Core Java

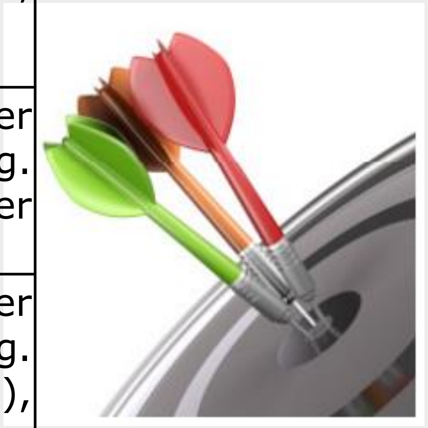
Lesson 03 : Basic Language Constructs

Lesson Objectives

- After completing this lesson, participants will be able to:
  - Naming Conventions
  - Variables and Data Types
  - Operators and Assignments
  - Promotion and Demotion Rules in Java
  - Loops in Java
  - Conditional Statements
  - break and Continue statements
  - Reference Variables
  - Arrays

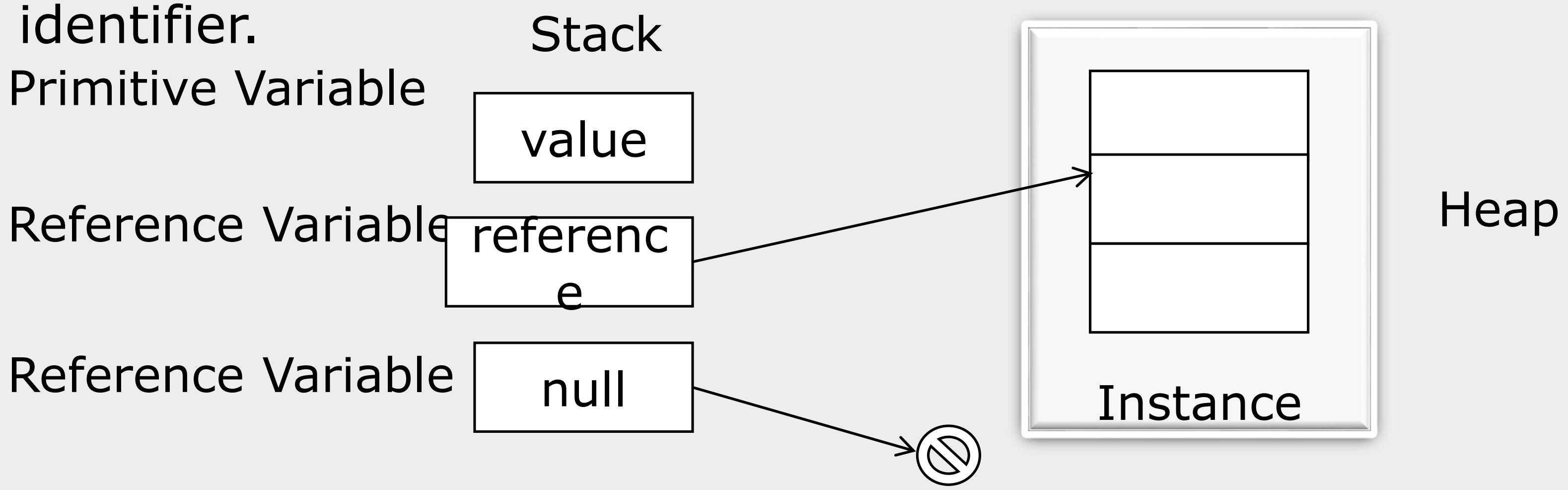
Naming Conventions in Java

class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package name	should be in lowercase letter e.g. java, lang, sql, util etc.
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.



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.



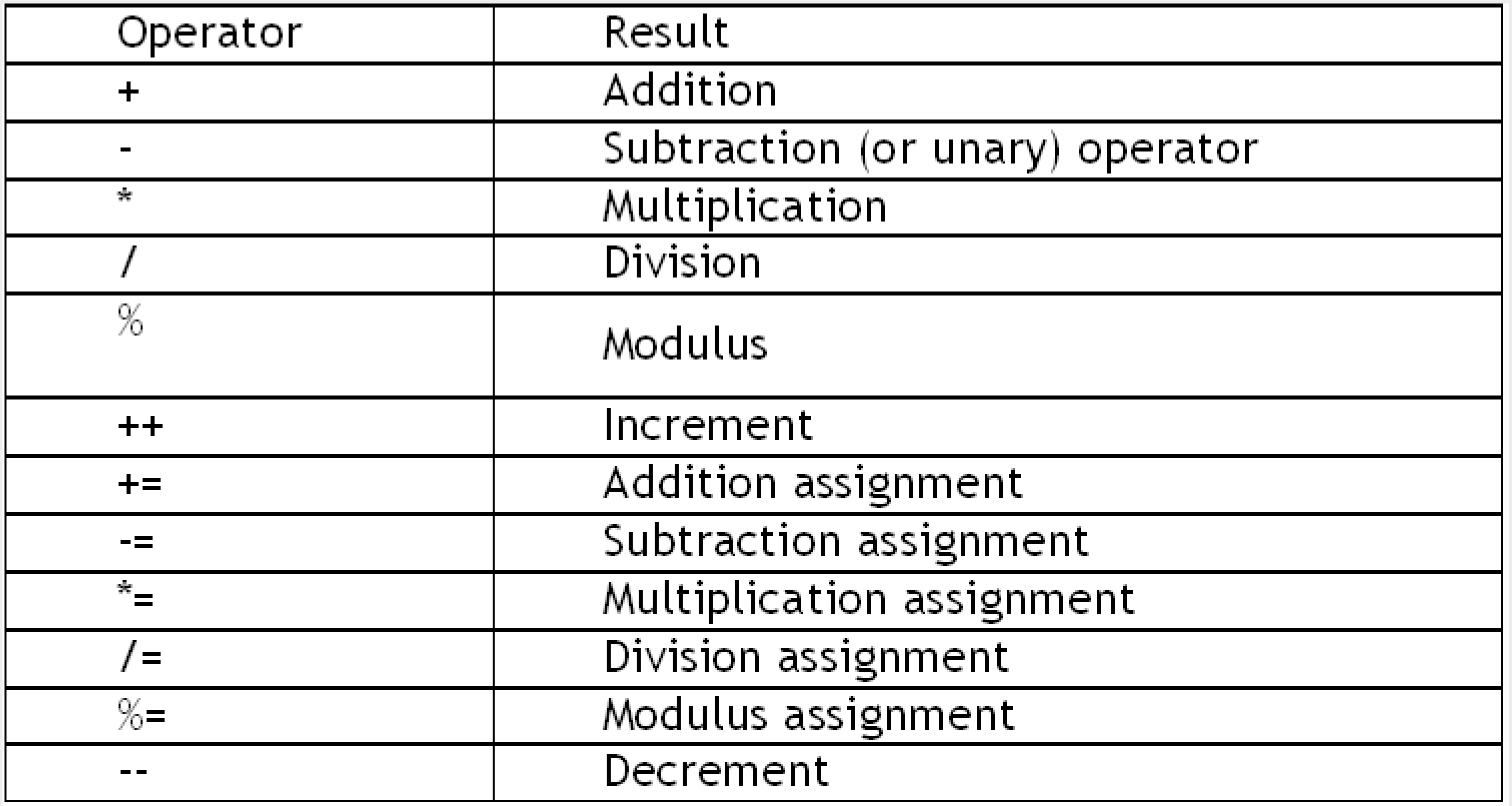
Java Data types

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

Operators in Java

- Operators can be divided into following groups:
  - Arithmetic
  - Bitwise
  - Relational
  - Logical

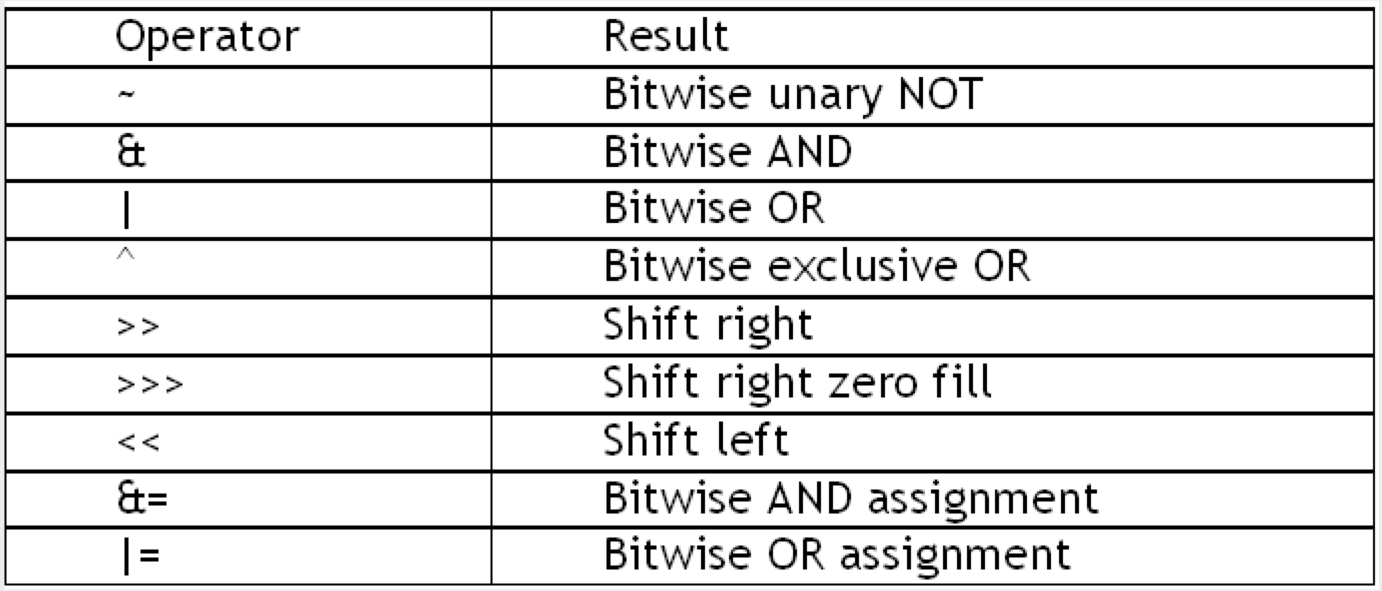
Arithmetic Operators



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

Bitwise Operators

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



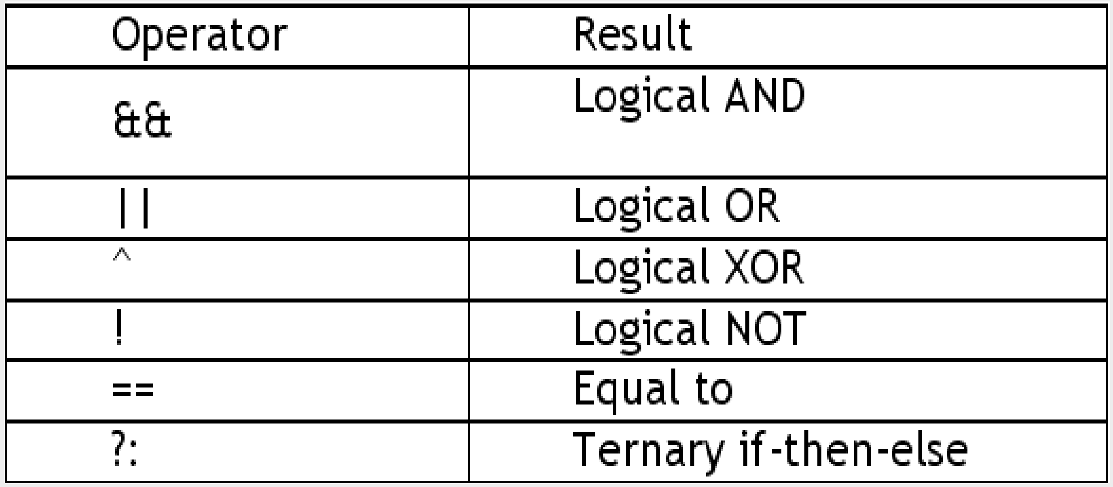
Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
|=	Bitwise OR assignment

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

Logical Operators



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

Demo

- Execute Following Program:
  - AssignmentOperator.java
  - BasicArithmeticOperator.java
  - CommaOperator.java
  - TernaryOperator.java

Promotion and Demotion Rules in Java

➢ Promotion Rules-

A widening (Promotion) primitive conversion does not lose information about the overall magnitude of a numeric value.

- All byte and short values are promoted to int.
- If one operand is a long, the whole expression is promoted to long.
- If one operand is a float, the entire expression is promoted to float.
- If any of the operands is double, the result is double.

➢ Demotion Rules-

Narrowing(demotion) primitive conversion may lose information about the overall magnitude of a numeric value and may also lose precision and range

- int to byte, short, or char
- long to byte, short, char, or int
- float to byte, short, char, int, or long
- double to byte, short, char, int, long, or float

Looping 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)
```

Looping 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

Demo

- DoWhileEg.java
- ForEg.java
- WhileEg.java

Conditional 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!

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.
```

break and continue Statement

The break keyword is used to break(stopping) a loop execution ,which may be a for loop ,while ,do while.

The continue keyword is used to skip the particular iteration only in a loop execution which may be for loop, while or do while loop.

Demo

```
class BreakAndContinue {
 public static void main(String args[]) {
System.out.println("Break Statement\n...................");
for(int i=1;i<=5;i++) {
 if(i==4) break;
 System.out.println(i);
}
// System.out.println("Continue Statement\n...................");
 for(int i=1;i<=5;i++) {
if(i==1)
 continue;
System.out.println(i);
 }
 }
 }
```

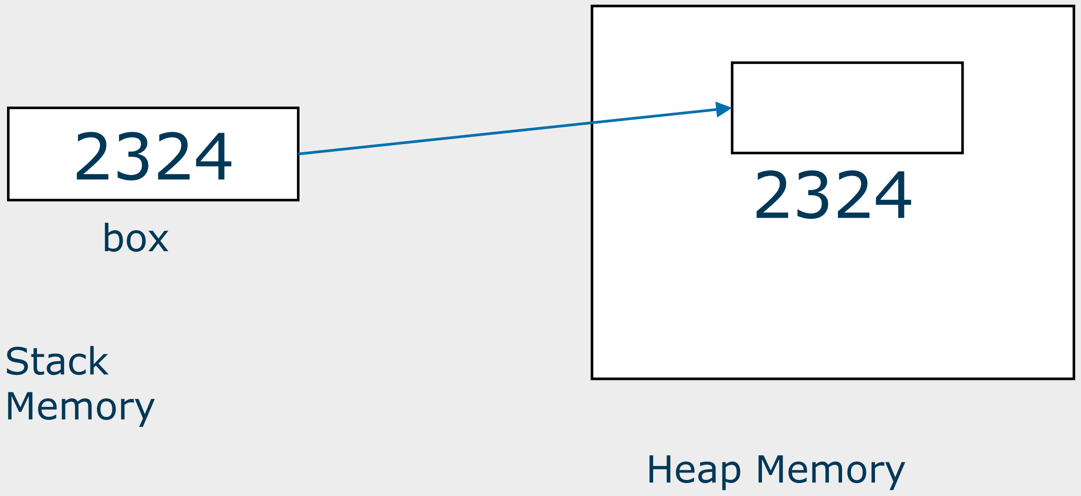
Demo

- Data types in Java
- UsingStringInSwitch.java
- SwitchExample.java
- IfExample.java

Reference Variables in Java

- Reference variables are used to refer to an object. They are declared with a specific type which cannot be changed.

- Box box= new Box();



Arrays

- A group of like-typed variables referred by a common name
- Array declaration and initialization:
  - int arr [];
    arr = new int[10];
  - int arr[] = {2,3,4,5};
  - int twoDim [][] = new int[4][5];

Creating Array Objects

➢ Arrays of objects too can be created:

- Example 1:

```
Box barr[] = new Box[3];
barr[0] = new Box();
barr[1] = new Box();
barr[2] = new Box();
```

- Example 2:

```
String[] Words = new String[2];
Words[0]=new  String(“Bombay”);
Words[1]=new  String(“Pune”);
```

Demo

- Executing the ArrayDemo.java program

Lab

- Lab 3

Summary

- In this lesson you have learnt:
  - Keywords
  - Primitive Data Types
  - Operators and Assignments
  - Variables and Literals
  - Flow Control: Java’s Control Statements
  - Arrays

Review Question

- Question 1: 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

- Question 2: If a display method accepts an integer array and returns nothing , is following call to display method is correct? State true or false.
  - display({10,20,30,40,50})