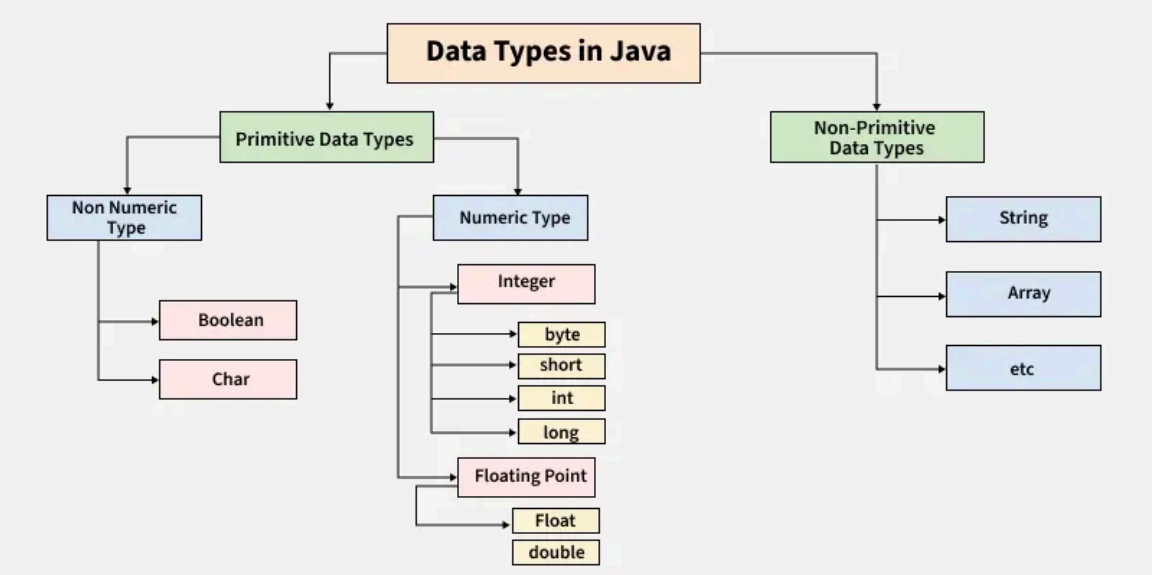
Data Types



Java has two categories in which data types are segregated.

1. Primitive Data Type

Predefined data types that are supported by a programming language and cannot be broken down into further simpler types are known as Primitive data types.

We have eight Primitive data types namely;-

- boolean
- char
- byte
- short
- int
- long

- float
- double

1. Boolean

- The boolean data type allows the programmer to store only two values- True or False.
- Hence it represents only one bit of information but the size of this data type is implementation dependent.
- It is often used as simple flag that is used to determine a True / False situation.

```
public class Main
{
    public static void main(String[] args) {
        boolean a = true;
        if (a == true)
            System.out.println("Hey There Prepster");
    }
}
```

2. Char

- The char data type is typically used to store a single Unicode character, and it is often represented as 16 bits (2 bytes) in many programming languages.
- Its value range spans from ‘\u0000’ (which is equivalent to 0) to ‘\uffff’ (equivalent to 65,535), inclusive.
- This data type is primarily employed for character storage and manipulation in various programming contexts.

```
public class CharExample {
    public static void main(String[] args) {
```

```
        char myChar = 'A';
        System.out.println("The value of myChar is: " + myChar);
    }
}
```

3. Byte

- The Byte data type is an 8-bit signed two’s complement integer.
- This can be used instead of int or other integer types to save memory when we are certain that the value will be within -128 and 127, since byte is 4 times smaller than the int type.
- A special feature of this data type is that it is *cyclic* in nature.

```
class Main {
    public static void main(String[] args) {
    byte num;

        num = 126;

        num++;

        System.out.println(num);

        num++;

        System.out.println(num);

    }
}
```

4. Short

- The short data type is a 16-bit signed two’s complement integer.
- Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

- The short data type can also be used to save memory in large arrays , just like byte data type. A short data type is 2 times smaller than an integer.

```
public class Main
{
   public static void main(String[] args) {
      short num = 10000;
      System.out.println(num);
   }
}
```

5. Int

- The *int* data type is the preferred data type when we create variables with a numeric value.

```
public class Main {
   public static void main(String[] args) {
      int myInt = 42; // Declare an int variable and assign a value to it
      System.out.println("The value of myInt is: " + myInt); // Print the value
of the int variable
   }
}

}
```

6. Long

- The long data type is a 64-bit integer that employs a two’s complement representation.
- Its value range extends from -9,223,372,036,854,775,808 (which is equivalent to -2^63) to 9,223,372,036,854,775,807 (which is equivalent to 2^63 – 1), inclusive.

- The minimum value it can hold is -9,223,372,036,854,775,808, while the maximum value is 9,223,372,036,854,775,807.
- By default, a long variable is initialized to 0. This data type is utilized when a wider range of values is required compared to the int data type

```
public class LongExample {
public static void main(String[] args) {
// Define two long integers and calculate their sum
long num1 = 123456789012345L;
long num2 = 987654321098765L;
long sum = num1 + num2;

// Display the result
System.out.println("Sum of " + num1 + " and " + num2 + " is: " + sum);
}
}
```

7. Float

- The float data type is a 32-bit IEEE 754 single-precision floating-point representation.
- It does not have a limited value range, which means it can represent a wide range of values, including very large and very small numbers.
- It is advisable to use float instead of double when conserving memory is essential, especially in situations involving large arrays of floating-point numbers.

```
float myFloat = 3.14f;
System.out.println(myFloat);
```

8. Double

- The double data type is a 64-bit IEEE 754 double-precision floating-point format.
- It possesses an extensive value range, accommodating a wide variety of numeric values, including those of substantial magnitude or extreme precision. Similar to the float data type, double is commonly used for decimal values.
- However, it’s crucial to note that for applications requiring high precision, such as currency calculations, it is not recommended to use the double data type due to its inherent limitations.

```
public class DoubleExample {
    public static void main(String[] args) {
        // Define two double numbers and calculate their sum
        double num1 = 3.14159265359;
        double num2 = 2.71828182846;
        double sum = num1 + num2;

        // Display the result
        System.out.println("Sum of " + num1 + " and " + num2 + " is: " + sum);
    }
}
```

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

Ques: Why Data Types Matter in Java?

Data types matter in Java because of the following reasons, which are listed below:

- **Memory Efficiency:** Choosing the right type (byte vs int) saves memory.
- **Performance:** Proper types reduce runtime errors.
- **Code Clarity:** Explicit typing makes code more readable.

Ques: Why is the Size of char 2 bytes in Java?

Unlike languages such as C or C++ that use the **ASCII character** set, Java uses the Unicode character set to support internationalization. Unicode requires more than 8 bits to represent a wide range of characters from different languages, including Latin, Greek, Cyrillic, Chinese, Arabic, and more. As a result, Java uses 2 bytes to store a char, ensuring it can represent any **Unicode** character.

2. Non-Primitive (Reference) Data Types

The **Non-Primitive (Reference) Data Types** will contain a memory address of variable values because the reference types won't store the variable value directly in memory. They are strings, objects, arrays, etc.

1. Strings

2. Class

3. Object

4. Interface

5. Array

Primitive vs Non-Primitive Data Types

The table below demonstrates the difference between Primitive and Non-Primitive Data types

Aspect	Primitive	Non-Primitive
Memory	Stored on the stack	Stored on the heap
Speed	Primitive data types are faster	Non-primitive data types are slower

Aspect	Primitive	Non-Primitive
Example	int x = 5;	String s = "Hello";