HowToDoInJava

Java Data Types

Last Updated: January 30, 2022 By: Lokesh Gupta Java Basics Data Types

Learn about various **data types in Java**. Learn the differences between primitive datatypes and non-primitive datatypes (or reference datatypes). We will also learn about the data types sizes and best practices to use datatypes in Java.

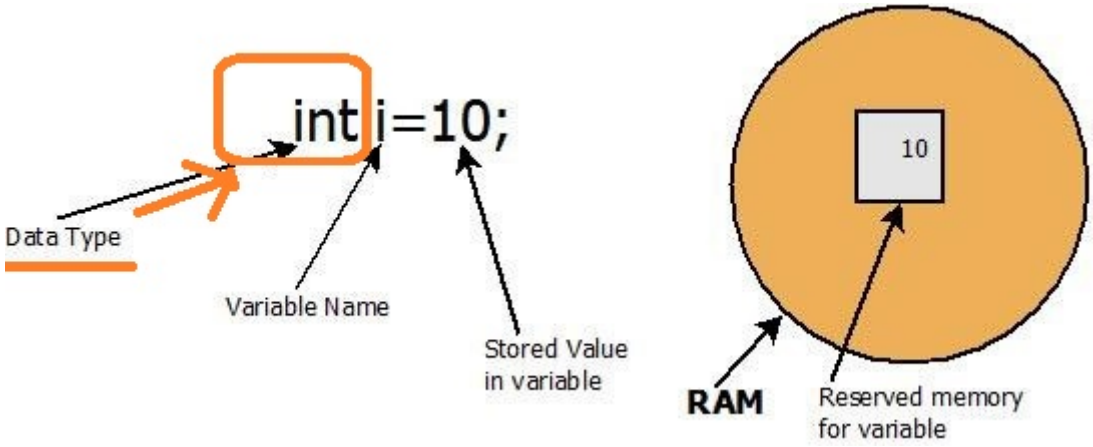
Table Of Contents

1. Variable Declarations

In Java, typically datatypes are associated with variables. A variable declaration has three parts:

1. A **variable name** (also called identifier) to refer to the memory location
2. The **variable type** stored at the memory location (it is called *datatype*)
3. A **memory location** to hold the value of the variable

The second property is called **data type**.



Java Data Type

> The data type of the variable determines the **range of the values** that the memory location can hold. Therefore, the amount of **memory allocated for a variable depends on its data type**.

For example, 32 bits of memory is allocated for a variable of the `'int'` data type.

Java is a statically-typed language. This means **all variables MUST be declared before they can be used**.

```
boolean flag = true;

int counter = 20;
```

2. Data Types

Java supports **two kinds of data types**:

1. Primitive data type
2. *Non-primitive* or reference data type.

2.1. Primitive Data Types

- A primitive data type *directly* holds a value in memory. For instance, a number or a character.
- Primitive data types are not objects, as well as no references to the objects.
- The values stored in primitives are called *literals*.

A **literal** is the source code representation of a fixed value; literals are represented directly in your code without requiring computation.

Java has **eight primitive data types**.

Data Type	`boolean`
Description	A binary value of either `true` or `false`
Default Value	`false`
Memory Size	1 bit
Data Type	`char`
Description	Any Unicode character
Default Value	`\u0000 (0)`
Memory Size	16-bit Unicode character
Data Type	`byte`
Description	Values from -128 to 127
Default Value	0
Memory Size	8-bit signed value
Data Type	`short`
Description	Values from -32768 to 32767
Default Value	0
Memory Size	16-bit signed value
Data Type	`int`

Description	Values from -2^{31} to $2^{31}-1$
Default Value	0
Memory Size	32-bit signed value
Data Type	`long`
Description	Values from -2^{63} to $2^{63}-1$
Default Value	0
Memory Size	64-bit signed value
Data Type	`float`
Description	IEEE 754 floating point
Default Value	0.0
Memory Size	32-bit floating-point value
Data Type	`double`
Description	IEEE 754 floating point
Default Value	0.0
Memory Size	64-bit floating-point value

> In Java SE 7 and later, any number of **underscore** characters (`'_'`) can appear anywhere between digits in a numerical literal. e.g. `10_000_000` is a valid number in Java. Read More

Type Conversion between Primitives

Except `boolean`, we can assign a primitive value to another primitive type. Though, sometimes it may result in **data loss** when a primitive of large memory capacity is assigned to a primitive with smaller memory capacity.

It's just like you are transferring the water from a large vessel and putting it in a smaller vessel, so the loss of water is natural.

```
int counter = 20_000_000;

//Assign int to short (data loss)
short shortCounter = (short) counter;

//assign int to long (no data loss)
long longCounter = counter;

System.out.println(counter);            //20000000
System.out.println(shortCounter);       //11520
System.out.println(longCounter);        //20000000
```

Notice that when Java detects that type conversion may result in data loss (bigger data type to smaller one), then gives a **type-mismatch error** and explicitly asks for **type casting** (e.g. 'int' to 'short' assignment). It helps in detecting and resolving accidental data loss assignments.

2.2. Non-primitive Data Types

A non-primitive or reference data type holds the reference to an object in memory. Using the reference stored in the variable, you can access the fields and methods of the referenced object.

For example, `java.lang.String` is a class defined in the Java library and you can use it to manipulate text (sequence of characters). You declare a reference variable of type `String` as:

```
String str = "Hello World !!";
```

What happens when this code is executed?

- First, a memory block is allocated, and the name of the variable `str` is associated with that memory location. This process is the same as declaring a primitive data type variable.
- The second part of code creates a new `String` object in memory with text `"Hello World !!"` and stores the reference (or memory address) of the `String` object into the variable `'str'`.

Multiple variables can refer to same object

You can also assign the reference of an object stored in one reference variable to another reference variable. In such cases, both reference variables will refer to the same object in memory.

```
// Declares String reference variable str1 and str2
String str1;
String str2;

// Assigns the reference of a String object "Hello" to str1
str1 = new String( "Hello World !!" );

// Assigns the reference stored in str1 to str2
str2 = str1;

System.out.println( str1 );          //Hello World !!
System.out.println( str2 );          //Hello World !!
```

There is a reference constant (also known as reference literal) **null**, which can be assigned to any reference variable. If `null` is assigned to a reference variable, which means that the reference variable is not referring to any object in memory.

3. Wrapper Classes

A wrapper class is a class whose object **wraps or contains primitive data types**. In other words, we can wrap a primitive value into a wrapper class object.

> Please note that Java has **one wrapper class mapped to each primitive data type**.

For example, `java.lang.Integer` class is the object version of int data type. Similarly, we have total of 8 wrapper classes for all 8 primitive data types.

The wrapper class names are the same as primitive data types names, only starting with a capital letter.

These wrapper classes are `Boolean`, `Byte`, `Short`, `Character`, `Integer`, `Long`, `Float` and `Double`.

4. Auto-boxing

In Java, you can assign a primitive type value to a wrapper class, directly.

For example, you can assign a **int** value to **Interger** class reference.

```
Integer counter = 20;

static Float PI = 3.14f;
```

It's worth mentioning that all wrapper class instances are immutable. They also maintain an internal cache for performance reason.

5. Difference between primitive and non-primitive data types

1. Primitives store values directly, which are called literals. Reference types store references to actual objects in memory area.
2. There are 8 fixed primitive data types. In Java, each class is a data type including wrapper classes.

6. Best practices

1. Use Java variable naming conventions and follow best practices.
2. Use primitives for variables which are local in scope. e.g. inside methods, counter for loops and intermediate results.
3. When data is transferred among method or classes, it's better to use objects because only their references will be copied, no memory overhead will be added.
4. When dealing with collections (which need objects), you shall be using Objects.
5. While sending data over network, use objects and make them `Serializable`. Wrapper classes are automatically `Serializable`.
6. Always know the size of data type you will need. Use appropriate data sizes. Using `int` to store `boolean` values (0 and 1) is waste of memory.
7. Use underscores (above Java 7) in numbers. It make them **more readable**.

Happy Learning !!

Recommended Reading:

1. Java Primitive Data Types
2. Python – Data Types

3. RESTEasy – Share Context Data with ResteasyProviderFactory
4. Reading and Writing UTF-8 Data into File
5. [Solved] HsqlException: data exception: invalid character value for cast
6. Hadoop – Big Data Tutorial
7. Masking Sensitive Data with Logback
8. Java – JDBC Driver Types
9. Cascade Types in JPA and Hibernate
0. TypeScript Union Types

Leave a Comment

Name *

Email *

Website

Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search ...

HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial