**1. Introduction to Data Types in Java**

In Java, data types are used to specify the type of data a variable can hold. Java is a **strongly typed** language, meaning variables must be explicitly declared with a type. There are two main categories of data types in Java:

1. **Primitive Data Types**
2. **Reference Data Types**

**2. Primitive Data Types**

Primitive data types are the most basic data types that represent simple values. Java has **8 primitive data types**:

1. **byte**
   * **Size**: 1 byte (8 bits)
   * **Range**: -128 to 127
   * **Description**: Used to save memory in large arrays, mostly in places where memory savings are important.
   * **Example**:

java

byte b = 100;

1. **short**
   * **Size**: 2 bytes (16 bits)
   * **Range**: -32,768 to 32,767
   * **Description**: Used for smaller integer values where memory optimization is a concern.
   * **Example**:

java

short s = 10000;

1. **int**
   * **Size**: 4 bytes (32 bits)
   * **Range**: -2^31 to 2^31 - 1
   * **Description**: Most commonly used for integers. It is the default choice for integers in Java.
   * **Example**:

java

int i = 100000;

1. **long**
   * **Size**: 8 bytes (64 bits)
   * **Range**: -2^63 to 2^63 - 1
   * **Description**: Used when the range of int is not enough.
   * **Example**:

java

long l = 10000000000L; // 'L' is used to indicate a long literal

1. **float**
   * **Size**: 4 bytes (32 bits)
   * **Range**: 1.4E-45 to 3.4E+38
   * **Description**: Used for storing decimal numbers with single precision.
   * **Example**:

java

float f = 10.5f; // 'f' is used to indicate a float literal

1. **double**
   * **Size**: 8 bytes (64 bits)
   * **Range**: 4.9E-324 to 1.8E+308
   * **Description**: Used for storing decimal numbers with double precision. It's the default choice for floating-point numbers in Java.
   * **Example**:

java

double d = 10.123456789;

1. **char**
   * **Size**: 2 bytes (16 bits)
   * **Range**: 0 to 65,535 (Unicode characters)
   * **Description**: Represents a single 16-bit Unicode character. Used to store characters.
   * **Example**:

java

char c = 'A';

1. **boolean**
   * **Size**: 1 byte (but JVM might implement it differently)
   * **Description**: Represents a value of true or false. Used for conditional statements and flags.
   * **Example**:

java

boolean isJavaFun = true;

**3. Reference Data Types**

Reference data types are used to store references (memory addresses) to objects, arrays, or instances of classes. Unlike primitive data types, reference types do not hold the actual data but instead hold a reference to the location in memory where the data is stored.

1. **Classes**
   * A class is a blueprint for creating objects. The data stored in the object is accessed using an instance of the class.
   * **Example**:

java

class Person {

String name;

int age;

}

Person person1 = new Person();

person1.name = "John";

person1.age = 30;

1. **Interfaces**
   * An interface in Java defines a contract for classes that implement it, specifying the methods that must be implemented.
   * **Example**:

java

interface Animal {

void sound();

}

class Dog implements Animal {

public void sound() {

System.out.println("Woof!");

}

}

1. **Arrays**
   * An array is a collection of similar types of data elements stored at contiguous memory locations. The array is a reference type in Java.
   * **Example**:

java

int[] numbers = new int[5]; // Array of integers

numbers[0] = 10;

1. **String**
   * Strings in Java are objects, even though they behave like primitive types. Strings are immutable, meaning once a string is created, its value cannot be changed.
   * **Example**:

java

String greeting = "Hello, World!";

**4. Autoboxing and Unboxing**

In Java, you can convert between primitive types and their corresponding wrapper classes automatically. This feature is called **autoboxing** (primitive to object) and **unboxing** (object to primitive).

* **Autoboxing**: Converting a primitive type to its wrapper class.
  + **Example**:

java

int i = 10;

Integer obj = i; // Autoboxing: int to Integer

* **Unboxing**: Converting a wrapper class object to its corresponding primitive type.
  + **Example**:

java

Integer obj = 10;

int i = obj; // Unboxing: Integer to int

**5. Wrapper Classes**

Java provides wrapper classes for each primitive data type. These wrapper classes are part of the **java.lang** package and provide methods to convert between primitive types and objects.

* **Byte** for byte
* **Short** for short
* **Integer** for int
* **Long** for long
* **Float** for float
* **Double** for double
* **Character** for char
* **Boolean** for boolean

**6. Type Casting**

Type casting allows you to convert from one data type to another. There are two types of type casting in Java:

1. **Implicit Casting (Widening)**
   * Automatically performed by the Java compiler when you convert a smaller data type to a larger one.
   * **Example**:

java

int i = 10;

long l = i; // Implicit casting from int to long

1. **Explicit Casting (Narrowing)**
   * Requires you to explicitly define the conversion using parentheses.
   * **Example**:

java

double d = 10.5;

int i = (int) d; // Explicit casting from double to int