Non-Primitive Data types in Java with Example

-Siva Yannam

We know that data types in Java have been categorized into primitive and non-primitive data types.
Primitive data types are those data types that are used by programmers when creating variables in their programs. It is used to store a single entity (value).

For example, we take an “int” type that can store only one integer value. “boolean” can store only one value either true or false.

There are eight primitive data types in Java. They are boolean, char, byte, short, int, long, float, and double.

Now, we will learn another data type supported by Java known as non-primitive data types or advanced data types.

Non-Primitive Data types (Referenced Data types) in Java

Non-primitive data types are created by programmers. They are not predefined in java like primitive data types. These data types are used to store a group of values or several values.
For example, we take an array. It can store a group of values. Similarly, another example is a class that can store different values. Therefore, these data types are also known as **advanced data types in Java**.
When we define a variable of non-primitive data types, it references a memory location where data is stored in the heap memory. That is, it references a memory where an object is actually placed.

Therefore, a non-primitive data type variable is also called **referenced data type in Java** or simply object reference variable.

This object reference variable lives on the stack memory and the object to which it points always lives on the heap memory. The stack holds a pointer to the object on the heap.

In Java programming, all non-primitive data types are simply called objects which are created by instantiating a class.

Key points:

1. The default value of any reference variable is null.
2. Whenever you will pass a non-primitive data type to a method, you are actually passing an address of that object where data is stored.

How to Declare Non-primitive type Data types in Java?

In primitive data type, we declare like this:

```
int p = 100; // p is an int data type which can store the only integer value.
```

In reference data types, an object reference variable (or simply called reference variable) is declared just like we declare a primitive variable.

```
School sc;
```

Here, School is the name of a class, and “sc” is the name of a reference variable. No object has yet been created.

We create an object of a class using new keyword. For example, the following statement creates an object of a class School and assigns it to the reference variable “sc”.

```
sc = new School();
```

where,

School ➞ name of the class.
sc ➞ Object reference. An object reference is a variable that stores the address of an object in the computer's memory. An object represents an instance through which we can access member.

School() ➞ Constructor of the class. The constructor of a class is generally used to initialize an object.
new ➞ is a special keyword that creates the memory in java.
Now an object of class School lives on the heap and the object reference variable "sc" refers to it.

The declaration of an object reference variable, object creation, and initialization of reference variable can also be done in a single line statement like this:

```
School sc = new School();
```

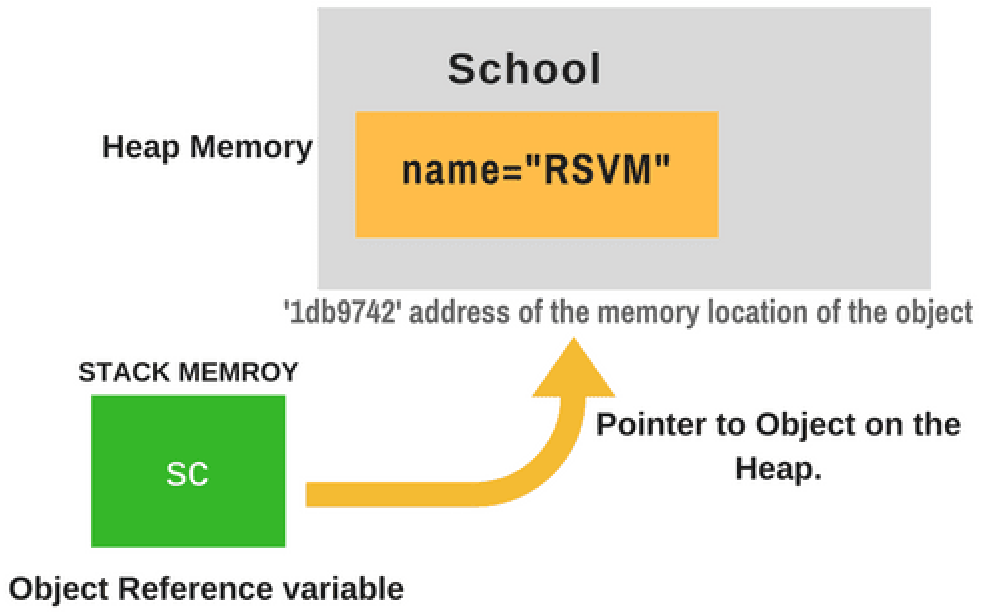
Let's take a simple example program. In this example program, we will get the object's address as output stored in object reference variable on the stack memory.

Program source code 1:

```
package scientecheasy;

public class School

{

// Declaration of a primitive variable.

  String name = "RSVM"; // Instance variable.



public static void main(String[] args)
{

// Creating an object of the class.

  School sc = new School(); // sc is Non-primitive data type i.e Object REFERENCE.

// Print the address of the memory location of an Object.

  System.out.println(sc);



// Now we cannot access instance variable directly. we call instance variable by using
reference variable sc which is created above.

  System.out.println(sc.name);

 }

}

Output:

        School@1db9742

        RSVM
```

Memory Allocation of Object & Object Reference Variable

From above example program, you will have understood that in Java, a variable whose type is a class, does not actually hold an object. Actually, it holds the memory location of an object.


School
Heap Memory
name="RSVM"
'1db9742' address of the memory location of the object
STACK MEMROY
sc
Pointer to Object on the Heap.
Object Reference variable

As shown in the above figure, Object reference variable ‘sc’ contains address ‘1db9742’ which is the address of memory location of an object on the heap. On this address, data is stored inside the heap memory.

Creating an object means storing data in memory. So, we can say that “sc” variable does not contain the object. It refers to an object.

Types of Non-Primitive Data types

There are five types of non-primitive data types in Java. They are as follows:

1. Class
2. Object
3. String
4. Array

5. Interface

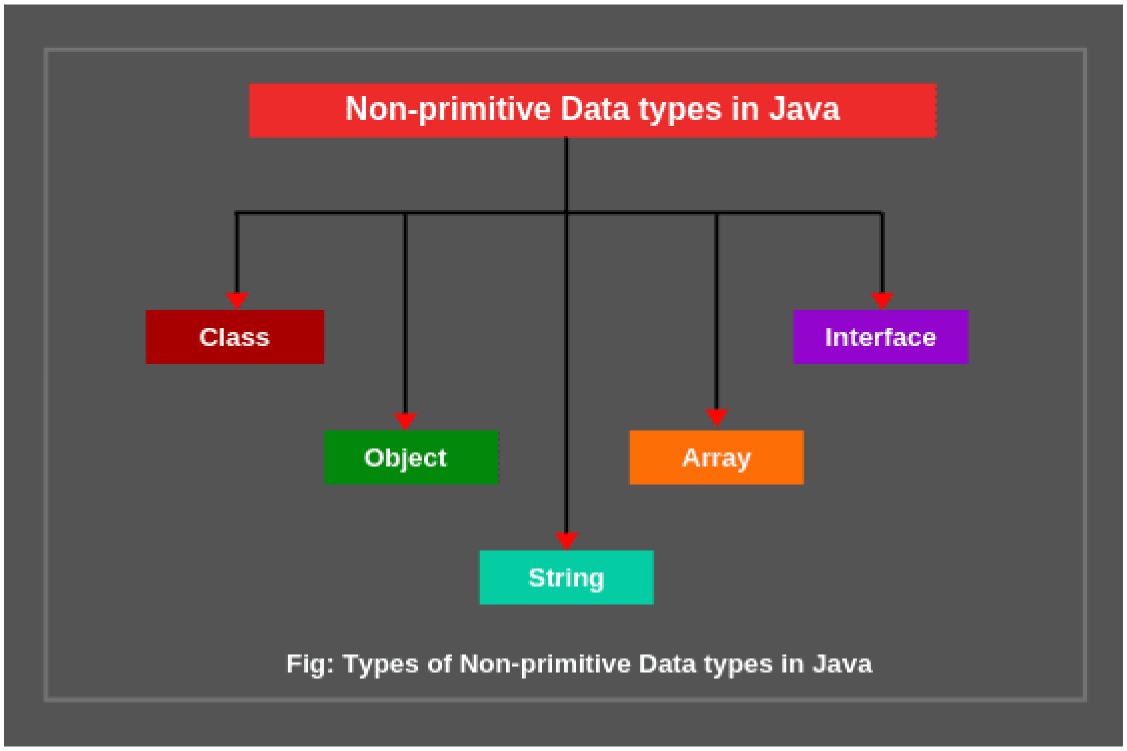


Fig: Types of Non-primitive Data types in Java

1. Class and objects: Every class is data type and it is also considered as user-defined data types. This is because a user creates a class. For more details: Class and objects in java

2. String: A string represents a sequence of characters like India, ABC123, etc. The simplest way to create a string object is by storing sequence of characters into string type variable like this:
String str = “Universe”;

Here, string type variable str contains “Universe”. A string is also a class. For more details: String in Java.

3. Array: An array in java is an object which is used to store multiple variables of the same type. These variables can be primitive or non-primitive data types.
The example of declaring an array variable of primitive data type int is as follows:

int [ ] scores;

The example of declaring an array variable of non-primitive data type is

Student [ ] students; // Student is a name of class.

You will learn more details in further tutorials.

4. Interface: An interface is declared like a class but the only difference is that it contains only final variables and method declarations. It is a fully abstract class.
Here, we have given just basic knowledge of non-primitive data types in java. You will get more knowledge in further tutorials.

Difference between Primitive and Non-primitive Data types in Java

---

1. Primitive data types are predefined in Java whereas non-primitive data types are created by programmers. They are not predefined in Java.

2. In primitive data type, variables can store only one value at a time whereas, in non-primitive data type, we can store multiple values either the same type or different type or both.

3. All the data for primitive type variables are stored on the stack whereas, for reference types, the stack holds a pointer to the object on the heap.

Hope that this tutorial has covered almost all the important points related to non-primitive data type in Java with example program. Keep in mind the following important points related to Java non-primitive data types.

Key Points:

1. Class, object, array, string, and interface are called non-primitive data types in Java. These data types are not predefined in Java. They are created by programmers.

2. Non-primitive data types are used to store a group of values.

3. When we define a variable of non-primitive data type, it references a memory location where data is stored in heap memory. Therefore, it is also known as reference data type in Java.

4. Passing a value of non-primitive data type to a method refers to actually passing an address of that object where data is stored.