**Variable Types:**

In **Java**, variables are basically classified into two categories:

1. **Primitive Variables**
2. **Non-Primitive Variables**

**1. Primitive Variables**

* **Primitive variables** store **simple values** like numbers, characters, or Boolean values.
* These variables are **predefined** by Java and are **not objects**.
* They store data **directly** in memory.

**Example:**

|  |
| --- |
| **public class** PrimitiveExample  {  **public static void** main(String[] args)  {  **int** num = 25;  **char** letter = 'A';  **boolean** isActive = true;  **double** price = 99.99;  System.out.println("Integer: " + num);  System.out.println("Character: " + letter);  System.out.println("Boolean: " + isActive);  System.out.println("Double: " + price);  }  } |

## ****2. Non-Primitive Variables****

* **Non-Primitive variables** store **references to objects** instead of actual data.
* They are **created by the programmer** using classes.
* Examples include **Strings, Arrays, Classes, and Objects.**
* They are also called **Reference Variables**.

### **Common Non-Primitive Data Types: String, Arrays, Classes, Objects, Interfaces**

**Example:**

|  |
| --- |
| **public class** NonPrimitiveExample  {  **public static void** main(String[] args)  {  **// String is a non-primitive data type**  String name = "Java Programming";  System.out.println("Name: " + name);    **// Object (Non-Primitive)**  NonPrimitiveExample **np** = new NonPrimitiveExample ();  }  } |

**Key Differences Between Primitive and Non-Primitive Variables:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Primitive Variables** | **Non-Primitive Variables** |
| **Definition** | Stores actual values | Stores reference (memory address) of the object |
| **Data Type** | Predefined (int, char, Boolean, etc.) | Created using classes (String, Array, Object, etc.) |
| **Memory** | Stored in **Stack** memory | Stored in **Heap** memory |
| **Operations** | Direct operations like addition, comparison | Methods and functions can be applied |
| **Default Value** | Based on the data type (e.g., 0, false) | null if not initialized |
| **Example** | int a = 10; | String name = "Java"; |

## ****Conclusion:****

* **Use Primitive Variables** when you need to store simple data like numbers, characters, or Boolean values.
* **Use Non-Primitive Variables** when you want to store and manipulate complex data like strings, arrays, or objects.

Again above **2-types** are further divided into **3-types** based on the position of declaration and behavior:

1. **Instance Variables** (Non-Static Variables or Global variables)
2. **Static Variables** (Class Variables)
3. **Local Variables**

### 1**. Instance Variables**

* **Declared inside a class but outside any method.**
* **Scope**: Belongs to an object of the class.
* **Lifetime**: Exists as long as the object exists.
* **Initialization**: Has default values (e.g., 0 for int, 0.0 for float, null for objects).

### **2. Static Variables**

* **Declared using the static keyword inside a class.**
* **Scope**: Shared among all objects of the class.
* **Lifetime**: Exists throughout the program's lifecycle.
* **Initialization**: Can be initialized once and accessed using the class name.

### **3. Local Variables**

* **Declared inside a method, constructor, or block.**
* **Scope**: Limited to the block in which they are defined.
* **Lifetime**: Created when the method is called and destroyed when it exits.
* **Initialization**: Must be initialized before use.

**static method()**

{

**instance family(variable/method) :** Through Class Object

**static family(variable/method) :** either directly or through class-name

}

**instance method()**

{

**instance family(variable/method) :** Directly

**static family(variable/method) :** either directly or through class-name

}