JAVA – DATA TYPES

Data type is used to allocate sufficient memory space for the data. Data types specify the different sizes and values that can be stored in the variable.

- ***Java is a strongly Typed Language.***
- Definition: ***strongly Typed Language:***

Java is a strongly typed programming language because every variable must be declared with a data type. A variable cannot start off life without knowing the range of values it can hold, and once it is declared, the data type of the variable cannot change.

Data types in Java are of **two types**:

1. **Primitive data types (Intrinsic or built-in types) :- :** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types (Derived or Reference Types):** The non-primitive data types include Classes, Interfaces, Strings and Arrays.

1. Primitive Types:

Primitive data types are those whose variables allow us to store only one value and never allow storing multiple values of same type. This is a data type whose variable can hold maximum one value at a time.

There are eight primitive types in Java:

Integer Types:

1. int
2. short
3. long
4. byte

Floating-point Types:

5. float
6. double

Others:

7. char
8. Boolean

- **Integer Types:**

The integer types are form numbers without fractional parts. Negative values are allowed.Java provides the four integer types shown below:

Type	Storage Requirement	Range	Example	Default Value
int	4 bytes	-2,147,483,648(-2^31) to 2,147,483,647 (2^31-1)	int a = 100000, int b = -200000	0
short	2 bytes	-32,768 (-2^15) to 32,767 (2^15-1)	short s = 10000, short r = -20000	0
long	8 bytes	-9,223,372,036,854,775,808 (-2^63) to 9,223,372,036,854,775,808 (2^63-1)	long a = 100000L, int b = -200000L	0L
byte	1 byte	-128 (-2^7) to 127 (2^7-1)	byte a = 100 , byte b = -50	0

- **Floating-point Types:**

The floating-point types denote numbers with fractional parts. The two floating-pointtypes are shown below:

Type	Storage Requirement	Range	Example	Default Value
float	4 bytes	Approximately ±3.40282347E+38F (6-7 significant decimal digits)	float f1 =234.5f	0.0f
double	8 bytes	Approximately ±1.79769313486231570E+308 (15 significant decimal digits)	double d1 = 123.4	0.0d

- **char:**
  - char data type is a single 16-bit Unicode character.
  - Minimum value is '\u0000' (or 0).
  - Maximum value is '\uffff' (or 65,535 inclusive).
  - Char data type is used to store any character.
  - Example: char letterA ='A'

- **boolean:**
  - boolean data type represents one bit of information.
  - There are only two possible values: true and false.
  - This data type is used for simple flags that track true/false conditions.
  - Default value is false.
  - Example: boolean one = true

2. Derived Types (Reference Types):

- **Derived data types are those whose variables allow us to store multiple values of same type. But they never allow storing multiple values of different types.**
- A reference variable can be used to refer to any object of the declared type or any

compatible type.

- These are the data type whose variable can hold more than one value of similar type.
- **The value of a reference type variable, in contrast to that of a primitive type, is a reference to (an address of) the value or set of values represented by the variable.**
- Example

```
int a[] = {10,20,30};                  // valid
int b[] = {100, 'A', "ABC"};           // invalid
Animal animal = new Animal("giraffe");  //Object
```

JAVA – VARIABLES

- **A Variable is a named piece of memory that is used for storing data in java Program.**
- A variable is an identifier used for storing a data value.
- A Variable may take different values at different times during the execution if the program, unlike the constants.
- The variable's type determines what values it can hold and what operations can beperformed on it.
- **Syntax to declare variables:**

  datatype identifier [=value][,identifier [=value] ...];

- Example of Variable names:

  int average=0.0, height, total height;

- **Rules followed for variable names (consist of alphabets, digits, underscore anddollar characters)**
  1. A variable name must begin with a letter and must be a sequence of letter or digits.
  2. They must not begin with digits.
  3. Uppercase and lowercase variables are not the same.
     a. **Example:** Total and total are two variables which are distinct.
  4. It should not be a keyword.
  5. Whitespace is not allowed.
  6. Variable names can be of any length.

- **Initializing Variables:**
  - ✓ After the declaration of a variable, it must be initialized by means of assignment statement.
  - ✓ It is not possible to use the values of uninitialized variables.

- ✓ Two ways to initialize a variable:

1. Initialize after declaration:

Syntax:

variablename=value;

```
int months;
months=1;
```

2. Declare and initialize on the same line:

Syntax:

Datatype variablename=value;

```
int months=12;
```

- **Dynamic Initialization of a Variable:**

Java allows variables to be initialized dynamically using any valid expression at thetime the variable is declared.

Example: Program that computes the remainder of the division operation:

```
class FindRemainer
{
public static void main(String arg[]) {int num=5,den=2;
int rem=num%den; System.out.println("Remainder is " rem);
}
}
```

Output:

Remainder is 1

In the above program there are three variables **num, den** and **rem. num** and **den** are initialized by constants whereas **rem** is initialized dynamically by the modulo division operation on **num** and **den.**

JAVA - VARIABLE TYPES

There are three kinds of variables in Java:

1. Local variables
2. Instance variables
3. Class/static variables

Local Variables	Instance Variable	Class / Static Variables
Local variables aredeclared in methods, constructors, or blocks.	Instance variables are declared in a class, but outside a method, constructor or any block.	Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block. There would only be one copy of each class variable per class, regardless of how many objects are created from it.
Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor or block.	Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.	Static variables are created when the program starts and destroyed when the program stops.
Access modifiers cannot be used for local variables.	Access modifiers can be used forinstance variables.	Access modifiers can be used for class variables.
Local variables are visible only within the declared method, constructor or block.	The instance variables are visible for all methods, constructors and block in the class.	Visibility is similar to instance variables.
There is no default value for local variables so local variables should be declared and an initial value should be assigned before the first use.	Instance variables have default values. For numbers the default value is 0, for Booleans it is false and for object references it is null. Values can be assigned during the declaration or within the constructor.	Default values are same as instance variables.
Local variables can only be access inside the declared block.	Instance variables can be accesseddirectly by calling the variable name inside the class. However within static methods and different class should be called using the fully qualified name as follows: **ObjectReference.VariableName.**	Static variables can be accessed by calling with the class name. **ClassName.VariableName.**

Example program illustrating the use of all the above variables:

```
class area
{
int length=20;
int breadth=30;
```

```
static int classvar=2500;
void calc()
{
        int areas=length*breadth;
        System.out.println("The area is "+areas+" sq.cms");
}

public static void main(String args[])
{
   area a=new area();
   a.calc();
   System.out.println("Static Variable Value : "+classvar);
}
}
```

Output:

The area is 600 sq.cms
Static Variable Value : 2500

Program Explanation:

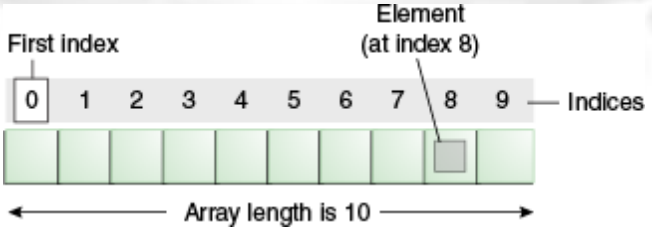
Class name: area
Method names: calc() and main()
Local variables: areas (accessed only in the particular method)
Instance variables: length and breadth (accessed only through the object's method)
Static variable: accessed anywhere in the program, without object reference

ARRAYS

Definition:

An array is a collection of similar type of elements which has contiguous memory location.

Java array is an object which contains elements of a similar data type.
Additionally, The elements of an array are stored in a contiguous memory location.
It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.
Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.



Advantage of Array:

- **Code Optimization:** It makes the code optimized; we can retrieve or sort the data easily.
- **Random access:** We can get any data located at any index position.

Disadvantage of Array:

- **Size Limit:** We can store only fixed size of elements in the array. It doesn't growits size at runtime.

Types of Array:

There are two types of array.

1. One-Dimensional Arrays
2. Multidimensional Arrays

1. One-Dimensional Array:

Definition: One-dimensional array is an array in which the elements are stored in onevariable name by using only one subscript.

- **Creating an array:**

Three steps to create an array:

1. Declaration of the array
2. Instantiation of the array
3. Initialization of arrays

1. Declaration of the array:

Declaration of array means the specification of array variable, data_type and array_name.

Syntax to Declare an Array in java:

```
dataType[]  arrayRefVar;   (or)
dataType  []arrayRefVar;   (or)
dataType arrayRefVar[];
```

Example:

int[] floppy; (or) int []floppy (or) int floppy[];

2. Instantiation of the array:

Definition:

Allocating memory spaces for the declared array in memory (RAM) is called as **Instantiation of an array.**

Syntax:

arrayRefVar=new datatype[size];

Example: floppy=new int[10];

3. Initialization of arrays:Definition:

Storing the values in the array element is called as **Initialization of arrays.**

Syntax to initialize values to array element:

arrayRefVar[index value]=constant or value;

Example:

floppy[0]=20;

SHORTHAND TO CREATE AN ARRAY OBJECT:

Java has shorthand to create an array object and supply initial values at the same timewhen it is created.

dataType[] arrayRefVar={list of values};
(or)
dataType []arrayRefVar={list of values};
(or)
dataType arrayRefVar[]={list of values};
(or)
dataType arrayRefVar[]=arrayVariable;

Example 1:

int regno[]={101,102,103,104,105,106};
int reg[]=regno;

Example 2: double[] myList = new double[10];

ARRAY LENGTH:

The variable **length** can identify the length of array in Java. To find the number of elements of an array, use **array.length.**

Example1:

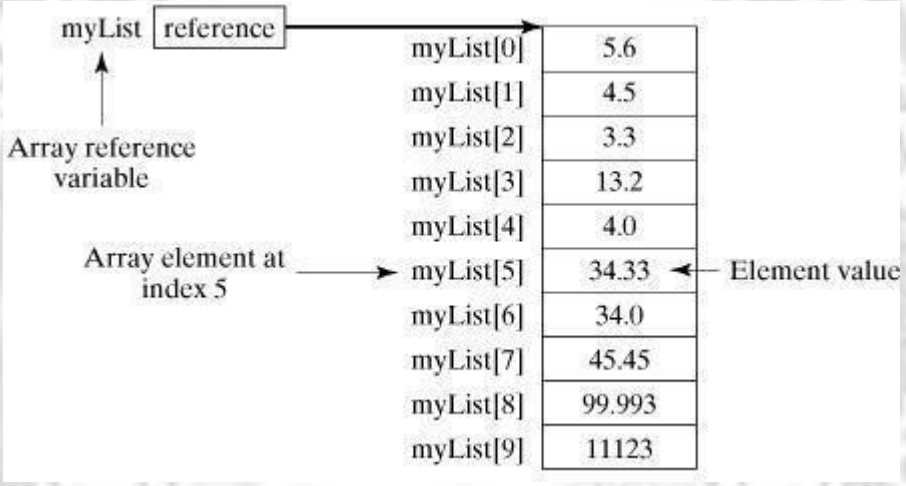
int regno[10]; len1=regno.length;

Example 2:

```
for(int i=0;i<reno.length;i++)
        System.out.println(regno[i]);
```

Following picture represents array myList. Here, myList holds ten double values and the indicesare from 0 to 9.

Example: (One-Dimensional Array)




```
class Array
{
public static void main(String[] args)
{
    int month_days[];
    month_days=new int[12];
    month_days[0]=31;
    month_days[1]=28;
    month_days[2]=31;
    month_days[3]=30;
    month_days[4]=31;
    month_days[5]=30;
    month_days[6]=31;
    month_days[7]=31;
    month_days[8]=30;
    month_days[9]=31;
    month_days[10]=30;
    month_days[11]=31;

    System.out.println("April has "+month_days[3]+ " days.");
}
}
```

Output:

April has 30 days.

Example 2: Finding **sum of the array elements** and **maximum** from the array:

```
public class TestArray
{
public static void main(String[] args)
{
        double[] myList = {1.9, 2.9, 3.4, 3.5};

        // Print all the array elements
        for (double element: myList)
        {
                System.out.println(element);
        }

        // Summing all elementsdouble total = 0;
        for (int i = 0; i < myList.length; i++)
        {
                total += myList[i];
        }
        System.out.println("Total is " + total);

        // Finding the largest elementdouble

        max = myList[0];
        for (int i = 1; i < myList.length; i++)
        {
                if (myList[i] > max)
                        max = myList[i];
        }
        System.out.println("Max is " + max);
    }
}
```

Output:

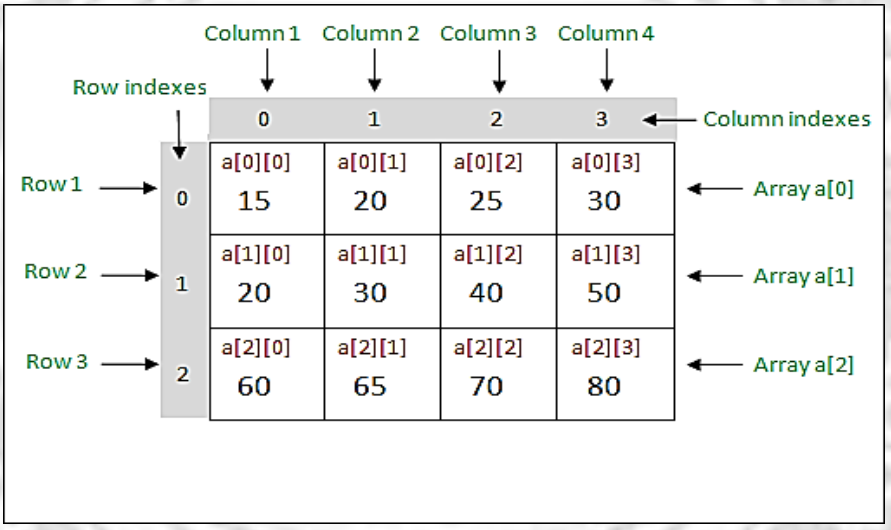
```
1.9
2.9
3.4
3.5
```

Total is 11.7
Max is 3.5

2. Multidimensional Arrays:

Definition:

Multidimensional arrays are ***arrays of arrays.*** It is an array which uses more than one index to access array elements. In multidimensional arrays, data is stored in row and column based index (also known as matrix form).



Uses of Multidimensional Arrays:

- ✓ Used for table
- ✓ Used for more complex arrangements

Syntax to Declare Multidimensional Array in java:

1. dataType[][] arrayRefVar; (or)
2. dataType [][]arrayRefVar; (or)
3. dataType arrayRefVar[][]; (or)
4. dataType []arrayRefVar[];

Example to instantiate Multidimensional Array in java:

```
int[][] arr=new int[3][3]; //3 row and 3 column - internally this matrix is implemented as arrays of arrays of int.
```

Example to initialize Multidimensional Array in java:

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
```

```
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;
```

Examples to declare, instantiate, initialize and print the 2Dimensional array:

```
class twoDarray
{
public static void main(String args[])
{
int array1[][]=new int[4][5];// declares an 2D array.
int array2[][]={{1,2,3},{2,4,5},{4,4,5}}; //declaring and initializing 2D arrayint i,j,k=0;

  // Storing and printing the values of Array1

System.out.println("-------Array 1 -------" );
for(i=0;i<4;i++)
{
     for(j=0;j<5;j++)
     {
               array1[i][j]=k;k++;
               System.out.print(array1[i][j]+ " ");
     }
     System.out.println();
}

// printing 2D array2
System.out.println("-------Array 2 -------");
for(int i=0;i<3;i++)
     {
     for(int j=0;j<3;j++)
     {
     System.out.print(array2[i][j]+
     }
     System.out.println();
     }
     }
}
```

Output:

----------Array1----------

0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19

----------Array2----------

1 2 3
2 4 5
4 4 5

In the above program, the statement **int array1[][]=new int[4][5];** is interpreted automatically as follows:
array1[0]=new int[5];array1[1]=new int[5];array1[2]=new int[5];array1[3]=new int[5];

It means that, when we allocate memory for a multidimensional array, we need to only specify the memory for the first (leftmost) dimension. We can allocate the remaining dimensions separately with different sizes.

Example: Manually allocate differing size second dimensions:

```
class twoDarray
{
public static void main(String args[])
{
int array1[][]=new int[4][];          // declares an 2D array.

array1[0]=new int[1];
array1[1]=new int[2];
array1[2]=new int[3];
array1[3]=new int[4];
int i,j,k=0;

  // Storing and printing the values of Array
for(i=0;i<4;i++)
{
     for(j=0;j<i+1;j++)
     {
              array1[i][j]=k;k++;
              System.out.print(array1[i][j]+ " ");
```

```
    }
    System.out.println();
}
}
}
```

Output:

```
0
1 2
3 4 5
6 7 8 9
```