

Core Java 8 and Development Tools

Lesson 03 : Language
Fundamentals





Lesson Objectives

After completing this lesson, participants will be able to:

- Understand Basic Java Language constructs like:
 - Keywords
 - Primitive Data Types
 - Operators
 - Variables
 - Literals
- Write Java programs using control structures
- Best Practices



3.1 : Keywords

Keywords in Java

abstract	continue	for	new	switch
assert ^{***}	default	<u>goto</u> [*]	package	synchronized
<u>boolean</u>	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	<u>enum</u> ^{****}	<u>instanceof</u>	return	transient
catch	extends	<u>int</u>	short	try
char	final	interface	static	void
class	finally	long	<u>strictfp</u> ^{**}	volatile
const [*]	float	native	super	while



3.2: Primitive Data types

Java Data types

Type	Size/Format	Description
byte	8-bit	Byte-length integer
short	16-bit	Short Integer
int	32-bit	Integer
long	64-bit	Long Integer
float	32-bit IEEE 754	Single precision floating point
double	64-bit IEE 754	Double precision floating point
char	16-bit	A single character
boolean	1-bit	True or False



3.3 : Operators and Assignments

Operators in Java

Operators can be divided into following groups:

- Arithmetic
- Bitwise
- Relational
- Logical
- *instanceof* Operator



3.3 : Operators and Assignments

Arithmetic Operators

Operator	Result
+	Addition
-	Subtraction (or unary) operator
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition assignment
-=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment
--	Decrement



3.3 : Operators and Assignments

Bitwise Operators

Apply upon *int*, *long*, *short*, *char* and *byte* data types:

Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
=	Bitwise OR assignment



Relational Operators

Determine the relationship that one operand has to another.

- Ordering and equality.

Operator	Result
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to



3.3 : Operators and Assignments

Logical Operators

Operator	Result
&&	Logical AND
	Logical OR
^	Logical XOR
!	Logical NOT
==	Equal to
?:	Ternary if-then-else



3.4: Variables and Literals

Variables

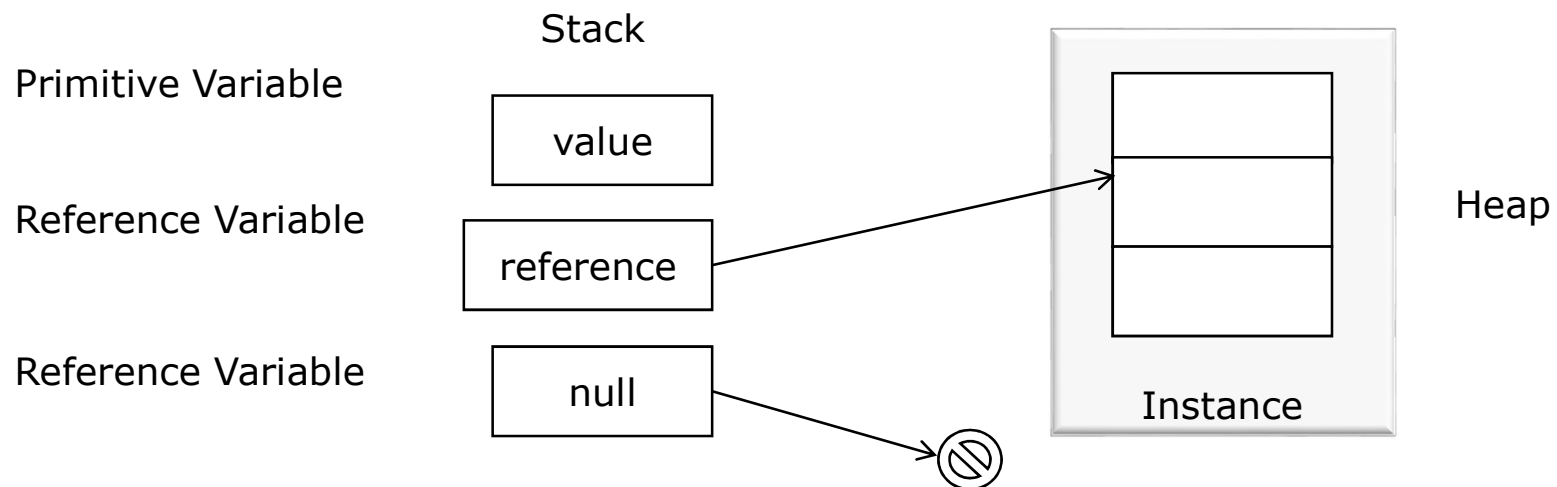
Variables are data placeholders.

Java is a strongly typed language, therefore every variable must have a declared type.

The variables can be of two types:

- reference types: A variable of reference type provides a reference to an object.
- primitive types: A variable of primitive type holds a primitive.

In addition to the data type, a Java variable also has a name or an identifier.





3.4: Variables and Literals

Types of Variables

Variable is basic storage in a Java program

Three types of variables:

- Instance variables
 - Instantiated for every object of the class
- Static variables
 - Class Variables
 - Not instantiated for every object of the class
- Local variables
 - Declared in methods and blocks



3.4: Variables and Literals

Types of Variables

```
public class Box {  
    private double dblWidth;  
    private double dblHeight;  
    private double dblDepth;  
    private static int boxid;  
    public double calcVolume() {  
        double dblTemp;  
        dblTemp = dblWidth * dblHeight * dblDepth;  
        return dblTemp;  
    }  
}
```

Instance Variable

Static Variable

Local Variable



3.4: Variables and Literals

Literals

Literals represents value to be assigned for variable.

Java has three types of literals:

- Primitive type literals
- String literals
- null literal

Primitive literals are further divided into four subtypes:

- Integer
- Floating point
- Character
- Boolean

For better readability of large sized values, Java 7 allows to include '_' in integer literals.



3.5: Flow Control: Java's Control Statements

Control Statements

Use control flow statements to:

- Conditionally execute statements
- Repeatedly execute a block of statements
- Change the normal, sequential flow of control

Categorized into two types:

- Selection Statements
- Iteration Statements



3.5: Flow Control: Java's Control Statements

Selection Statements

Allows programs to choose between alternate actions on execution.

"if" used for conditional branch:

```
if (condition) statement1;  
else statement2;
```

"switch" used as an alternative to multiple "if's":

```
switch(expression){  
    case value1:    //statement sequence  
                    break;  
    case value2:    //statement sequence  
                    break; ...  
    default:        //default statement sequence  
}
```

**Expression can be
of String type!**



3.5: Flow Control: Java's Control Statements

switch case : an example

```
class SampleSwitch {  
    public static void main(String args[]) {  
        for(int i=0; i<=4; i++)  
            switch(i) {  
            case 0:  
                System.out.println("i is zero."); break;  
            case 1:  
                System.out.println("i is one."); break;  
            case 2:  
                System.out.println("i is two."); break;  
            case 3:  
                System.out.println("i is three."); break;  
            default:  
                System.out.println("i is greater than 3.");  
            }  
        }  
    }  
}
```

Output:

i is zero.
i is one.
i is two.
i is three.
i is greater than 3.



3.5: Flow Control: Java's Control Statements

Iteration Statements

Allow a block of statements to execute repeatedly

- While Loop: Enters the loop if the condition is true

```
while (condition)
{ //body of loop
}
```

- Do – While Loop: Loop executes at least once even if the condition is false

```
do
{ //body of the loop
} while (condition)
```



3.5: Flow Control: Java's Control Statements

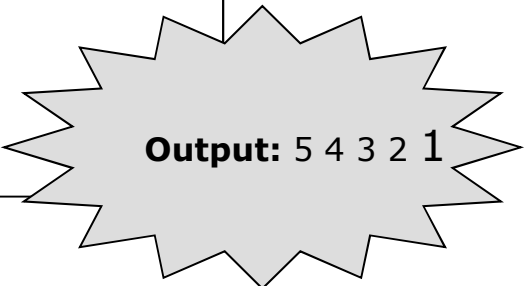
Iteration Statements

- For Loop:

```
for( initialization ; condition ; iteration)
{ //body of the loop }
```

- Example

```
// Demonstrate the for loop.
class SampleFor {
    public static void main(String args[]) {
        int number;
        for(number =5; number >0; n--)
            System.out.print(number +"\t");
        }
    }
```



Output: 5 4 3 2 1



3.5: Control Structures Demo

Data types in Java

Switch Statement using String as expression



3.6: Best Practices

Best practices: Iteration Statements

Always use an int data type as the loop index variable whenever possible

Use for-each liberally

Switch case statement

Terminating conditions should be against 0

Loop invariant code motion

E.g If you call `length()` in a tight loop, there can be a performance hit.



3.6: Best Practices

Best practices: Iteration Statements

Notes Pages

Summary



In this lesson you have learnt:

- Keywords
- Primitive Data Types
- Operators and Assignments
- Variables and Literals
- Flow Control: Java's Control Statements
- Best Practices



Review Question

Question 1: Java considers variable number and NuMbEr to be identical.

- True/False

Question 2: The *do...while* statement tests the loop-continuation condition _____ it executes executing the loop's body; hence, the body executes at least once.

- **Option1:** before
- **Option2:** after