
Java Variables and Literals

In this tutorial, we will learn about Java variables and literals with the help of examples.

Java Variables

A variable is a location in memory (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name (identifier).

Learn more about [Java identifiers](#).

Create Variables in Java

Here's how we create a variable in Java,

```
int speedLimit = 80;
```

Here, `speedLimit` is a variable of `int` data type and we have assigned value **80** to it.

The `int` data type suggests that the variable can only hold integers. To learn more, visit [Java data types](#).

In the example, we have assigned value to the variable during declaration. However, it's not mandatory.

You can declare variables and assign variables separately. For example,

```
int speedLimit;  
  
speedLimit = 80;
```

Note: Java is a statically-typed language. It means that all variables must be declared before they can be used.

Change values of variables

The value of a variable can be changed in the program, hence the name **variable**. For example,

```
int speedLimit = 80;

... ..

speedLimit = 90;
```

Here, initially, the value of `speedLimit` is **80**. Later, we changed it to **90**.

However, we cannot change the data type of a variable in Java within the same scope.

What is the variable scope?

Don't worry about it for now. Just remember that we can't do something like this:

```
int speedLimit = 80;

... ..

float speedLimit;
```

To learn more, visit: [Can I change declaration type for a variable in Java?](#)

Rules for Naming Variables in Java

Java programming language has its own set of rules and conventions for naming variables. Here's what you need to know:

- Java is case sensitive. Hence, `age` and `AGE` are two different variables. For example,

```
int age = 24;int AGE = 25;

System.out.println(age); // prints 24

System.out.println(AGE); // prints 25
```

-
-

-
- Variables must start with either a **letter** or an **underscore, _** or a **dollar, \$** sign. For example,

```
int age; // valid name and good practice
int _age; // valid but bad practice
int $age; // valid but bad practice
```

-
- Variable names cannot start with numbers. For example,

```
int 1age; // invalid variables
```

-
- Variable names can't use whitespace. For example,

```
int my age; // invalid variables
```

-

Here, if we need to use variable names having more than one word, use all lowercase letters for the first word and capitalize the first letter of each subsequent word. For example, `myAge`.

- When creating variables, choose a name that makes sense. For example, `score`, `number`, `level` makes more sense than variable names such as `s`, `n`, and `l`.
- If you choose one-word variable names, use all lowercase letters. For example, it's better to use `speed` rather than `SPEED`, or `SPEED`.

There are 4 types of variables in Java programming language:

- Instance Variables (Non-Static Fields)
 - Class Variables (Static Fields)
 - Local Variables
 - Parameters
-

If you are interested to learn more about it now, visit [Java Variable Types](#).

Java literals

Literals are data used for representing fixed values. They can be used directly in the code. For example,

```
int a = 1; float b = 2.5; char c = 'F';
```

Here, `1`, `2.5`, and `'F'` are literals.

Here are different types of literals in Java.

1. Boolean Literals

In Java, boolean literals are used to initialize boolean data types. They can store two values: true and false. For example,

```
boolean flag1 = false; boolean flag2 = true;
```

Here, `false` and `true` are two boolean literals.

2. Integer Literals

An integer literal is a numeric value (associated with numbers) without any fractional or exponential part. There are 4 types of integer literals in Java:

1. binary (base 2)
2. decimal (base 10)
3. octal (base 8)
4. hexadecimal (base 16)

For example:

```
// binaryint binaryNumber = 0b10010; // octal int octalNumber = 027;  
  
// decimalint decNumber = 34;
```

```
// hexadecimal int hexNumber = 0x2F; // 0x represents hexadecimal// binary int binNumber = 0b10010;  
// 0b represents binary
```

In Java, binary starts with **0b**, octal starts with **0**, and hexadecimal starts with **0x**.

Note: Integer literals are used to initialize variables of integer types like `byte`, `short`, `int`, and `long`.

3. Floating-point Literals

A floating-point literal is a numeric literal that has either a fractional form or an exponential form. For example,

```
class Main {  
    public static void main(String[] args) {  
  
        double myDouble = 3.4;  
        float myFloat = 3.4F;  
  
        // 3.445*10^2  
        double myDoubleScientific = 3.445e2;  
  
        System.out.println(myDouble); // prints 3.4  
        System.out.println(myFloat); // prints 3.4  
        System.out.println(myDoubleScientific); // prints 344.5  
    }  
}
```

Note: The floating-point literals are used to initialize `float` and `double` type variables.

4. Character Literals

Character literals are unicode character enclosed inside single quotes. For example,

```
char letter = 'a';
```

Here, `a` is the character literal.

We can also use escape sequences as character literals. For example, `\b` (backspace), `\t` (tab), `\n` (new line), etc.

5. String literals

A string literal is a sequence of characters enclosed inside double-quotes. For example,

```
String str1 = "Java Programming";  
String str2 = "Programiz";
```

Here, `Java Programming` and `Programiz` are two string literals.
