JAVA Programming Fundamentals

JAVA character set

Character set is a set of valid characters

- that a language can recognize. It may be any letter, digit or any symbol or sign.
- JAVA uses 2-Byte UNICODE character set, which supports almost all characters in almost all languages like English,Chinese, Arbic etc.
- In Unicode, first 128 characters are similar to ASCII character set. Next 128 character equal to Extended ASCII code. Rest capable to support other languages. Any character in Unicode can be represented by \u followed by 4 digit Hexadecimal

number. E.g. \u0394 to represent Delta Symbol.

JAVA Tokens

- The smallest individual unit in a program is known as Token. It may any word, symbols or punctuation mark etc.
- Following types of tokens used in Java-
  - Keywords
  - Identifiers
  - Literals
  - Punctuators (; [] etc)
  - Operators (+, -,/,*, =, == etc.)

Keywords in Java

- Keywords are the reserve words that have a special meaning to the compiler.
- Key words can't be used as identifiers or variable name etc.
- Commonly used key words are-

  char, long, for, case, if, double, int, short, void, main, while , new etc.

Identifiers in Java

- Identifiers are fundamental building block of program and used as names given to variables, objects, classes and functions etc.
- The following rules must be followed while using identifiers.
  - Identifiers may have alphabets, digits and dollar ($), underscore (_) sign.
  - They must not be Java keywords.
  - They must not begin with digit.
  - They can be of any length.
  - They are Case Sensitive ie. Age is different from age.
- Example of Valid identifiers-

  MyFile, Date9_7_7, z2t09, A_2_Z, $1_to_100, _chk etc.
- Example of Invalid identifiers-

  Date-RAC, 29abc, My.File, break, for

Literals in Java

- Literals or constants are data items that have fixed data value.
- Java allows several types of literals like-
  - Integer Literals
  - Floating Literals
  - Boolean Literals
  - Character Literals
  - String Literals
  - The null literals

Integer Literals

- An integer constant or literals must have at least one +/- digit without decimal point.
- Java allows three types of integer literals -
  - Decimal Integer Literals (Base 10)

    e.g. 1234, 41, +97, -17 etc.
  - Octal Integer Literals (Base 8)

    e.g.010, 014 (Octal must start with 0)
  - Hexadecimal Integer Literals (Base 16)

    e.g. 0xC, 0xab (Hex numbers must starts with 0x)
- L or U suffix can used to represent long and unsigned literals respectively.

Floating / Real Literals

- A real literals are fractional numbers having at least one digit before and after decimal point with + or - sign.

  The following are valid real numbers-

  2.0, 17.5, -13.0. -0.00626

  The following are invalid real numbers- 7, 7. , +17/2, 17,250.26 etc.

- A real literals may be represented in Exponent form having Matissa and exponent with base 10 (E). Mantissa may be a proper real numbers while exponent must be integer.

  The following are valid real in exponent form-

  152E05, 1.52E07, 0.152E08, -0.12E-3, 1.5E+8

  The following are invalid real exponent numbers-

172.E5, 1.7E, 0.17E2.3, 17,22E05, .25E-7

Other Literals

- The Boolean Literals represents either TRUE or FALSE. It always Boolean type.
- A null literals indicates nothing. It always null type.
- Character Literals must contain one character and must enclosed in single quotation mark.

  e.g. 'a', '%' , '9' , '\\' etc.

  Java allows some non-graphic characters (which can not be typed directly through keyboard) by using Escape sequence (\) .

  E.g.

  \a (alert), \b (backspace), \f (Form feed),
  \n (new line), \r (return key), \t (Horizontal tab),
  \v (vertical tab), \\ (back slash), \' (single quote) ,
  \" (double quote) , \? (question mark), \0 (null) etc.
- String Literals is a sequence of zero or more characters enclosed in double quotes. E.g. "abs", "amit" , "1234" , "12 A" etc.

Concept of Data types

Data types are means to identify the type of data and associated operations of handling it.

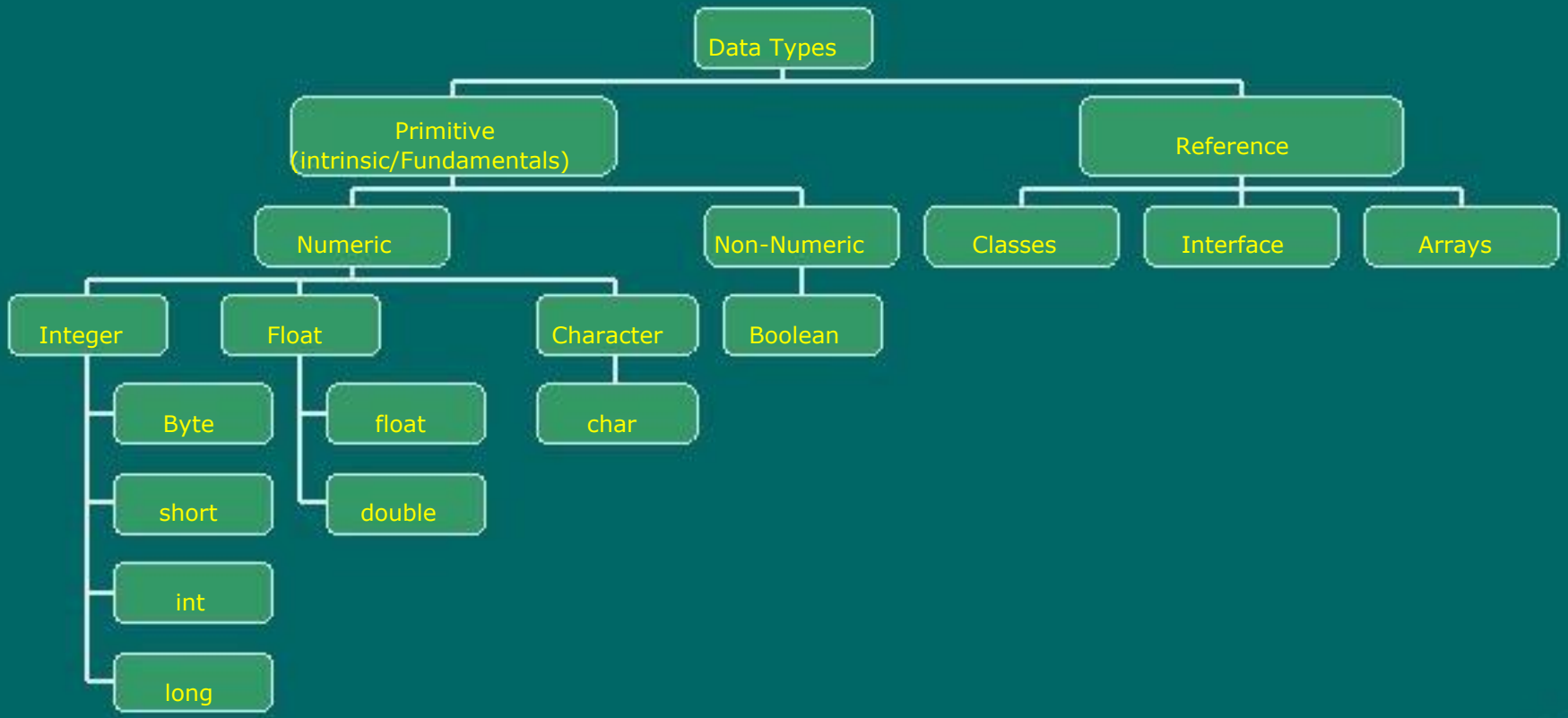
Java offers two types of data types.

- Primitive:
- These are in-built data types offered by the compiler. Java supports 8 primitive data types e.g. byte, short, int, long, float, double, char, boolean.
- Reference:

These are constructed by using primitive data types, as per user need. Reference data types store the memory address of an object.

Class, Interface and Array are the example of Reference Data types.

Data Types in Java



String Data type is also used in Java as Reference data type

Primitive Data types

Type	Size	Description	Range
byte	1 Byte	Byte integer	-128 to +127
short	2 Byte	Short integer	-32768 to +32767
int	4 Byte	integer	-2_{31} to $2_{31}-1$
long	8 Byte	Long integer	-2_{63} to $2_{63}-1$
float	4 Byte	Single precision floating point (up to 6 digit)	-3.4E+38 to +3.4E+38
double	8 Byte	Double precision floating (up to 15 digit)	-1.7E+308 to 1.7E+308
char	2 Byte	Single character	0 to 65536
Boolean	1 Byte	Logical Boolean values	True or False

- L suffix can used to indicate the value as long.
- By default Java assume frictional value as double, F and D suffix can be used with number to indicate float and double values respectively.

Working with Variables

- A variable is named memory location, which holds a data value of a particular data type.
- Declaration and Initialization of variable-

  <data type> <variable Name>;

Example:

```
int age;
double amount;
double salary, wage;
double price=214.70, discount =0.12;
String name="Amitabh"
long x=25L;
byte a=3;
float x= a+b;
```

- By default all Numeric variables initialized with 0, and character and reference variable with null, boolean with false, if it is not initialized.
- The keyword final can be used with variable declaration to indicate the value stored on a variable can't be changed i.e. it will be constant.

  E.g. final double SERVICE_TAX=0.020

Text interaction in GUI

- In GUI application often we require to store the values of text fields to variable or vice-versa. Java offers three method for this purpose-
- getText():

  It returns the text stored in the text based GUI components like Text Field, Text Area, Button, Label, Check Box and Radio Button etc. in string type.

e.g. String str1=jTextField1.getText();

- parse□ □ .()

  This method convert textual data from GUI component in to numeric type.

  Byte.parseByte(String s) - string into byte.
  Short.parseShort(String s) - string into short.
  Integer.parseInt(string s) - string into integer.
  Long.parseLong(string s) - string into long.
  Float.parseFloat(string s) - string into float.
  Double.parseDouble(string s) - string into double.

e.g. int age=Integer.parseInt(jTextField1.getText());

- setText()

  This method stores string into GUI component.

e.g. jTextField1.setText("Amitabh");
jLabel1.setText(""+payment);

A sample Java Program

```
import java.io.*;
class Program2
{
   public static void main(String arg[]) throws IOException
    {
      int a=5,x;
      float b=3.5;
      char c='a';
      double d=4.5;
      String s1="Hello";
      String s2="56";
     x=Integer.parseInt(s2);
     System.out.println("interger"+a);
     System.out.println("float"+b);
     System.out.println("character"+c);
     System.out.println("double"+d);
     System.out.println(s1);
    System.out.println(""+x);
   }
}
```

System.out.println() and System.out.print() is used to get output on console window.

Operators in Java

- The operators are symbols or words, which perform specified operation on its operands.
- Operators may Unary, Binary and Turnery as per number of operands it requires.
- Java offers the following types of Operators:-
  - Arithmetic Operator
  - Increment/Decrement Operator
  - Relational or Comparison Operators
  - Logical Operators
  - Assignment Operators
  - Other Operators.

Arithmetic Operators

+	Unary plus	Represents positive values.	int a=+25
-	Unary minus	Represents negative values.	int a=-25
+	Addition	Adds two values	int x= a+b;
-	Subtraction	Subtract second operands from first.	int x=a-b;
*	Multiplication	Multiplies two values	int x= a*b;
/	Division	Divides first operand by second	int x=a/b;
%	Modulus (remainder)	Finds remainder after division.	int x= a%b;
+	Concatenate or String addition	Adds two strings	"ab"+"cd" =>"abcd" "25"+"12" =>"2512" ""+5 =>"5" ""+5+"xyz" =>"5xyz"

Increment & Decrement Operator

- Java supports ++ and -- operator which adds or subtract 1 from its operand. i.e.

  a=a+1 equivalent to ++a or a++

  a=a-1 equivalent to --a or a--

- ++ or -- operator may used in Pre or Post form.

  ++a or --a (increase/decrease before use)

  a++ or a- (increase/decrease after use)

  Ex. Find value of P? (initially n=8 and p=4)

  p=p* --n; => 28

  p=p* n--; => 32

  Ex. Evaluate x=++y + 2y if y=6.

  =7+14 = 21

Relational Operator

- Relational operators returns true or false as per the relation between operands.
- Java offers the following six relational operators.

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Relational operators solved from left to right.

Ex: 3>=3 true

3!=3 false

a==b true if a and b have the same value.

a<b<c => (a<b)<c true if a I smallest.

Logical Operator

- Logical operators returns true or false as per the condition of operands. These are used to design more complex conditions.
- Java offers the following six (5 binary and 1 unary) logical operators.

Operator	Name	use	Returns true if
&&	And	x&&y	X and y both true
||	Or	x||y	Either x or y is true
!	Not	!x	X is false
&	Bitwise and	x&y	X and y both true
|	Bitwise or	x|y	Either x or y is true
^	Exclusive or	x^y	If x and y are different

Ex: 5>8 || 5<2 (false)

6<9 && 4>2 (true)

!(5!=0) (false)

1==0||0>1 (false)

6==3&&4==4 (false)

!(5>9) (true)

Assignment Operator

- In Java = operator is known as Assignment operator, it assigns right hand value to left hand variables.

  Ex: int x=5;

  z= x+y;

- Java offers some special shortened Assignment operators, which are used to assign values on a variable.

Operator	use	Equivalent to
+=	X+=y	X=x+y
-=	X -=y	X=x-y
*=	X*=y	X=x*y
/=	x/=y	X=x/y
%=	X%=y	X=x%y

Ex: x-=10 => x=x -10

x%=y => x=x%y

Other Operators

- In Java some other operators are also used for various operations. Some operators are-

Operator	Equivalent to
? :	Shortcut of If condition (turnery operator) <Condition> ? <true action>:<false action>
[]	Used to declare array or access array element
.	Used to form qualified name (refer)
(type)	Converts values as per given data type
new	Creates a new object
instanceof	Determin es whether the first operator is instance of other.
<<,>>	Performs bitwise left shift or right shift operation.
~	(compliment) Inverts each bit (0 to 1 or 1 to 0)

Ex. result = marks>=50 ? 'P' : 'F'

6>4 ? 9:7 evaluates 9 because 6>4 is true.

Operator's Precedence

- Operator's precedence determines the order in which expressions are evaluated. There is certain rules for evaluating a complex expression.
  e.g. y=6+4/2 (why 8 not 5 ?)

Operators	Remark	Associativity
. [] ()	() used to make a group, [] used for array and . Is used to access member of object	L to R
++ -- ! ~	Returns true or false based on operands	R to L
New (type)	New is used to create object and (type) is used to convert data into other types.	R to L
* / %	Multiplication, division and modulus	L to R
+ -	Addition and Subtraction	R to L
<< >>	Bit wise left and right shift	L to R
== !=	Equality and not equality	L to R
&	Bitwise And	L to R
^	Bitwise Exclusive Or	L to R
|	Bitwise or	L to R
&&	Logical And	L to R
||	Logical or	L to R
? :	Shortcut of IF	R to L
= += -= *= /= %=	Various Assignment operators	R to L

Expression in Java

- An expression is a valid combination of operators, constants and variable and keywords i.e. combination of Java tokens.
- In java, three types of expressions are used.

Arithmetic Expression

Arithmetic expression may contain one or more numeric variables, literals and operators. Two operands or operators should not occur in continuation.

e.g. x+*y and q(a+b-z/4) is invalid expressions.

Pure expression: when all operands are of same type.

- Mixed expressions: when operands are of different data types.

Compound Expression

It is combination of two or more simple expressions.

- E.g. (a+b)/(c+d) (a>b)||(b<c)

Logical Expression

Logical or Boolean expression may have two or more simple expressions joined with relational or logical operators. E.g.

x>y (y+z)>=(x/z) x||y && z (x) (x-y)

Type Conversion in JAVA

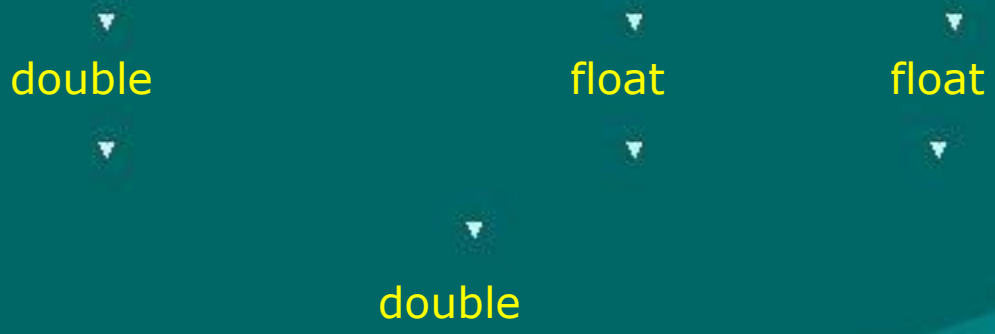
- The process of converting one predefined type into another is called type conversion.
- In mixed expression, various types of constant and variables are converted into same type before evaluation.
- Java facilitates two types of conversion.
- Implicit type conversion
- Explicit type conversion

Implicit Type Conversion

- It is performed by the compiler, when different data types are intermixed in an expression.
- In Implicit conversion, all operands are promoted (Coercion) up to the largest data type in the expression.

Ex. Consider the given expression, where f is float, d is double and I is integer data type.

Result= (f * d) - (f + i)+ (f / i)



Explicit Conversion in JAVA

- An explicit conversion is user defined that forces to convert an operand to a specific data type by (type) cast. Ex. (float) (x/2) suppose x is integer.

  The result of x/2 is converted in float otherwise it will give integer result.

- In pure expression the resultant is given as expression's data type.

E.g. 100/11 will give 9 not 9.999 (since both are integer)

- In mixed expression the implicit conversion is applied (largest type promotion)

E.g. int a, mb=2, k=4 then evaluate a=mb*3/4+k/4+8-mb+5/8

= 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

= 6 / 4 + 1 + 8 - 2 + 5 / 8

= 1 + 1 + 8 - 2 + 0 (6/4 will give 1) = 8

JAVA Statements

- A statement in Java is a complete unit of execution. It may consists of Expression, Declaration, Control flow statements and must be ended with semicolon (;)
- Statements forms a block enclosed within { }. Even a block may have no statement (empty).

  E.g.
```
If(a>b)
{ .
  . 
}
```
- Note:

System.out.print('h'+'a') will give 169

System.out.print("+'h'+'a') will give ha

System.out.print("2+2="+2+2) will give 2+2=22

System.out.print("2+2="+(2+2)) will give 2+2=4