

Data types in Java

Data types in Java specify how the values of the variable get stored in memory. Each variable has a data type that decides what type of value the variable will hold.

First, one is a Statically typed language where each variable and expression type is already known at compile time. Once a variable is declared to be of a certain data type, it cannot hold values of other data types. For example C, C++, Java.

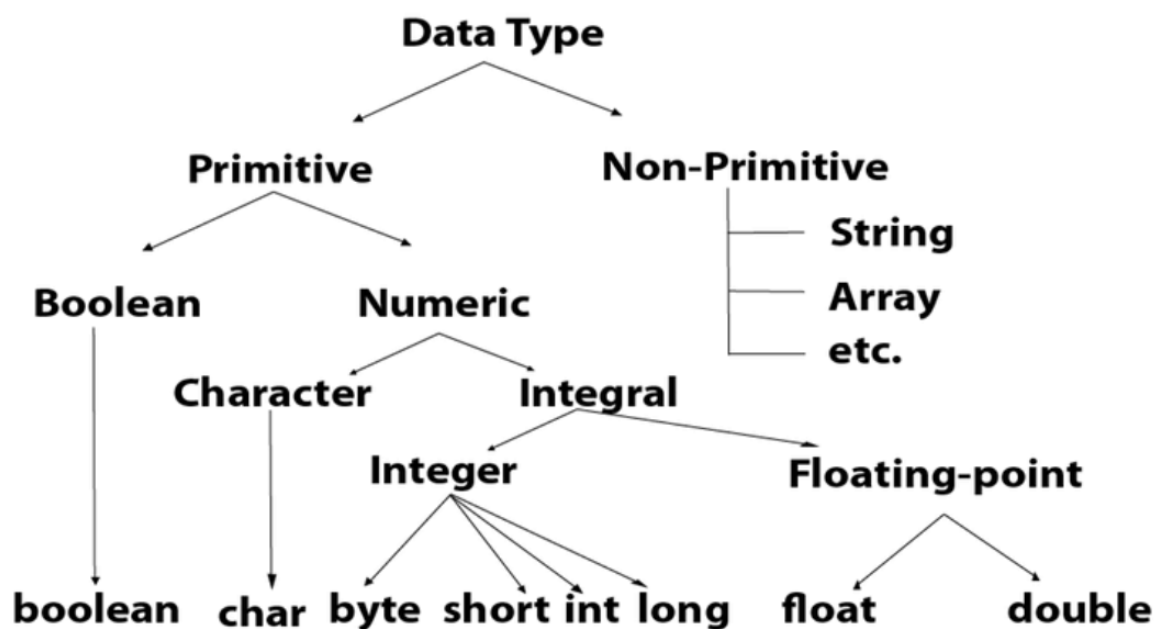
The other is Dynamically typed languages. These languages can receive different data types over time. For example Ruby, Python

Java is statically typed and also a strongly typed language because, in Java, each type of data (such as integer, character, hexadecimal, packed decimal, and so forth) is predefined as part of the programming language and all constants or variables defined for a given program must be described with one of the data types.

Java has two categories in which data types are segregated

Primitive Data Type: such as boolean, char, int, short, byte, long, float, and double

Non-Primitive Data Type or Object Data type: such as String, Array, etc



Types Of Primitive Data Types

Primitive data are only single values and have no special capabilities. There are 8 primitive data types. They are depicted below in tabular format below as follows:

Type 1: boolean

Boolean data type represents only one bit of information either true or false which is intended to represent the two truth values of logic and Boolean algebra, but the size of the boolean data type is virtual machine-dependent. Values of type boolean are not converted implicitly or explicitly (with casts) to any other type. But the programmer can easily write conversion code.

Syntax:

boolean booleanVar;

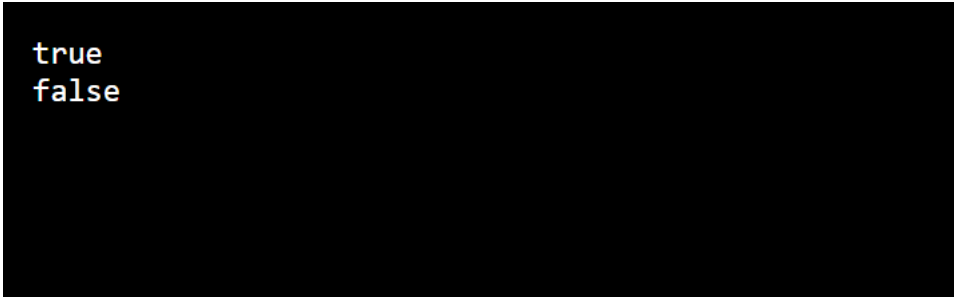
Size: Virtual machine dependent

Values: Boolean such as true, false

Default Value: false

Example:

```
public class Main {  
    public static void main(String[] args) {  
        boolean x = true;  
        boolean y = false;  
        System.out.println(x);  
        System.out.println(y);  
    }  
}
```

Result:

```
true  
false
```

Type 2: byte

The byte data type is an 8-bit signed two's complement integer. The byte data type is useful for saving memory in large arrays.

Syntax:

byte byteVar;

Size: 1 byte (8 bits)

Values: -128 to 127

Default Value: 0

Example 1:

```
public class Main {
    public static void main(String[] args) {
        byte myNum = 100;
        System.out.println(myNum);
    }
}
```

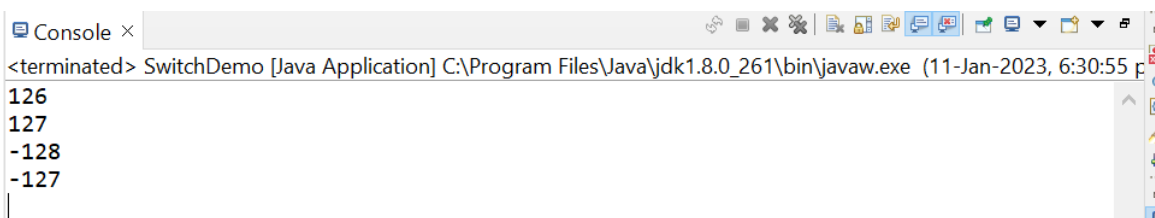


100

Example 2:

```
// Class
class App {
    // Main driver method
    public static void main(String args[]) {

        byte a = 126;
        // byte is 8 bit value
        System.out.println(a);
        a++;
        System.out.println(a);
        // It overflows here because
        // byte can hold values from -128 to 127
        a++;
        System.out.println(a);
        // Looping back within the range
        a++;
        System.out.println(a);
    }
}
```

Result:


```
Console x
<terminated> SwitchDemo [Java Application] C:\Program Files\Java\jdk1.8.0_261\bin\javaw.exe (11-Jan-2023, 6:30:55 p
126
127
-128
-127
```

Type 3: short

The short data type is a 16-bit signed two's complement integer. Similar to byte, use a short to save memory in large arrays, in situations where the memory savings actually matters.

Syntax:

short shortVar;

Size: 2 byte (16 bits)

Values: -32, 768 to 32, 767 (inclusive)

Default Value: 0

Example:

```
public class Main {  
    public static void main(String[] args) {  
        short myNum = 5000;  
        System.out.println(myNum);  
    }  
}
```

Result:

5000

Type 4: int

It is a 32-bit signed two's complement integer.

Syntax:

int intVar;

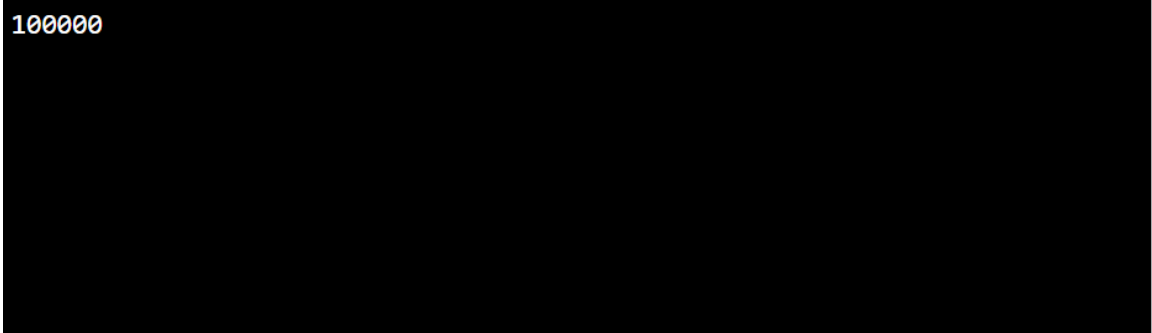
Size: 4 byte (32 bits)

Values: -2, 147, 483, 648 to 2, 147, 483, 647 (inclusive)

Note: The default value is '0'

Example:

```
public class Main {  
    public static void main(String[] args) {  
        int myNum = 100000;  
        System.out.println(myNum);  
    }  
}
```

Result:

100000

Type 5: long

The range of a long is quite large. The long data type is a 64-bit two's complement integer and is useful for those occasions where an int type is not large enough to hold the desired value.

Syntax:

long longVar;

Size: 8 byte (64 bits)

Values: {-9, 223, 372, 036, 854, 775, 808} to {9, 223, 372, 036, 854, 775, 807} (inclusive)

Note that you should end the value with an "L":

Example:

```
public class Main {  
    public static void main(String[] args) {  
        long myNum = 150000000000L;  
        System.out.println(myNum);  
    }  
}
```

Type 6: float

The float and double data types can store fractional numbers. Note that you should end the value with an "f" for float.

Syntax:

`float floatVar;`

Size: 4 byte (32 bits)

Values: upto 7 decimal digits

Note: The default value is '0.0'.

Example:

```
public class Main {
    public static void main(String[] args) {
        float myNum = 5.75f;
        System.out.println(myNum);
    }
}
```

Result:


5.75

Example 2:

```
3 public class App {
4     // Main driver method
5     public static void main(String[] args) {
6         // If we remove f ,we get compile time error
7         float value2 = 9.87;
8         System.out.println(value2);
9     }
10 }
11
```

Console ×

<terminated> App (14) [Java Application] C:\Program Files\Java\jdk1.8.0_261\bin\javaw.exe (11-Jan-2023, 6:44:05 pm – 6:44:05 pm) [pid: 6580]

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Type mismatch: cannot convert from double to float

at com.ashokit.corejava.decisionmaking.App.main(App.java:9)

Type 7: double

The double data type is a double-precision 64-bit IEEE 754 floating-point. For decimal values, this data type is generally the default choice.

Syntax:

double doubleVar;

Size: 8 bytes or 64 bits

Values: Upto 16 decimal digits

Example:

```
public class Main {  
    public static void main(String[] args) {  
        double myNum = 19.99d;  
        System.out.println(myNum);  
    }  
}
```

Result:

19.99

Note: Writing d is optional.

Use float or double?

The precision of a floating point value indicates how many digits the value can have after the decimal point.

The precision of float is only six or seven decimal digits, while double variables have a precision of about 15 digits. Therefore it is safer to use double for most calculations.

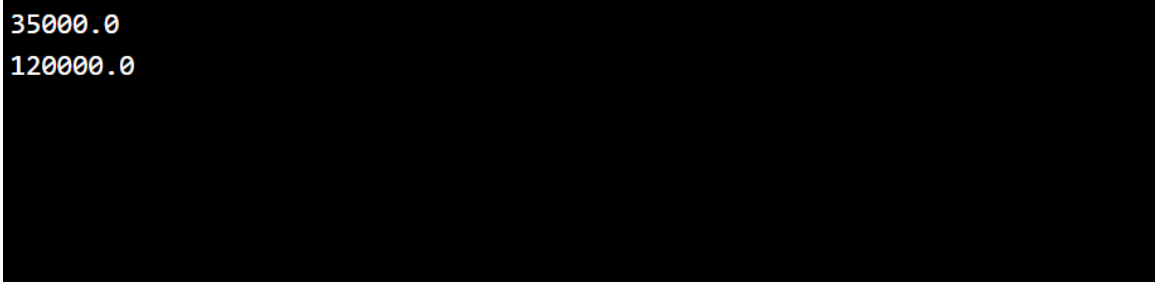
Scientific Numbers

A floating point number can also be a scientific number with an "e" to indicate the power of 10:

Example:

```
public class Main {  
    public static void main(String[] args) {  
        float f1 = 35e3f;  
        double d1 = 12E4d;  
        System.out.println(f1);  
        System.out.println(d1);  
    }  
}
```

Result:



```
35000.0  
120000.0
```

Type 8: char

The char data type is used to store a single character. The character must be surrounded by single quotes, like 'A' or 'c':

Syntax:

char charVar;

Size: 2 byte (16 bits)

Values: '\u0000' (0) to '\uffff' (65535)

Example:

```
public class Main {  
    public static void main(String[] args) {  
        char myGrade = 'B';  
        System.out.println(myGrade);  
    }  
}
```

Result:

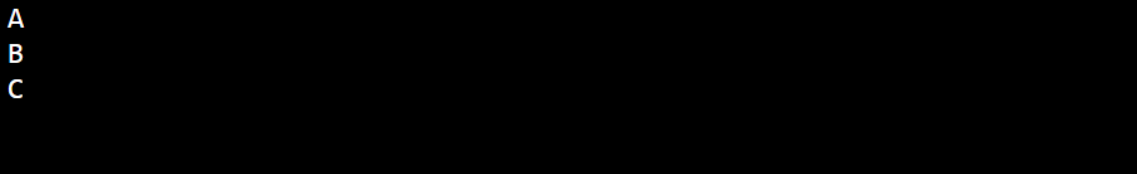
B

Alternatively, if you are familiar with ASCII values, you can use those to display certain characters:

Example:

```
public class Main {  
    public static void main(String[] args) {  
        char myVar1 = 65, myVar2 = 66, myVar3 = 67;  
        System.out.println(myVar1);  
        System.out.println(myVar2);  
        System.out.println(myVar3);  
    }  
}
```

Result:

A
B
C

You must be wondering why is the size of char 2 bytes in Java?

So, in other languages like C/C++ uses only ASCII characters, and to represent all ASCII characters 8-bits is enough.

But java uses the Unicode system not the ASCII code system and to represent the Unicode system 8 bits is not enough to represent all characters so java uses 2 bytes for characters. Unicode defines a fully international character set that can represent most of the world's written languages. It is a unification of dozens of character sets, such as Latin, Greeks, Cyrillic, Katakana, Arabic, and many more.

Example:

```
public class App {  
    public static void main(String args[])  
    {  
        char a = 'G';  
        int i = 89;  
        byte b = 4;  
        short s = 56;  
        double d = 4.355453532;  
        // for float use 'f' as suffix as standard  
        float f = 4.7333434f;  
        long l = 12121L;  
        System.out.println("char: " + a);  
        System.out.println("integer: " + i);  
        System.out.println("byte: " + b);  
        System.out.println("short: " + s);  
        System.out.println("float: " + f);  
        System.out.println("double: " + d);  
        System.out.println("long: " + l);  
    }  
}
```

Result:

```
char: G  
integer: 89  
byte: 4  
short: 56  
float: 4.7333436  
double: 4.355453532  
long: 12121
```