Linkdin : sachinshetkar

L2_LangauageFundamentals & DataTypes

Reserved words, keywords Associated with Datatypes

Reserve Words:

In java some identifiers are reserved to associate some functionality or meaning such type of reserved identifiers are called as “reserve words”.

Total 53 Reserve Words in java:

Keywords(50) and Reserved Literals(3)(It means value)

1.Keywords :

(i)Used Keywords(48) =>

For Datatypes(8) : byte, short, int, long, float, double, char, boolean

For Control(11) : if, else, switch, case, default, for, while, do, break, continue, return.

For Identifiers(11) : private, public, protected, final, abstract, native, strictfp, synchronized, transiet, volatile.

For ExceptionHandling(6): try, catch, finally, throw, throws, assert(1.4V).

For Class Types(7) : class, package, import, extends, implements, interface,enum.

For object Types(4) : new, instanceof, super, this.

For ReturnType of Methods(1) : void .

(ii)Unused Keywords(2) =>goto, const.

2.Reserved Literals(3) : true, false, null.

Conclusions :

1.All reserve words in java contains only lower-case alphabets.

2.In java to create an object for class we have “new” keywords, but we

don’t have “delete” keyword to destroy the object whereas destroying object is taken care by a program(thread) called “GarbageCollector”.

Which of the following list contains only java reserve words:

a. final, finally, finalize(invalid)
b. throw, throws, thrown(invalid)
c. break, continue, return, exit(invalid)
d. goto, constant(invalid)
e. byte, short, Integer(invalid),long
f. extends, implements, imports(invalid)
g. finalized(invalid), synchronized
h. instanseof, sizeOf(invalid)
i. new, delete(invalid)
j. public, static, void , main(invalid), String(invalid), args(invalid)

Identifier: Name of class , method , variable , label.

Reserve Words : special meaning given for few identifiers(known to Compiler and JVM).

Datatypes :

Every variable has type, every expression has a type and all types are strictly defined moreover every assignment should be checked by the compiler for the type compatibility, so we say java is “Strictly Typed Programming Langauge”.

We should know which type of value we are storing in variable i.e where we require datatypes.

Primitive-DataTypes :

a. Numeric Type :
   - Integral Type (Whole Numbers) : byte, short, int, float.
   - Floating Type (real numbers) : float , double.

b. Character type: char ex : char gender = ‘M’;

c. Boolean type: boolean ex: boolean isPass = true;

1) byte :

byte data = 10;

MAX RANGE :: 127

MIN RANGE :: -128

Corresponding wrapper class is java.lang.Byte

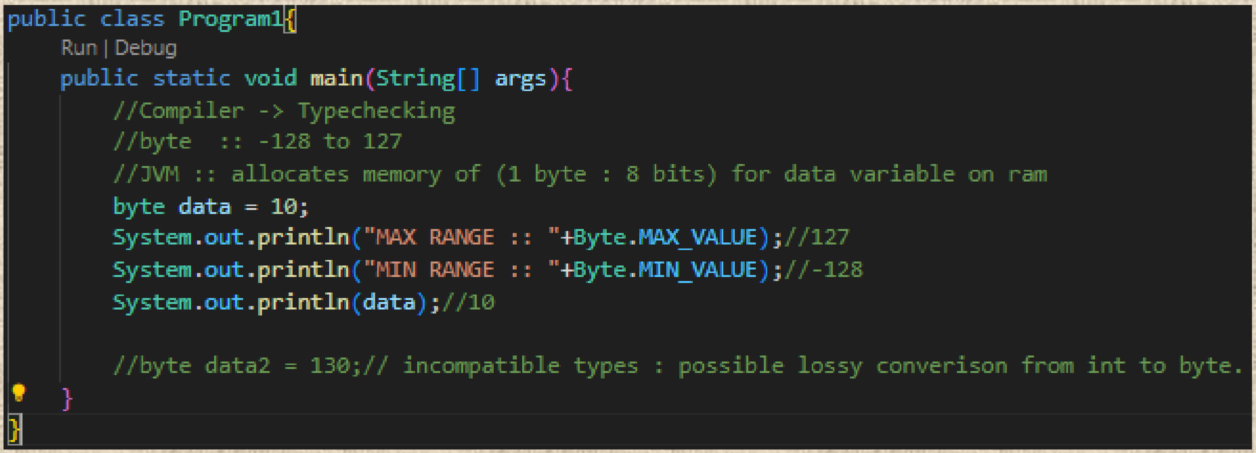
When to use byte datatype?

- ⇨ byte a datatype is suitable only when we work with handling data in terms of streams either from the file or from the network.

Note :: If the no is positive number then first bit will be zero

If the no is negative number, then first bit will be one.

Negative number will be stored in 2’s Compliment.



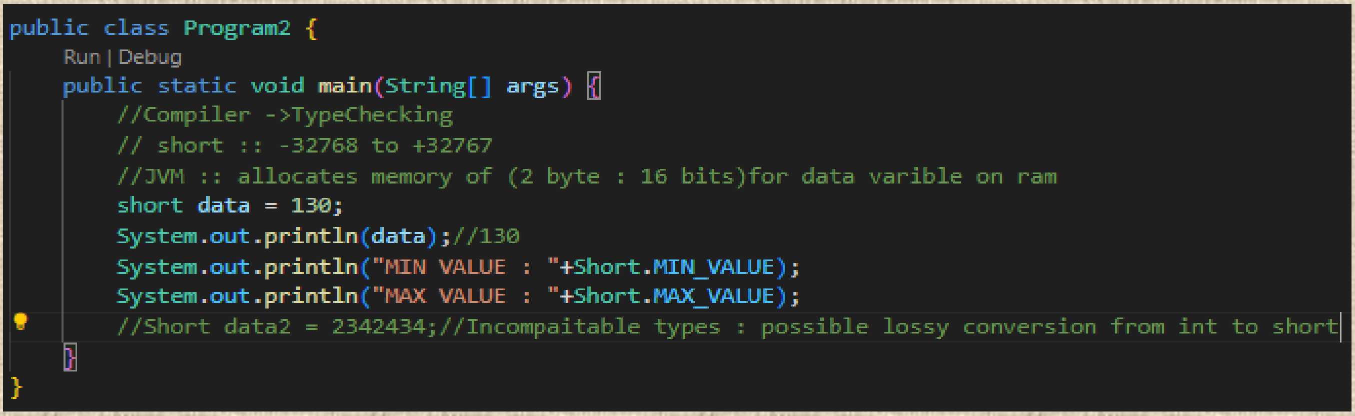
```
public class Program1{
    Run | Debug
    public static void main(String[] args){
        //Compiler -> Typechecking
        //byte  :: -128 to 127
        //JVM :: allocates memory of (1 byte : 8 bits) for data variable on ram
        byte data = 10;
        System.out.println("MAX RANGE :: "+Byte.MAX_VALUE);//127
        System.out.println("MIN RANGE :: "+Byte.MIN_VALUE);//-128
        System.out.println(data);//10

        //byte data2 = 130;// incompatible types : possible lossy converison from int to byte.
    }
}
```

byte data = ‘A’ ; // No Error

For storing value grater than byte range use short.

2.short (Range : -32768 to 32767) => 2 bytes (16 bits)

Corresponding wrapper class is java.lang.Short



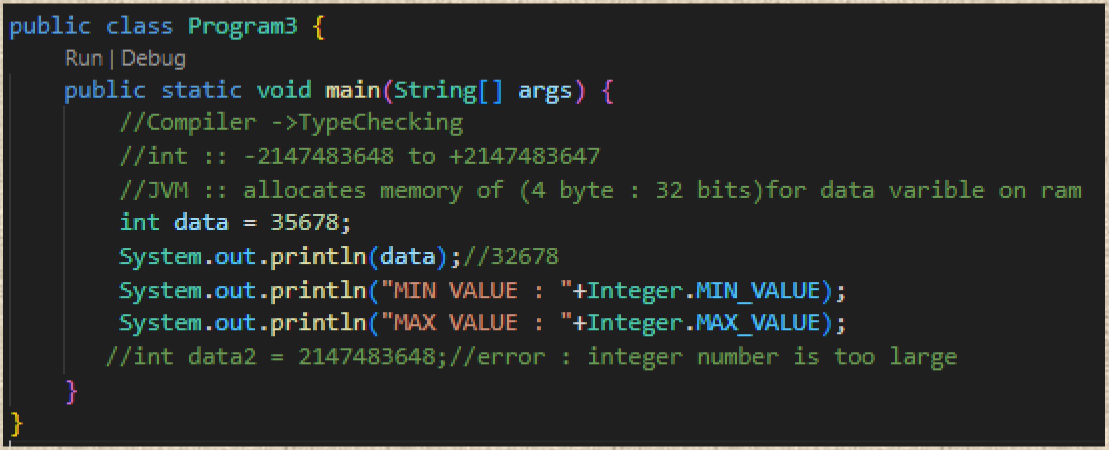
```
public class Program2 {
    Run | Debug
    public static void main(String[] args) {
        //Compiler ->TypeChecking
        // short :: -32768 to +32767
        //JVM :: allocates memory of (2 byte : 16 bits)for data varible on ram
        short data = 130;
        System.out.println(data);//130
        System.out.println("MIN VALUE : "+Short.MIN_VALUE);
        System.out.println("MAX VALUE : "+Short.MAX_VALUE);
        //Short data2 = 2342434;//Incompaitable types : possible lossy conversion from int to short
    }
}
```

If we want to store value greater than range of short use int datatype.

3.int (Range : -2147483648 to 2147483647)=> 4 bytes (32 bits)

=> Most commonly used numeric data type.

Corresponding wrapper class is java.lang.Integer



```
public class Program3 {
    Run | Debug
    public static void main(String[] args) {
        //Compiler ->TypeChecking
        //int :: -2147483648 to +2147483647
        //JVM :: allocates memory of (4 byte : 32 bits)for data varible on ram
        int data = 35678;
        System.out.println(data);//32678
        System.out.println("MIN VALUE : "+Integer.MIN_VALUE);
        System.out.println("MAX VALUE : "+Integer.MAX_VALUE);
       //int data2 = 2147483648;//error : integer number is too large
    }
}
```

If you want to store value greater than integer range use long datatype.

4.long(Range : -2^63 to 2^63-1) => 8 bytes (64 bits)

-whenever int is not enough to hold the value then we opt for long datatype.

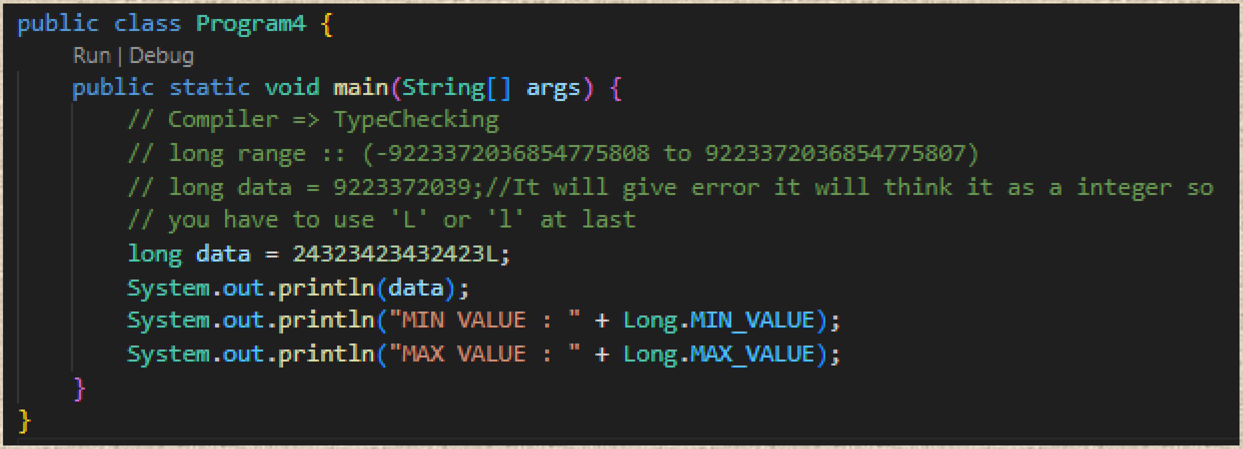
Corresponding wrapper class is java.lang.Long

When to use long datatype?

- ⇨ It is preferred when we work with file whose size is in terms of GB

Ex : long l = 35; long l = 35L; long l = 10l;

To specify the long type we can prefix it with ‘L’ or ‘l’.



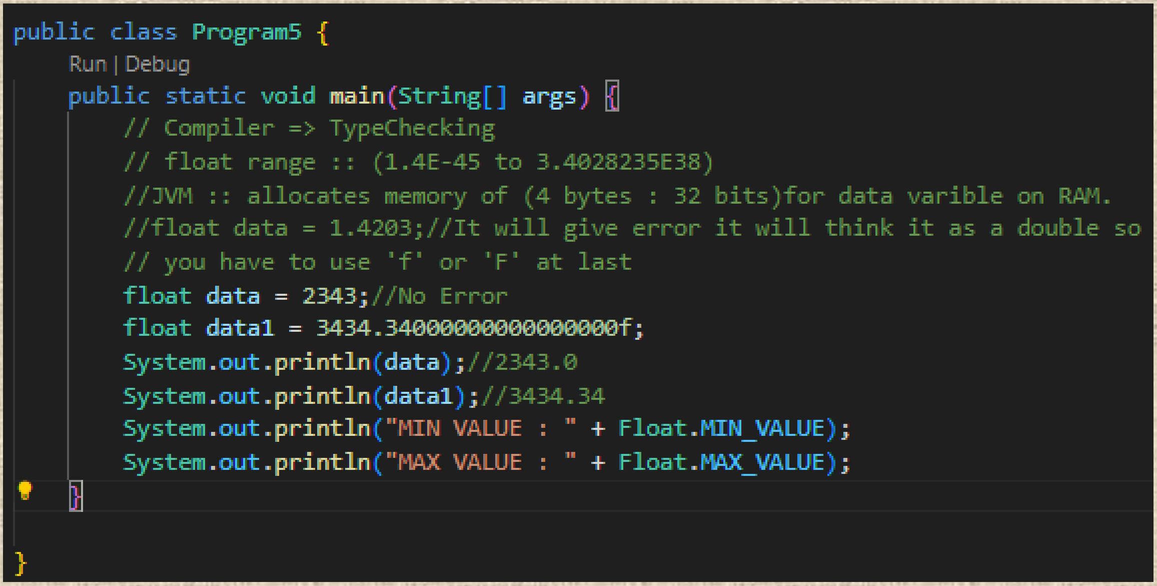
```
public class Program4 {
    Run | Debug
    public static void main(String[] args) {
        // Compiler => TypeChecking
        // long range :: (-9223372036854775808 to 9223372036854775807)
        // long data = 9223372039;//It will give error it will think it as a integer so
        // you have to use 'L' or 'l' at last
        long data = 24323423432423L;
        System.out.println(data);
        System.out.println("MIN VALUE : " + Long.MIN_VALUE);
        System.out.println("MAX VALUE : " + Long.MAX_VALUE);
    }
}
```

Now if we have to store floating values or real numbers we have to use float or double.

5.float (Range: 1.4E-45 to 3.4028235E38) => 4 bytes (32 bits)

Corresponding wrapper class is java.lang.Float

To specify the float type we can prefix it with ‘F’ or ‘f’ otherwise it will consider as double it will give error: Incompatible types : Possible lossy conversion.



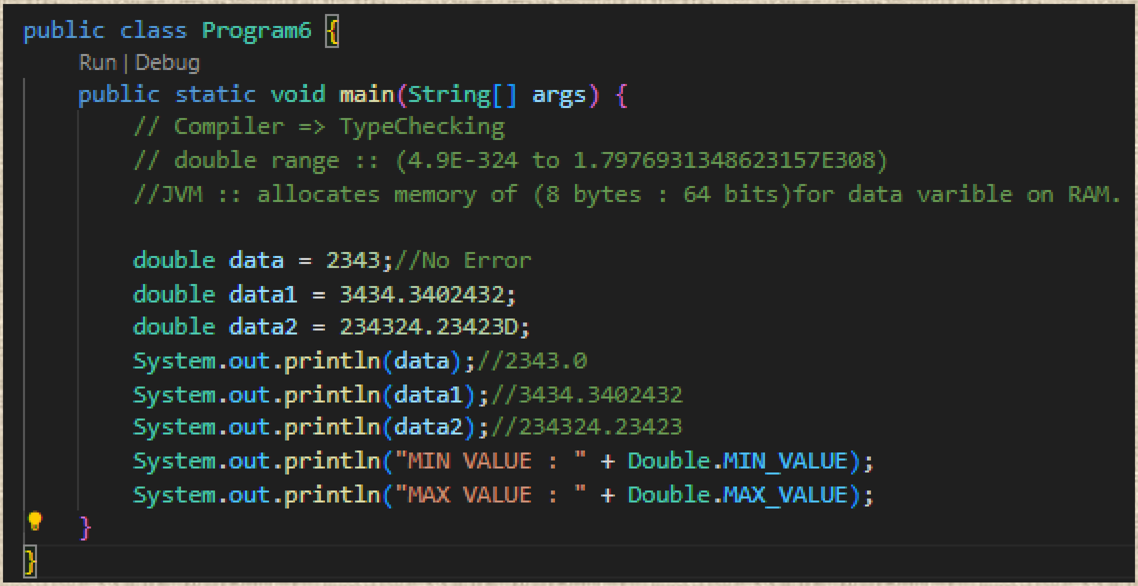
```
public class Program5 {
    Run | Debug
    public static void main(String[] args) {
        // Compiler => TypeChecking
        // float range :: (1.4E-45 to 3.4028235E38)
        //JVM :: allocates memory of (4 bytes : 32 bits)for data varible on RAM.
        //float data = 1.4203;//It will give error it will think it as a double so
        // you have to use 'f' or 'F' at last
        float data = 2343;//No Error
        float data1 = 3434.340000000000000000f;
        System.out.println(data);//2343.0
        System.out.println(data1);//3434.34
        System.out.println("MIN VALUE : " + Float.MIN_VALUE);
        System.out.println("MAX VALUE : " + Float.MAX_VALUE);
    }

}
```

If you want more accurate or precision value use double datatype.

6.double (Range : 4.9E-324 to 1.7976931348623157E308) => 8 bytes(64 bits)

Corresponding wrapper class is java.lang.Double

There is no need to explicitly to tell about by using prefix ‘l’ or ‘L’ for double because it is by default double . If you want you can use.



```
public class Program6 {
    Run | Debug
    public static void main(String[] args) {
        // Compiler => TypeChecking
        // double range :: (4.9E-324 to 1.7976931348623157E308)
        //JVM :: allocates memory of (8 bytes : 64 bits)for data varible on RAM.

        double data = 2343;//No Error
        double data1 = 3434.3402432;
        double data2 = 234324.23423D;
        System.out.println(data);//2343.0
        System.out.println(data1);//3434.3402432
        System.out.println(data2);//234324.23423
        System.out.println("MIN VALUE : " + Double.MIN_VALUE);
        System.out.println("MAX VALUE : " + Double.MAX_VALUE);
    }
}
```

For storing single character value use char datatype.

7.char (Range : space to ?)=> 2 bytes (16 bits)

Java will follow UNICODE not ASCII code.

Corresponding wrapper class is java.lang.Character

Ex : char ch = 'A';   char ch = '*';

8.boolean (1 byte but it will take only 1 bit) – stores true or false values.

Ex : boolean isPass = true;

==