public class Main
{
    public static void main(String[] args) {
        int time = 11;
        if (time < 10) {
            time += 1;
            System.out.println(time);
        } else if (time < 20) {
            System.out.println("Good day.");
        } else {
            System.out.println("Good evening.");
        }
    }
}
```

```
public class Main
{
    public static void main(String[] args) {
        int time = 9;
        if (time < 10) {
            time += 1;
            System.out.println(time);
        } else if (time < 20) {
            System.out.println("Good day.");
        } else {
            System.out.println("Good evening.");
        }
    }
}
```



Java Switch



Switch Syntax

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

- **switch** statement is to select one of many code blocks to execute.
 - The value of the expression is compared with the value of each **case**.
 - **break** will stop the execution of the following code.
 - **default** specifies some code to run if there is no case match.
-

Q: What will be the result?

```
int day = 4;  
switch (day) {  
    case 6:  
        System.out.println("Today is Saturday");  
        break;  
    case 7:  
        System.out.println("Today is Sunday");  
        break;  
    default:  
        System.out.println("Looking forward to the  
Weekend");  
}
```



Java While Loop



While Loop – It will execute a block of code when a specified condition is **true**.

Syntax

```
while (condition) {  
    // code block to be executed  
}
```

```
int i = 0;  
while (i < 5) {  
    System.out.println(i);  
    i += 1;  
}
```

Do/while Loop – The loop will always be executed at least once, since the code block is executed before the condition is tested.

Syntax

```
do {  
    // code block to be executed  
}  
while (condition);
```

```
int i = 0;  
do {  
    System.out.println(i);  
    i += 1;  
}  
while (i < 5);
```



Java For Loop



For Loop – It can execute a block of code when certain condition is met.

Syntax

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

Statement 1 is executed before the execution of the code block. It can be a default value.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed after the code block has been executed.

Q: What will be the results?

```
for (int i = 0; i < 5; i++) {  
    System.out.println(i);  
}
```

```
for (int i = 0; i <= 10; i = i + 2) {  
    System.out.println(i);  
}
```



Java

Break/Continue



break statement can be used to jump out a **switch** statement and also a **loop** statement.

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        break;  
    }  
    System.out.println(i);  
}
```

continue statement can stop a iteration in the loop, if a specified condition is met, and continues with the next iteration in the loop.

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        continue;  
    }  
    System.out.println(i);  
}
```



Java Arrays



Java Arrays – Arrays can store multiple values in a single variable. To declare an array, the variable type can be defined with **square brackets**.

```
String[] cars;
```

- We can set value to this array using an array list with comma-separated values.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

- We can also create an array of integers.

```
int[] myNum = {10, 20, 30, 40};
```

- The array element can be accessed through the index number.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
System.out.println(cars[0]);
```

Note: Array indexes start with 0.

- The value of specific element can be changed through accessing the index number.

```
String[] cars = {"Volvo", "BMW", "Ford",  
"Mazda"};  
cars[0] = "Opel";  
System.out.println(cars[0]);
```

- Through array length, the total number of array elements can be counted.

```
String[] cars = {"Volvo", "BMW", "Ford",  
"Mazda"};  
System.out.println(cars.length);
```

Arrays Loop – The loop can run through the array elements with the **for** loop and use the **length** property to specify how many times the loop should be iterated.

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
    System.out.println(cars[i]);
}
```

For-each loop – It can loop through elements in an array.

Syntax

```
for (type variable : arrayname) { ...
}
```

For each String element (called **i** - as in **index**) in **cars**, print out the value of **i**

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
    System.out.println(i);
}
```

Access multidimensional arrays using for loop

We can define a multidimensional array.

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
```

We can access a certain element through index number.

```
int x = myNumbers[1][2];  
System.out.println(x);
```

Note: First is the row index and the second is the column index.

To get the elements of a multidimensional array, we can use for loop to access these two indexes.

```
public class Main {  
    public static void main(String[] args) {  
        int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };  
        for (int i = 0; i < myNumbers.length; ++i) {  
            for(int j = 0; j < myNumbers[i].length; ++j) {  
                System.out.println(myNumbers[i][j]);  
            }  
        }  
    }  
}
```



Thank you!
Any questions?

