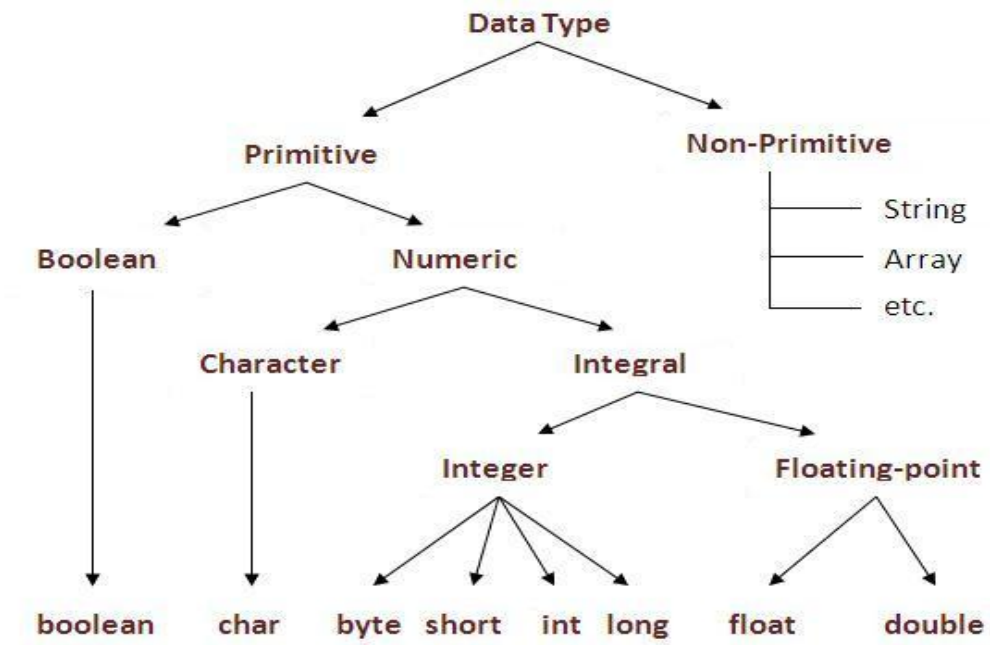
Data Types and Variables

What are DataTypes?

- Data-types means containers.
- They specify what kind of data to hold
- In java, there are two types of data types
  - **Primitive data types**
  - **Non-primitive data types**



Primitive Data Types

Sl. No.	Type	Size	Description
	Integers		
1	Byte	8 bits	Byte length integer
2	Short	16 bits	Short Integer
3	Int	32 bits	Integer
4	Long	64 bits	Long Integer
	Real Numbers		
5	Float	32 bits	Single Precision floating point
6	Double	64 bits	Double Precision floating point
	Other Types		
7	char	16 bit Unicode character	A single character
8	Boolean	True or false	A Boolean value

Non Primitive Data Types

- Strings
- Arrays
- Class

Variables

- Variable is name of reserved area allocated in memory.
- There are three types of variables in java
  - **local variable**
    - A variable that is declared inside the method is called local variable.
  - **instance variable**
    - A variable that is declared inside the class but outside the method is called instance variable . It is not declared as static.
  - **static variable**
    - A variable that is declared as static is called static variable. It cannot be local.

Variables

Syntax:

Datatype variable_name = value ;

Example

```
int id = 34;
float mks=67.34;

int int =45;    //invalid
```

Literals

- Literals are numbers, characters, and string representations used in a program.

- **Numeric Literals**

  int roll_no=45;

- **Character literals**

  char letter = ‘b';

  letter = ‘S';

- **String literals**

  String message = "Hello\tWorld\n";

Casting

- Casting is required when there is a need to explicitly convert a value from one type to another.
- The general syntax for a cast is:

```
(result_type)
value;
```

- Examples

```
float price = 37.53;
int dollars = (int) price;        //
fractional portion lost
                 // dollars = 37

char response = 'A';
byte temp =(byte) response;   // temp = 65
(ASCII value
                 // for 'a')
```

Type Casting

- Assigning a value of one type to a variable of another type is known as **Type Casting**.
- In Java, type casting is classified into two types,
  - Widening Casting(Implicit)
  - byte → short → int → long → float → double

    widening

    double → float → long → int → short → byte

    Narrowing

Widening Conversions

- A widening conversion occurs when a value stored in a smaller space is converted to a type of a larger space.
  - There will never be a loss of information

- Widening conversions occur automatically when needed.

From	To
byte	short, int, long, float, or double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

Narrowing Conversions

- A narrowing conversion occurs when a value stored in a larger space is converted to a type of a smaller space.
  - Information may be lost
  - Never occurs automatically. Must be explicitly requested by the programmer using a cast.

From	To
`byte`	`char`
`short`	`byte, char`
`char`	`byte, short`
`int`	`byte, short, char`
`long`	`byte, short, char, int`
`float`	`byte, short, char, int, long`
`double`	`byte, short, char, int, long, float`

Strings

- String is a sequence of characters. In the Java programming language, strings are objects. There are two types of String i.e
- 1. Mutable
- 2.Immutable
- In java, **string objects are immutable**.
- **Immutable** simply means unmodifiable or unchangeable. It is immutable because object reference is created.

String Formatting

Advantage of String class: many built-in methods for String manipulation

```
str.length();          // get length of string
str.toLowerCase()      // convert to lower case
str.toUpperCase()      // convert to upper case
str.charAt(i)          // what is at character i?
str.contains(..)       // String contains another string?
str.startsWith(..)     // String starts with some prefix?
str.indexOf(..)        // what is the position of a character?
```

StringBuffer Class

- Java StringBuffer class is used to created mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

```
class StringDemo{
    public static void main(String args[]){
     StringBuffer sb=new StringBuffer("Hello ");
     sb.append("Java");//now original string is changed
     System.out.println(sb);//prints Hello Java
     }
}
```

StringBuilder Class

- Java StringBuilder class is used to create mutable (modifiable) string.
- The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.

```
class A{
    public static void main(String args[]){
     StringBuilder sb=new StringBuilder("Hello ");
     sb.append("Java");//now original string is changed
     System.out.println(sb);//prints Hello Java
     }
}
```

Difference between String and StringBuffer

String

- String class is immutable
- String is slow and consumes more memory
- String class overrides the equals() method of Object class.

StringBuffer

- StringBuffer class is mutable
- StringBuffer is fast and consumes less memory
- StringBuffer class doesn't override the equals() method of Object class

Difference between String and StringBuffer conti..

StringBuffer

- StringBuffer is *synchronized*
- StringBuffer is *less efficient* than StringBuilder.

StringBuilder

- StringBuilder is *non-synchronized*
- StringBuilder is *more efficient* than StringBuffer.

Arrays

- An *array* is an ordered list of values

Each value has a numeric *index*

The entire array has a single name

	0	1	2	3	4	5	6	7	8	9
`rollno`	79 91	87	94	82	67	98	87	81	74	

An array of size N is indexed from zero to N-1

This array holds 10 values that are indexed from 0 to 9

Array - Syntax

- A particular value in an array is referenced using the array name followed by the index in brackets.
- *Datatype* [ ] variable_name = new datatype[size];
- Ex: int [] scores = new int [10];
- In JAVA, int is of 4 bytes, total space=4*10=40 bytes.
- Space Allocation is based on the datatype used to declare array.

Types of Array

- Single Dimensional Array (1 D Array)
- Two Dimensional Array (2 D Array)
- Multidimensional Array(Arrays of Arrays)

Dot Operator

- Dot operator is used to access methods and variables within classes.

- It is represented as . (dot)

Example

```
class Test
{
    public int a;
     public display()
{
    System.out.println("Hello");
    }
}
    public static void main(String args[])
  {
        Test t=new Test();
         t.a=32;       //accessing dot
         t.display();     //accessing display
     }
}
```