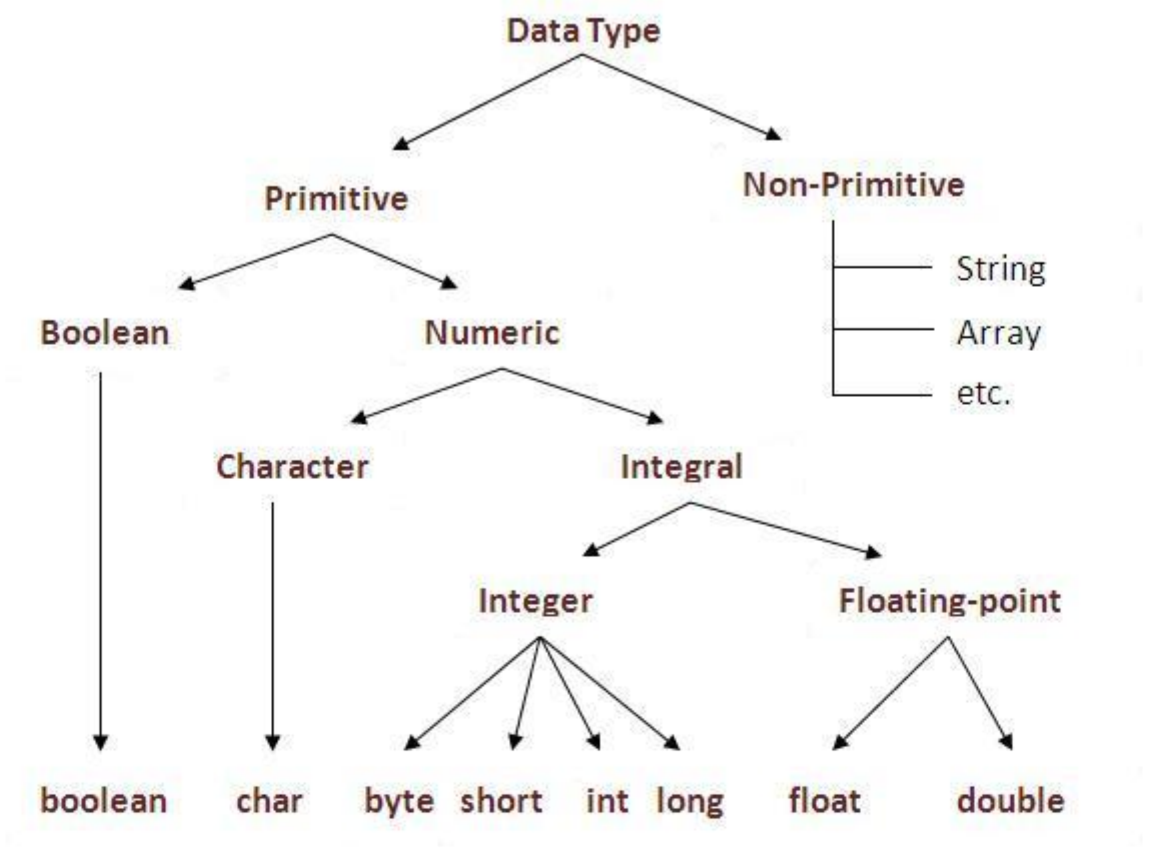
Data Types in Java:

In java, there are two types of data types

- primitive data types
- non-primitive data types



Primitive datatypes:

There are eight primitive data types in Java. These are as follows:

1. Byte: A byte, is one of the most basic units of memory made up of 8 individual bits. Byte data types in Java have the following characteristics:

- **Minimum Value:** -128 (2^7)
- **Maximum Value:** 127 (2^7-1)
- **Default Value:** 0

Examples:

byte x = 56

byte y = 68

Thus, you can save numbers between -128 and 127 (inclusive) in a byte. Bytes, because of their size, are useful for storing small data in large arrays. Another programmer looking through your code will also instantly recognize that a byte type will hold only a small value, thus improving your code's readability (a major issue for large applications).

.

2. Short: A short is twice the size of a byte, i.e. it is made up of 16-bits. Its chief characteristics are:

- **Minimum Value:** -32,768 (2^15)
- **Maximum Value:** 32,767 (2^15-1)
- **Default Value:** 0

Examples:

short x = 9870

short y = -635

Like bytes, short types are useful alternatives to int (see below) data types, particularly if your data falls within the specified range. As with byte, using short also improves code readability, besides saving memory.

3. Int: An integer is four times the size of a byte (i.e. it is made up of 32 bits). It is one of the most commonly used data types in Java.

- **Minimum Value:** -2,147,483,648 (2^31)
- **Maximum Value:** 2,147,483,647 (2^31 – 1)
- **Default Value:** 0

<u>**Examples:**</u>

int x = 150000

int y = -2004320

As the most easily understood data type, you will use int a lot in your code.

4. Long: A long data type is twice the size of an integer, i.e. it is made up of 64-bits. It's chief characteristics are:

- **Minimum Value:** -9,223,372,036,854,775,808 (2^63)
- **Maximum Value:** 9,223,372,036,854,775,807 (2^63 – 1)
- **Default Value:** 0

<u>**Examples:**</u>

long x = 6778005876543

long y = -554233254242

You'll use long only if you encounter data that doesn't fit within the int range (which will be rare).

5. Float: In programming, any decimal or fractional value is called a 'float'. If there is a decimal after the number, it will be classified as a float. In Java, a float is made up of 32-bits IEEE floating points*.

The minimum/maximum value of float is not the same as that of the int data type (despite both being made of 32-bits). The full range of float values is beyond the scope of this tutorial. For now, the only thing you need to know is that you'll use float (and double – see below) for saving decimal values.

<u>**Examples**</u>:

float x = 2.321

float y = 1.234

*The float value range depends on the IEEE standard classification for floating point numbers. You can read about it here.

6. Double: Double is a data type that is twice the size of a float. I.e. it is made up of 64-bit IEEE floating points.

double is a much more precise type than float. For all practical purposes, it is recommended that you use double instead of float for storing decimal values.

<u>**Examples:**</u>

double a = 1.245240

double y = 12.2232

7. Char: Char data type refers to a single 16-bit Unicode character. Unicode is a computer industry standard for representing text related data. This includes alphabets, symbols ($, &, *, #, @, !, etc.), and special figures such as ¢, £, ¥, etc. The Unicode character set includes over 110,000 characters covering more than 100 language scripts.

In other words, any data besides numbers goes into the char data type.

Examples:

char name = ‘John’

char country = ‘USA’

8. Boolean: Boolean is the smallest data type in Java, i.e. it is made up of only one bit. Thus, a Boolean data type can have only two values – 0 (or False) and 1 (or True).

Example:

boolean x = true

boolean y = false

(**Tip**: ‘True’ and ‘False’ written above are **not** strings. Do not enclose them within quotes as we did with the char example above)

Data Type	**Default Value**	**Default size**
Boolean	False	1 bit
Char	'\u0000'	2 byte
Byte	0	1 byte
Short	0	2 byte
Int	0	4 byte
Long	0L	8 byte
Float	0.0f	4 byte
Double	0.0d	8 byte

Why char uses 2 byte in java and what is \u0000 ?

Because java uses unicode system rather than ASCII code system. \u0000 is the lowest range of unicode system. To get detail about Unicode see below.

Non-Primitive Data Types:

Non-primitive data types are not defined by the programming language, but are instead created by the programmer. They are sometimes called "**reference variables**" or "**object references**"; since they reference a memory location, which stores the data. In the Java programming language, non-primitive data types are simply called "objects"; because they are created, rather than predefined. While an object may contain any type of data, the information referenced by the object may still be stored as a primitive data type.

Non-primitives are "Reference-types". These include class-based types, either core library defined or user defined, or array types (no class definition). The value assigned to variables of reference-type will either be a reference to an object (instance of a class) or null. A null value, meaning it does not currently refer to any instance in memory, is the initialization default for instance and class variables.

A reference value is a key value used to look up the actual object in the memory heap (i.e. used to map to an actual pointer maintained by the JVM object-management internals). You need merely think of the reference value as being a handle of sorts to the actual object. It "points" to an object or else it is null.

Variables of non-primitive datatypes or object references stores reference to the actual value stored in the primitive variable.

Objects (Classes & Interfaces) and Arrays are the reference or non-primitive data types in Java. They are so called because they are handled "by reference" i.e. variables of their type store the address of the object or array is stored in a variable. They are passed by reference. For Ex:

```
char[]  arr = { 'a', 'b', 'c', 'd' };

//'arr' stores the references for the 4 values
```

it is null.

Variables of non-primitive datatypes or object references stores reference to the actual value stored in the primitive variable.

The non-primitive data types in Java are objects and arrays. These non-primitive types are often called "reference types" because they are handled "by reference"--in other words, the address of the object or array is stored in a variable, passed to methods, and so on. By comparison, primitive types are handled "by value"--the actual primitive values are stored in variables and passed to methods.

Literals:

By literal we mean any number, text, or other information that represents a value. This means what you type is what you get. We will use literals in addition to variables in Java statement. While writing a source code as a character sequence, we can specify any value as a literal such as an integer. This character sequence will specify the syntax based on the value's type. This will give a literal as a result. For instance

Integer Literals:

Integer literals is a sequence of digits and a suffix as L. To represent the type as long integer we use L as a suffix. We can specify the integers either in decimal, hexadecimal or octal format. To indicate a decimal format put the left most digit as nonzero

Character Literals:

We can specify a character literal as a single printable character in a pair of single quote characters such as 'a', '#', and '3'. You must be knowing about the ASCII character set. The ASCII character set includes 128 characters including letters, numerals, punctuations etc. There are few character literals which are not readily printable through a keyboard.

Boolean Literals:

The values true and false are also treated as literals in Java programming. When we assign a value to a boolean variable, we can only use these two values. Unlike C, we can't presume that the value of 1 is equivalent to true and 0 is equivalent to false in Java. We have to use the values true and false to represent a Boolean value. Like

boolean chosen = true;

Remember that the literal true is not represented by the quotation marks around it. The Java compiler will take it as a string of characters, if it’s in quotation marks.

Floating-point literals:

Floating-point numbers are like real numbers in mathematics, for example, 4.13179, -0.000001. Java has two kinds of floating-point numbers: float and double. The default type when you write a floating-point literal is double.

A floating-point literal can be denoted as a decimal point, a fraction part, an exponent (represented by E or e) and as an integer. We also add a suffix to the floating point literal as D, d, F or f. The type of a floating-point literal defaults to double-precision floating-point.

String Literals:

The string of characters is represented as String literals in Java. In Java a string is not a basic data type, rather it is an object. These strings are not stored in arrays as in C language. There are few methods provided in Java to combine strings, modify strings and to know whether to strings have the same value.

We represent string literals as

String myString = "How are you?";

The above example shows how to represent a string. It consists of a series of characters inside double quotation marks.

Null Literals:

The final literal that we can use in Java programming is a Null literal. We specify the Null literal in the source code as 'null'. To reduce the number of references to an object, use null literal. The type of the null literal is always null. We typically assign null literals to object reference variables. For instance

```
s = null;
```

This example an object is referenced by s. We reduce the number of references to an object by assigning null to s. Now, as in this example the object is no longer referenced so it will be available for the garbage collection i.e. the compiler will destroy it and the free memory will be allocated to the other object. Well, we will later learn about garbage collection