Data Types, Literals, Operators

Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283

Learning - Topics

- Data Types
  - Why Data Types
  - Primitive Data Type
  - Reference Data Types
- Identifiers
  - Naming Standards
  - Naming Conventions
- Constant Values - Literals
- Variables
- Operator Types (Unary, Binary and Tertiary)
- Operators
  - Arithmetic Operators
  - Assignment Operator
  - Relational Operators
  - Logical Operators
  - Bit Wise Operators
- Type Casting

Java Data Types

There are two data types available in Java:

- Primitive Data Types
- Reference/Object Data Types

1. Primitive Data Types:

- There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named as keyword

Primitive Data Types

Primitive Data Types

Keyword	Description	Size/Format
(integers)		
`byte`	Byte-length integer	8-bit two's complement
`short`	Short integer	16-bit two's complement
`int`	Integer	32-bit two's complement
`long`	Long integer	64-bit two's complement
(real numbers)		
`float`	Single-precision floating point	32-bit IEEE 754
`double`	Double-precision floating point	64-bit IEEE 754
(other types)		
`char`	A single character	16-bit Unicode character
`boolean`	A boolean value (`true` or `false`)	true or false

Data Types: Default Values

Rule :- Every data types in java got default value

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Reference Data Types

- Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, Employee, Puppy etc.
- Class objects, and various type of array variables come under reference data type.
- Default value of any reference variable is null.
- A reference variable can be used to refer to any object of the declared type or any compatible type.
- Example :

```
Animal  animal = new Animal("giraffe");
```

Java Keywords

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

Identifiers- Naming Standard

- The identifiers are the names of the variables, methods, classes, packages and interfaces.
- The identifiers must be composed of the letters, numbers, the underscore _ and the dollar sign $.
- The identifiers may only begin with the letter, underscore or dollar sign.
- The following are legal variable names:
  - MyVariable, myvariable , MYVARIABLE, x, I, _myvariable , $myvariable , _9pins, andros
- The following are not the legal variable names
  - My Space // Contains a space
  - 9pin6s // Begins with a digit
  - k+h // The plus sign is not an alphanumeric character testing
  - 61-72-83 // The hyphen is not an alphanumeric character
  - O'Reilly_this'my // Apostrophe is not an alphanumeric character
  - OReilly_&_Associates // ampersand is not an alphanumeric character>

Identifier-Naming Conventions

Identifier Type	Rules for Naming	Examples
Packages	The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese
Classes	Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations	class Raster; class ImageSprite;
Interfaces	Interface names should be capitalized like class names.	interface RasterDelegate; interface Storing;

Methods	Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.	run();
		runFast();
		getBackground();

Variables	variables are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables.	int i; char c; float myWidth;

Constants	The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_"). (ANSI constants should be avoided, for ease of debugging.)	static final int MIN_WIDTH = 4;
		static final int MAX_WIDTH = 999;
		static final int GET_THE_CPU = 1;

Literals – Constant Values

- A constant value in a program is denoted by a *literal.* Literals represent numerical (integer or floating-point), character, boolean or string values.

  Example of literals:

  Integer literals:
  33 0 -9

  Floating-point literals:
  .3 0.3 3.14

  Character literals:
  '(' 'R' 'r' '{'

  Boolean literals:(predefined values)
  true false

  String literals:
  "language" "0.2" "r" ""

Integer Literals

- Any whole number value is an integer literal.
- Octal values are denoted in Java by a leading zero. Eg 056
- A hexadecimal constant with a leading zerox,

(**0x or 0X)**

- To specify integer literals using binary, prefix the value with **0b or 0B eg x = 0b1010 (JDK7.0 +)**
- To specify a **long literal, it must be** appended with an upper- or lowercase *L to the literal*
- Underscores can only be used to separate digits. int x = 123___456___789 **(JDK7.0 +)**

Floating Literals

- Can be represented in standard or scientific notation
- Standard eg 2.0, 3.145
- Scientific eg 6.022E23
- Floating-point literals in Java default to **double precision**
- **To specify a float** literal, append an *F or f to the* constant.
- To specify a **double** literal, append a ***D*** *or* ***d*** *to the constant*

Boolean, Character and String

- Boolean
  - Only two values (true and false)
- Character
  - A character is quoted in single quote (')
- String
  - A string literal is a sequence of characters which has to be double-quoted (") and occur on a single line

Variables

- A piece of your computer's memory that is given a name and type and can store a value.
  - Usage:
    - compute an expression's result
    - store that result into a variable
    - use that variable later in the program
  - Unlike a calculator, which may only have enough to store a few values, we can declare as many variables as we want.

Declaring variables

- **Variable declaration statement**: A Java statement that creates a new variable of a given type.
  - A variable is *declared* by writing a statement that says its type, and then its name.
  - Variables must be declared before they can be used.

- Declaration statement syntax:

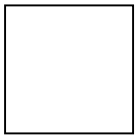
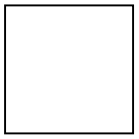
  <type> <name> ;

  - The <*name*> can be any identifier.
  - Examples:
```
int x;
double myGPA;
```

More on declaring variables

- Declaring a variable sets aside a piece of memory in which you can store a value.

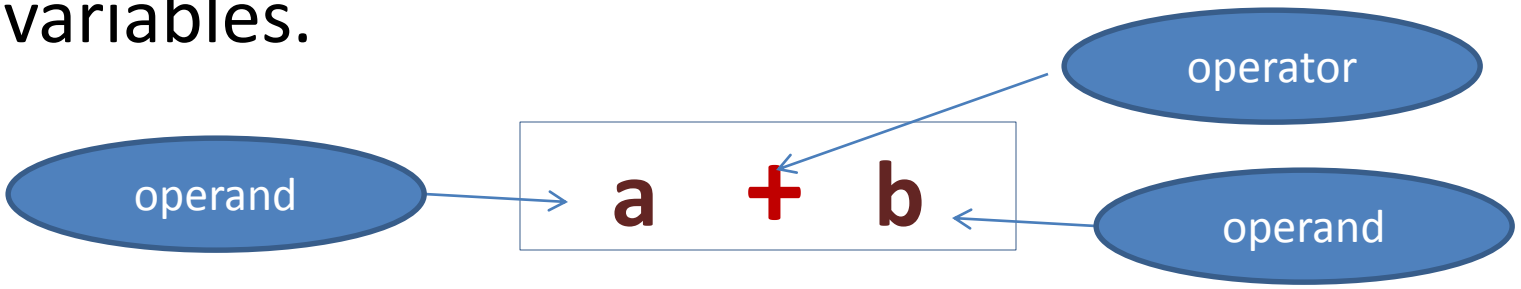
```
int x;
int y;
```

  - Part of the compute
  - x  y 

(The memory has no values in it yet.)

Java Basic Operators :

Java provides a rich set of operators to manipulate variables.



We can divide Java operators into the following groups:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Misc Operators
- Bitwise Operators

Arithmetic Operators

- Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:
- Assume A has value 10 and B has value 20

Operator	Description	Example
+	Addition - Adds values on either side of the operator	A + B will give 30
-	Subtraction - Subtracts right hand operand from left hand operand	A - B will give -10
*	Multiplication - Multiplies values on either side of the operator	A * B will give 200
/	Division - Divides left hand operand by right hand operand	B / A will give 2
%	Modulus - Divides left hand operand by right hand operand and returns remainder	B % A will give 0
++	Increment - Increase the value of operand by 1	B++ gives 21
--	Decrement - Decrease the value of operand by 1	B-- gives 19

Relational Operators

- There are following relational operators supported by Java language
- Assume A has value 10 and B has value 20

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

- The following table lists the logical operators:
- Assume boolean variables A holds true and variable B holds false then:

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

- **The Assignment Operators:**
  - There are following assignment operators supported by Java language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator	C |= 2 is same as C = C | 2

- **Misc Operators**
  - There are few other operators supported by Java Language.

- **Conditional Operator (? :):**
  - Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate boolean expressions. The goal of the operator is to decide which value should be assigned to the variable. The operator is written as :

```
variable x = (expression)  ?  Value  if  true : value  if  false
```

- Following is the example:

```
public class Test
 {
 public static void main(String  args[])
{
        int a , b; a = 10; b = (a == 1) ? 20: 30;
         System.out.println( "Value of b is : " + b );
        b = (a == 10) ? 20: 30;
         System.out.println( "Value of b is : " + b );
} }
```

- This would produce following result:

```
Value of b is : 30
Value of b is : 20
```

- **The Bitwise Operators:**
  - Java defines several bitwise operators which can be applied to the integer types, long, int, short, char, and byte.
  - Bitwise operator works on bits and perform bit by bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows:

a = 0011 1100

b = 0000 1101

---

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

The Bitwise Operators:

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in eather operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the efect of 'flipping' bits.	(~A) will give -60 which is 1100 0011
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>>	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

Type Casting

Type casting

- Conversion of one data type into another data type is called type casting

```
int m=10;
int f=3;
int r=m/f;
```

```
int m=10;
int f=3;
float r=float m/f;
```

Types

- Implicit type casting

  Byte b=10;

  Int c=b;

- Explicit type casting

  int x=10;

  Byte b=(byte)x;