# Data Types in Java

There are 8 primitive data types available in Java language as given below.

1. byte
2. short
3. int
4. long
5. float
6. double
7. char
8. boolean

All the above data type are divided into four groups as given below :-

- **Integer**

| Data Type | No. of bytes | Range |
|---|---|---|
| byte | 1 | -128 to 127 |
| short | 2 | -32768 to 32767 |
| int | 4 | -2147483648 to 2147483647 |
| long | 8 | -9223372036854775808 to 9223372036854775807 |

- **Floating**

| Data types | No. of bytes | Range |
|---|---|---|
| float | 4 | -3.4 e 38 to 3.4e 38 |
| double | 8 | -1.7e 308 to 1.7 e 308 |

- **Character**

| Data Type | No. of bytes | Range |
|---|---|---|
| char | 2 | 0 to 65535 |

char data type of java always follows the UNI ( Universal code ) range.

- **Boolean**

| Data Type | No. of bytes | Range |
|---|---|---|
| boolean | 1 | true or false |

## Default data types in Java

Default data types of Java of language to store Integer, floating and character constants are int, double and char respectively.

Default data types of Java to store any integer constant is always an int i.e. Java provides four bytes of memory to hold any integer constant and the range of int data type is various from –2147483648 to +2147483647.

```
class Example 1
{
   public static void main (String args[ ])
        {
        long n= 21474836479;
        System out println ("value="+n);
        }
}
```

In the above example, value 21474836479 is an integer constant during the compile time of program, compiler will check the range of default data type for the given constant value 21474836479. But the given constant value is not in the range of int data type therefore interpreter will not able to hold the constant value within the four bytes of memory of int data type and hence compiler will generate error as given below.

**Error : value 21474836479 is too long.**

To solve the above problem, it is necessary to increase the capacity of default data type from 4 bytes to 8 bytes by using suffix L as shown in the following example.

```
class Example1
{
        public static void main (String args[ ])
        {
            long n= 21474836479L;
            System. out.println("Value="+n);
         }
}
```

**Output** : Value = 21474836479

In the above Example character L (suffix L) represents long data type to hold constant integer values. During interpretation, interpreter will change its default data type from int to long for the execution of single statement.

Default data type of Java language to hold any float constant is always double i.e. for any floating constant value Java provides 8 bytes of memory.

```
class Example 2
{
        public static void main (String args[ ] )
        {
            float n = 1.2;
            System.out.println ("value="+n);
        }
}
```

In the above Example value 1.2 is a floating constant value but the default data type of Java to hold such value is double that is Java provides 8 bytes of memory to hold the above constant value but the variable is having only four bytes

of memory therefore during run-time of program Java Interpreter will not able to convert double value into float value hence above program will generate error. To solve the above problem, it is necessary to convert the double value into float value by using type casting as shown in the following example.

```
class Example 2
{
        public static void main (String args[ ])
        {
            float n= (float) 0.5;
            System.out.println ("value ="+n);
         }
}
```

**Output** : value = 0.5

```
class Example3
{
        public static void main (String args[ ])
        {
            float n = 0.5f;
            System.out.println ("value="+n);
        }
}
```

**Output**: value =0.5

# Type Casting

Forcefully conversion of one data type into another is known as type casting. Following are the possible reasons of generating errors related to typecasting.

**Syntax**

```
<target variable>=(target data type )<source variable>;
```

1. Number of bytes of target variable is less than the number of bytes of source variable.
2. Source and target variable are of different groups.
3. boolean cannot be type cast to other data type and vice versa.

```
class Example1
{
    public static void main (String args[ ])
        {
        short s;
        int  i = 12;
        s=i;
        System.out.println(s);
        System.out.println(i);
        }
}
```