Basics:

1. Identifiers
2. Reserved Words
3. Data Types
4. Literals
5. Arrays
6. Types of Variables
7. Var-arg methods
8. Main method
9. Command line arguments
10. Java coding standards
11. Identifiers:

Any name (Class name, method name, variable name, label name ….) in java programs which can be used for identifying purpose.

Ex:

Class Test{

Public static void main(String[] args){

Int I = 0;

}

}

In above code there are 5 identifiers are there

1. Test (Class name)
2. Main (Method name)
3. String – is pre-defined name in java)
4. args – Array nae
5. I – variable name
   1. **Allowed Identifiers :**

A to Z

a to z

0 to 9

$

\_

* 1. **Rules to write identifier:**

Should start with alphabets

Shouldn’t starts with number or $ or \_

* 1. Java is case-sensitive
  2. No length limit for identifier [But it’s not GPP to use lengthy identifiers]
  3. We shouldn’t use reserved keywords as identifier.
  4. But we can use pre-defined class names and interface names as identifiers[But its’ not GPP which causes confusion and readability]

EX : int String = 111,;

Int Runnable = 000;

**2. Reserved KeyWords (53)**

2.1. Keyworsd(50)

2.2.Reserved Literals(3)

true,false,null

**3.DataTypes**

**3.1.** In java every variable and expression has some type.

Int a,b,c;

a+b+c; This expression type is Int]

3.2. Each and every data type is clearly defined.

3.3. Every assignment should be checked by compiler for type compatibility.

Because of above reasons we can conclude JAVA is strongly typed programming language.

**3.2. Primitive Data Types:**

3.2.1.Numeric Data Types :

3.2.1.1. Integral Data Types : byte (8 bits -128 to 127), short,int,long

3.2.1.2.Floating Data Types : float,double

3.2.2. Non-Numeric Data Types : char,boolean

Except boolean and char, remaining data types are considered as signed datatypes. Because we can represent both +ve and –ve numbers.

**byte : (8 bits -128 to 127) : Best choice if** wants to handle data in terms of streams either from the file or from the network. [File supported and n/w supported form is byte]

**short : size - 2 bytes(16 bits) - -2 power 15 [-32768] to (2 power 15 – 1) [32767]) :**

**int : size – 4 bytes (32 bits) - -2 power31[-2147483648] to (2 power 31-1)[2147483647]**

**long : size – 8 bytes (64 bits) - - 2 power 63 to ( 2 power 63-1)**

**float : size – 4 bytes - - 1.7 e 38 to 1.7e38 [1.7e38 means 1.7 \* 10 power 38].**

**After point 5 to 6 decimal places of accuracy**

**double : size – 8 bytes - - 3.4e308 to 3.4e308**

**After point 14 to 15 decimal places of accuracy**

**boolean : size – NA, range – NA [but allowed values are : true/false]**

**char : size – 2 bytes - 0 to 65535**

**Note : 1. default value of all integral data type is 0.**

**2. default value of all floating point data type is 0.**

**3. default value of Boolean is false;**

**4. default value of char is 0 [represents space character]**

**5. null is the default value for object reference and we can’t apply for primitives**

**4.Literals :**

A constant value which can be assigned to the variable is called a literal.

Ex : int x = 10;

Int(datattype) x (identifier]= 10[onstant/literal];

**4.1.Integral Literals:**

1.**Decimal literals** (base-10) [allowed digits 0-9] EX : int x =10

2.**Octal literals**(base-8) [allowed digits 0-7 – base-8] EX : int x = 010.

Literl value should be pre-fixed with 0

3.**HexaDecimal literals** (base-16) [allowed digits 0-9 and a to f] EX : int x = 0X10.

We can use both lower case and upper case letters for extra values.Literal value should be pre- fixed with 0x or 0X

Ex : int x = 10;

Int x = 0786;

Int x = 0XFace;//valid

Inr x = 0XBeer;//invalid

EX : class Test{p s v m(String args[]{

int x=10;

int y = 010; (0\*8 + 1\*8 = 8)

int z = 0X10; (0\*16+1\*16 = 16)

sop(x+”,”+y+”,”+z); //compiler will always prints the value in decimal only

}

O/P : 10,8,16

**4.2.Floating point Literals:**

By default every floating point literal is of double type and hence we can/t assign directly to the float variable. But we can specify floating point literal as float type by suffixed with f or F.

Float f = 123.44456 [not allowed] – default is double hence can’t assign 8 bytes to 4 bytes

Float f = 123.44456f [Allowed]

**Note : We can specify floating point literals only in decimal form and we can’t xpecify in octal and hexa decimal forms**

**4.3.boolean literals:**

Only allowed values for Boolean data types are true/false

**4.4.char literals:**

We can specify char literal within single quotes.

EX : char ch = ‘ a’;

**We can specify char literal as integral literal, which represents Unicode value of the character and that integral literal can be specified either in decimal or octal or hexa-decimal . But allowed range is 0 to 65535**

EX : char ch = 97; sop(ch); //Will print a

char ch = 0XFace; //valid - HexaDecimal

char ch = 077777;//valid - Octal

**c**har ch = 65535; //valid;

char ch = 65536;// Not valid

**We can specify char literal in Unicode representation**

‘\u XXXX’ – 4 digit hexa-decimal number

Ex : char ch = ‘\u0061’;

Sop(ch); a will print// [Value of \u0061 is 97 and 97 respective charcter is a]

**We can specify char literal using escap character**

**EX : char ch = ‘\t’;**

**4.5.String literal :**

**Any sequence of characters in double quotes treated as String literal.**

**EX : String s = “JAK”;**

**Enhancement in literals from 1.7 version :**

1. **Binary literal (binary either 0 or 1) : For intergral data types until 1.6 version we can specify literal value in the following ways.**

**Decimal form, Octal form, Hexadecimal form**

**But 1.7 version onwards we can specify literal value even in Binary form also.**

**Allowed digits are 0 and 1.**

**Literal value should be pre-fixed with 0B or 0b**

**Ex: int x = 0B1111;sop(x); O/P : 15**

1. **Usage of underscore symbol in numeric literals.**

**From1.7 version onwords we can use underscore symbol between digits of numeric literals.**

**EX : double d= 1\_23\_456.7\_8\_9;**

**Double dd = 123\_456.7\_8\_9**

**Note :**

**Byte(1b) 🡪short (2b)🡪int (4b)🡪 float (8b)🡪 double(16b)**

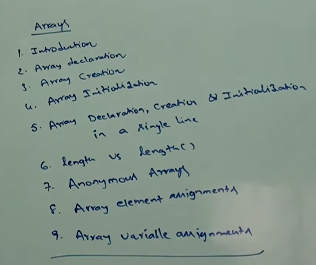
**char (2b)🡪 int 🡪 float 🡪 double**

* **8 Byte long value we can assign to 4 byte float variable because both are following different memory representations internally.**

**float f = 10l;**

**sop(f); //10.0**

**5.Arrays :**

****

**5.1.Introduction :**

An array is an Indexed collection of fixed number of homogeneous data elements

**Advantages :** Can represent huge number of values by using single variable, so that readability of the code will be improved.

**Limitations :** 1. Fixed in size, hence there is o chance of increasing or decreasing the size based on requirement. Hence to use arrays concept compulsory we should know size in advance

2.Can hold only homogeneous data elements

**5.2.Declaration :**

**Single-Dimensional Array :**

Int[] x;//Valid

Int []x;//Valid

Int x[];//Valid

Int[5] x;//Invalid [t the time of declaration we can’t specify the size]

**2D –Array :**

**Int[][] x;** //Valid

**Int [][]x;** //Valid

**Int x[][];**//Valid

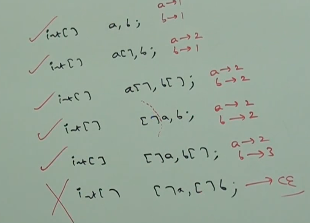
**Int[] []x;** //Valid

**Int[] x[];**//Valid

**Int []x[];**//Valid

**Int[] a,b; a🡪1, b🡪1**

**Which of the following is valid?**

****

If we want to specify dimension before the variable that facility is applicable only for first variable in a declaration. IF we try to apply for remaining variables will get compile time error as shown in last line of above image.

**5.3.Array Creation :**

Every array in java is object only, hence we can create arrays by using new operator.

EX: int[] x = new int[3];

**Int[] x = new int[0]//**valid

**Int[] x = new int[-3**];//No compile time error (as compiler will not bother about sign, its consider only datatype into consideration), but will get runtime exception

**Int[] x = new int[‘a’] //valid – array size is 97**

**byte b = 10;**

**Int[] x = new int[b];//valid**

**short s = 20;**

**int[] x = new int[s];//valid**

**Reason for above acceptability:**

**Byte(1b) 🡪short (2b)🡪int (4b)**

**char (2b)🡪 int**

**int[] x = new int[10L];]]invalid**

Max allowed array size in java is 2147483647, which is the maximum value of int data type;