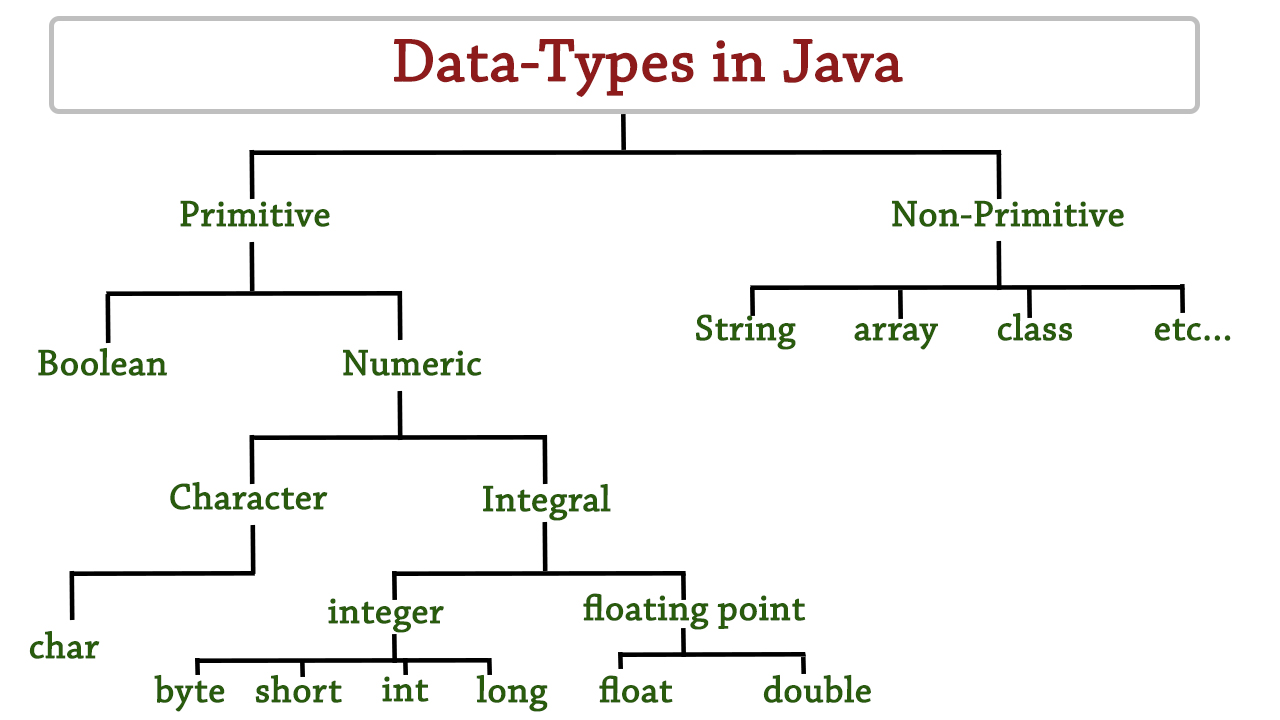
**Technical Documentation Module 1**

## **M01 L01**

* **Data Types**
  + A data type tells the compiler what kind of data is being stored in a variable and in memory
    - Must declare (assign) the type of each variable when you create it
    - Strongly Typed Language
    - In *numeric type conversions*, java will convert the smaller type to a larger type before operation occurs.

**Example**



* + **Type Casting**
    - ***Widening (Implicit Casting)-*** *Converting a value with a smaller range to one with a bigger range, Java does automatically*
    - ***Narrowing (Explicit Casting)-*** *Converting a value with a bigger range to one with a smaller range, must cast it explicitly so you don't lose data*
* **Class**
  + A blueprint that allows you to define and construct one or more objects with properties and behaviors
    - This does not set aside any memory
* **Object**
  + An instance of a class that contains its properties and behaviors
    - ***Properties-*** *a data fields (attributes, variables) that can define an object*
    - ***Behaviors-*** *the behavior (method) that operates and grants actions to the object so it can do things.*
      * ***Getter-*** *A method that returns the value of an instance variable*
      * ***Setter-*** *A method that sets the value of an instance variable*
    - **Instantiation-** an abbreviation for an instance of a class
* **Constructors**
  + It’s a special method used to create an object and initialize the state of the object (instance variables) when created within a class
    - The constructors are optional depending on what kind of code you are making but will automatically be made by java as a default constructor
      * It has the same name as the class it’s written in
      * It can be overloaded
      * Has no return type
      * It’s invoked by the new operator
* **Instance**
  + ***Instance Variable-*** *A variable that belongs to a specific object and or instance of a class, usually declared with ‘private’ for encapsulation.*
  + ***Instance Method-*** *A method that operates on an object's instance variables and can access instance variables.*
    - *Recommended use is when each object of the class needs an independent copy of the variable*
* **Static**
  + ***Static Variable (Class Variable)-*** *A variable shared by all objects of a class declared using the ‘static’ keyword.*
  + ***Static Method-*** *A method shared by all objects of a class and is declared using the ‘static’ keyword with no need to reference an instance of a class.*
    - *Recommended use is when only one copy of the variable is needed and used by all objects and or when all objects need to share a variable*
    - *Can also use the ‘private’*
* **Encapsulation**
  + It is to encase information and behavior within an object by restricting direct access to its data.
    - To prevent errors in code and to maintain more control over the code
    - This can be done using the ‘private’ modifier
* **Visibility Modifiers**
  + ***Public-*** *Indicates that the class, method or variable can be accessed from ANY other classes*
  + ***Private-*** *Indicates the method or variable can be accessed ONLY from within its own class and can only apply to members of a class (variables/methods)*
  + ***Default (Package Private)-*** *When no modified is specified it indicates the class, method, or variable can be accessed by any class within the same package*
* **Pass by Value or Reference** 
  + ***Pass by Value-*** *When the copy of the actual value is passed like primitive data types*
  + ***Pass by Reference-*** *When a memory address is passed that refers to where values are stored on heap.*
* **Stack and Heap**
  + ***Stack-*** *A part of memory that stores primitive datatypes and the address (id) of the object.*
    - In the debugger, if it stores the value, it is probably a primitive data type
    - If the debugger shows that it stored an id it probably stored the memory/reference of an object
  + ***Heap-*** *Another part of memory that stores non-primitive datatypes and the objects data.*

## **M01 L02**

* **Encoding**
  + to represent something with another thing
    - ***Unicode-*** *It's a 16-bit character scheme that can store the 'char' data type and is bigger than ASCII*
    - ***ASCII-*** *An 8-bit character scheme that provides encoding for 128 characters based on the English Alphabet*
* **Casting**
  + Converting a datatype to a different datatype
    - Ex) Numeric types like (byte, short, int, long, float, and double) can be cast into ‘char’.
    - Char uses single quotes ( ‘ ) while string uses double quotes ( “ )
* **String Methods**
  + ***.length()-*** *Returns the number of characters in a string.*
    - Ex) "Willy".length() **returns** 5.
  + ***.charAt(index)-*** *Returns the character at the specified index within a string.*
    - Ex) "Willy".charAt(2) **returns** 'l'.
  + ***.concat(string1)-*** *Combines the specified string at the end of the current string.*
    - Ex) "Hello".concat(" World") **returns** "Hello World"
  + ***.toLowerCase()-*** *Returns a string in all lowercase.* 
    - Ex) "HELLO".toLowerCase() **returns** "hello".
  + ***toUpperCase()-*** *Returns a string in all uppercase.* 
    - Ex) "hello".toUpperCase() **returns** “HELLO”
* **Array**
  + a data structure that provides a way to store multiple values of the same data type with a starting index of 0
    - Fixed size once created, and all elements must be the same type
    - Accessed through a single variable name
      * To declare an array, you need to use the ‘new’ keyword
        + Ex) (dataType[] arrayName = **new** dataType[size];)
      * To find the last index it is (length – 1)
    - If the index is beyond the preset size, it will cause an **out of bounds** runtime error
* **Scope**
  + It defines the section of a program where a variable is visible and can be accessed.