Java Data Types

By Umesh Sir

Contact us 7758094241

Data Types

- ★ What is Data Types?
- ★ Primitive data types
- ★ Non-Primitive data types
- ★ Java User Input (Scanner)

What is Data Types?

- The type for the data (Values) is called Data Type.
- Data type defines the values that a variable can take, for example if a variable has int data type, it can only take integer values.
- In java we have two categories of data type:

  1) Primitive data types :-boolean, char, byte, short, int, long, float and double

  2) Non-primitive data types :-Arrays and Strings are non-primitive data types.
- Java is a statically typed language. A language is statically typed, if the data type of a variable is known at compile time. This means that you must specify the type of the variable (Declare the variable) before you can use it.

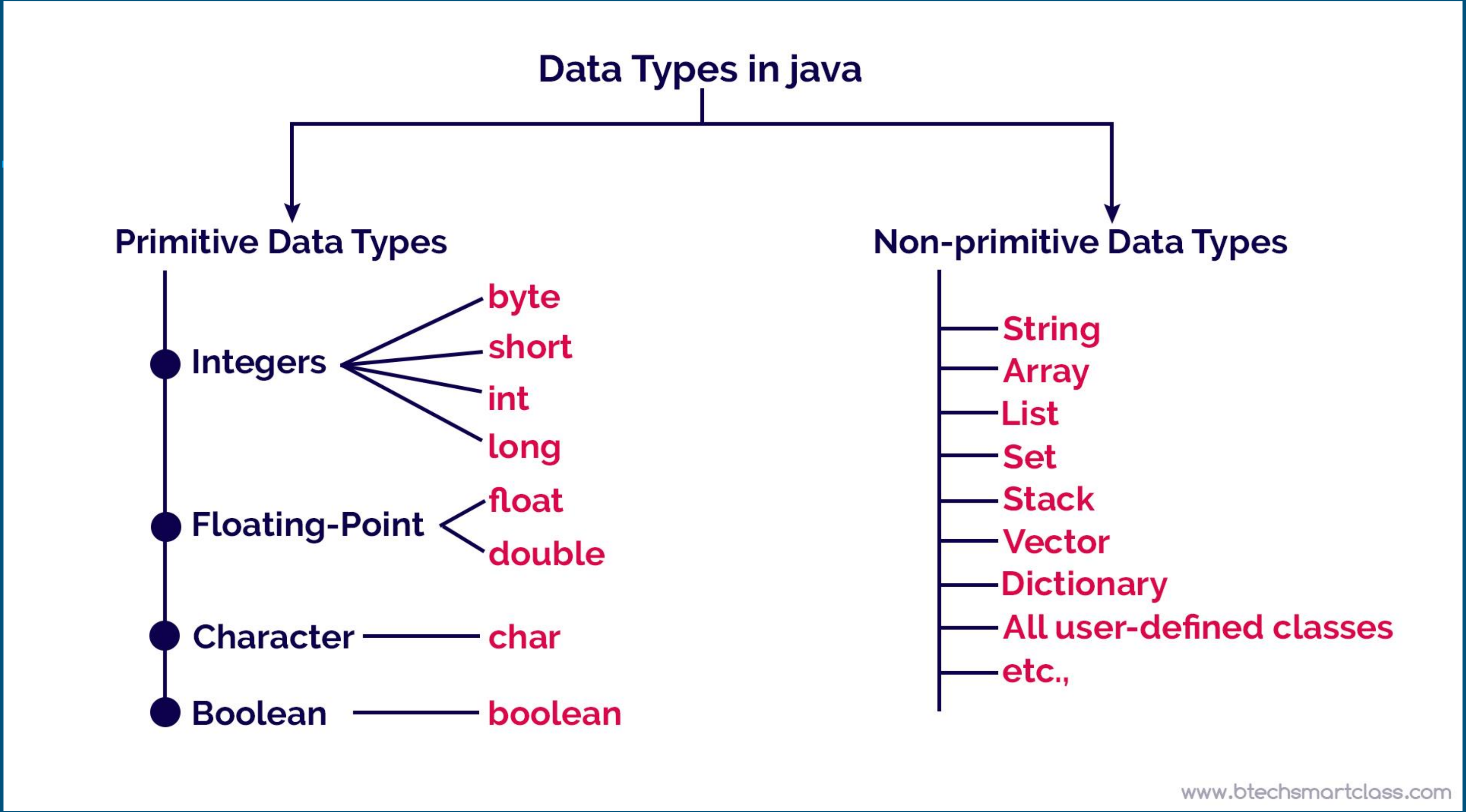
```
int num;
```

- So in order to use the variable num in our program, we must declare it first as shown above. It is a good programming practice to declare all the variables (that you are going to use) in the beginning of the program.

★ Primitive data types

Java developers included these data types to maintain the portability of java as the size of these primitive data types do not change from one operating system to another.

- ★ byte,short,int and long data types are used for storing whole numbers.
- ★ float and double are used for fractional numbers.
- ★ char is used for storing characters(letters).
- ★ boolean data type is used for variables that holds either true or false.


Data Types in java
Primitive Data Types
Integers
byte
short
int
long
Floating-Point
float
double
Character
char
Boolean
boolean
Non-primitive Data Types
String
Array
List
Set
Stack
Vector
Dictionary
All user-defined classes
etc.,
www.btechsmartclass.com

1.1] byte:

This can hold whole number between -128 and 127. Mostly used to save memory and when you are certain that the numbers would be in the limit specified by byte data type.

Default size of this data type: 1 byte.

Default value: 0

Example:

```
class JavaExample {
public static void main(String[] args) {
Byte num; num = 113;
System.out.println(num); } }
```

Output:

```
113
```

Try the same program by assigning value assigning 150 value to variable num, you would get type mismatch error because the value 150 is out of the range of byte data type. The range of byte as I mentioned above is -128 to 127.

1.2] short:

This is greater than byte in terms of size and less than integer. Its range is -32,768 to 32767.

Default size of this data type: 2 byte

Default value: 0

Short num = 45678;

The byte data type couldn’t hold the value 150 but a short data type can because it has a wider range.

1.3] int:

Used when short is not large enough to hold the number, it has a wider range: -2,147,483,648 to 2,147,483,647

Default size: 4 byte

Default value: 0

Example:

```
class JavaExample {
public static voidmain(String[] args)  {
            int num; num = 45678;
            System.out.println(num);  } }
```

Output: 45678

The short data type couldn’t hold the value 45678 but a int data type can because it has a wider range.

1.4] long:

Used when int is not large enough to hold the value, it has wider range than int data type, ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

size: 8 bytes

Default value: 0

Example:

```
class JavaExample{
 public static voidmain(String[] args) {
 long num = -12332252626L;
 System.out.println(num);  }}
```

Output:

```
-12332252626
```

1.5] double:

Sufficient for holding 15 decimal digits

size: 8 bytes

Example:

```
class JavaExample{
 public static void main(String[] args) {
  double num = -42937737.9d;
  System.out.println(num); }}
```

Output:

```
-4.29377379E7
```

1.6] float:

Sufficient for holding 6 to 7 decimal digits

size: 4 bytes

```
class JavaExample{
     public static void main(String[] args) {
     Float num = 19.98f;
    System.out.println(num); }}
```

Output: 19.98

1.7] boolean:

holds either true of false.

```
class JavaExample{
      public static voidmain(String[] args) {
     boolean b = false;
     System.out.println(b); }}
```

Output: False

1.8] char:

holds characters.

size: 2 bytes

```
class JavaExample{
  public static voidmain(String[] args) {
              char ch = 'Z';
                 System.out.println(ch); }}
                 Output: Z
```

★ Non-Primitive data types

Non-primitive data types are called reference types because they refer to objects.

The main difference between primitive and non-primitive data types are:

- Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for String).
- Non-primitive types can be used to call methods to perform certain operations, while primitive types cannot.
- A primitive type has always a value, while non-primitive types can be null.
- A primitive type starts with a lowercase letter, while non-primitive types starts with an uppercase letter.
- The size of a primitive type depends on the data type, while non-primitive types have all the same size.

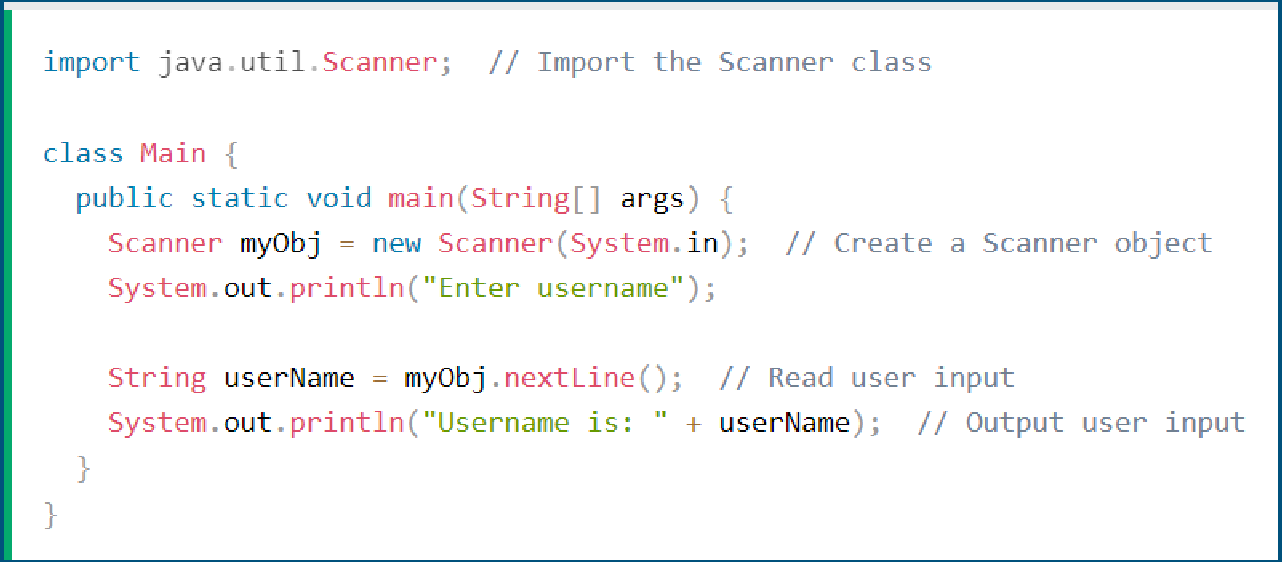
Example of Non-Primitive Data Type.

```
String name = “Umesh”;

String name1 = “Umesh Bichukale”;

Int[] arr = new int[10];

Int[] arr1 = new int[20];
```

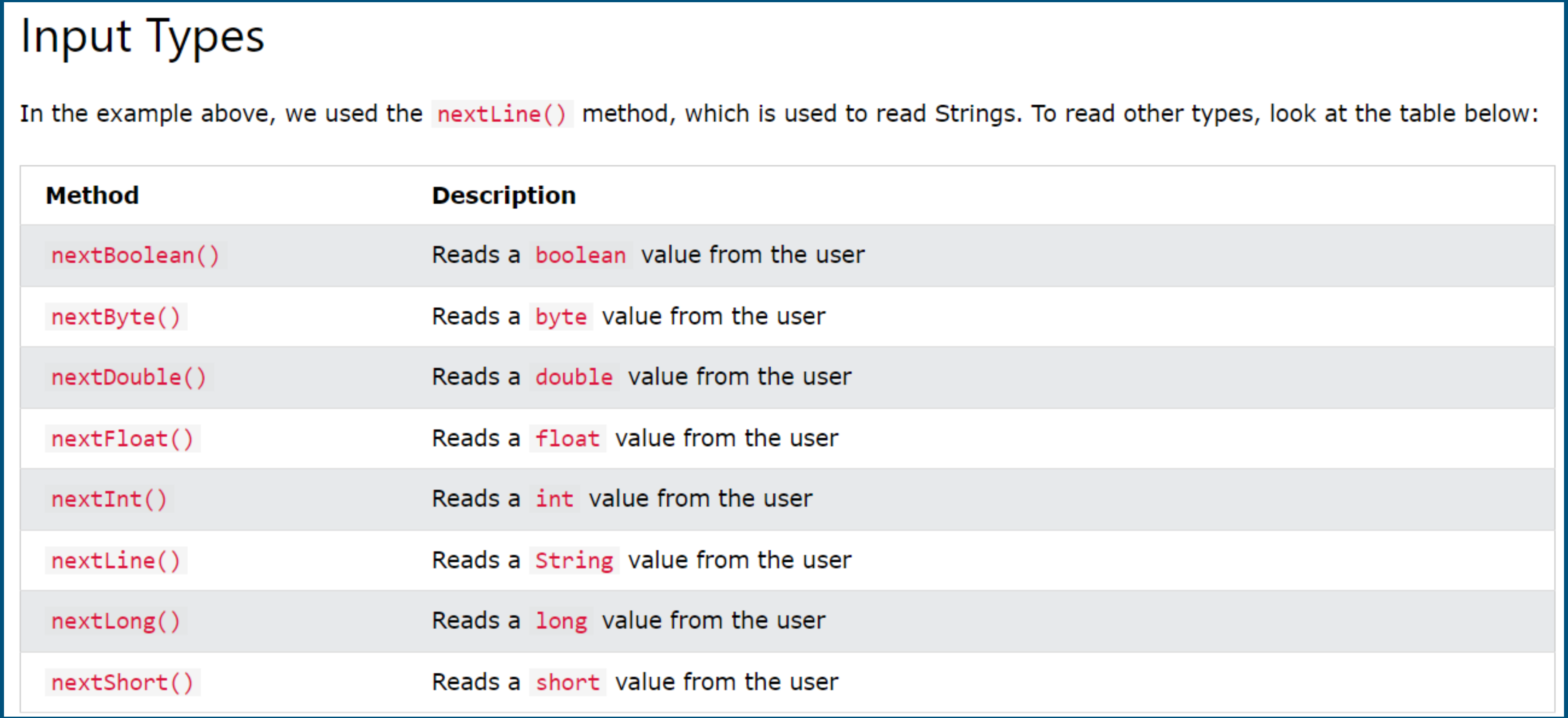
★ Java User Input (Scanner)

The Scanner class is used to get user input, and it is found in the java.util package.

★ To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation. In our example, we will use the nextLine() method, which is used to read Strings:



```
import java.util.Scanner;  // Import the Scanner class

class Main {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
    System.out.println("Enter username");

    String userName = myObj.nextLine();  // Read user input
    System.out.println("Username is: " + userName);  // Output user input
  }
}
```



Input Types

In the example above, we used the `nextLine()` method, which is used to read Strings. To read other types, look at the table below:

Method	**Description**
`nextBoolean()`	Reads a `boolean` value from the user
`nextByte()`	Reads a `byte` value from the user
`nextDouble()`	Reads a `double` value from the user
`nextFloat()`	Reads a `float` value from the user
`nextInt()`	Reads a `int` value from the user
`nextLine()`	Reads a `String` value from the user
`nextLong()`	Reads a `long` value from the user
`nextShort()`	Reads a `short` value from the user

Thank you