**DAY 2:**

**Give me some idea on what is a Data type. What are Primitive Data types and Non-Primitive Data types. Explain with some examples**

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 Java data types.

Data types in java are of different sizes and values that can be stored in the variable that is made as per convenience and circumstances to cover up all test cases. Java has two categories in which data types are segregated

1. Primitive Data Type: such as boolean, char, int, short, byte, long, float, and double. The Boolean with uppercase B is a wrapper class for the primitive data type boolean in Java.
2. Non-Primitive Data Type or Object Data type: such as String, Array, etc.

**Primitive Data Types in Java**

In Java, primitive data types are the most basic types of data that represent simple values. These are not objects and are directly handled by the Java Virtual Machine (JVM). They are the building blocks for data manipulation and serve as the foundation for data structures and objects.

**1. boolean**

* Represents a logical value: either true or false.
* Size: VM-dependent (typically 1 byte).
* Cannot be implicitly or explicitly converted to other types, though custom conversion can be implemented.

**2. byte**

* An 8-bit signed two’s complement integer.
* Size: 1 byte.
* Useful for saving memory in large arrays or data that doesn't require a larger range.

**3. short**

* A 16-bit signed two’s complement integer.
* Size: 2 bytes.
* Typically used for memory savings in large arrays when space is constrained.

**4. int**

* A 32-bit signed two’s complement integer.
* Size: 4 bytes.
* In Java SE 8 and later, can be used to represent an unsigned 32-bit integer with a range of [0, 2^32 - 1] using the Integer class.

**5. long**

* A 64-bit signed two’s complement integer.
* Size: 8 bytes.
* Can be used to represent an unsigned 64-bit integer (range 0 to 2^64 - 1) with Java SE 8+ using the Long class, supporting methods for unsigned arithmetic.

**6. float**

* A 32-bit IEEE 754 single-precision floating-point number.
* Size: 4 bytes.
* Used when memory efficiency is needed for large arrays of floating-point numbers, but with less precision than double.

**7. double**

* A 64-bit IEEE 754 double-precision floating-point number.
* Size: 8 bytes.
* Default for decimal values, but not ideal for high precision. If accuracy is critical, use BigDecimal instead.

**8. char**

* A 16-bit Unicode character.
* Size: 2 bytes.
* Used for representing single characters (letters, digits, symbols) in Java.

**Non- Primitive Data Types in Java**

Non-primitive data types, also known as reference types, are more complex than primitive types. These types refer to objects and can store multiple values. Non-primitive types are defined by classes and are used to represent more complex structures, collections, or user-defined data.

**1.String**:

* Represents a sequence of characters (text).
* Not a primitive type but a class in Java.
* Immutable, meaning once created, its value cannot be changed.
* Example: String name = "Java";

**2. Arrays:**

* A collection of elements (of the same type) stored in a single variable.
* Can be of any primitive or non-primitive type.
* Arrays in Java are objects, and their size is fixed once created.
* Example: int[] numbers = {1, 2, 3};

**3.Classes:**

* A blueprint for creating objects that define properties (fields) and methods (functions).
* Classes are used to define custom types.

**4.Interfaces**:

* Defines a contract that classes can implement.
* Cannot be instantiated but can be used to create a reference type.

**5.Enums**:

* A special type that represents a fixed set of constants (values).
* Used to define variables that can only have one of a small set of predefined values.

**6.Collections** (like Lists, Sets, Maps):

* These are classes provided in the Java Collections Framework (e.g., ArrayList, HashSet, HashMap).
* Used to store and manage groups of objects.