JAVA VARIABLES

Java Variables

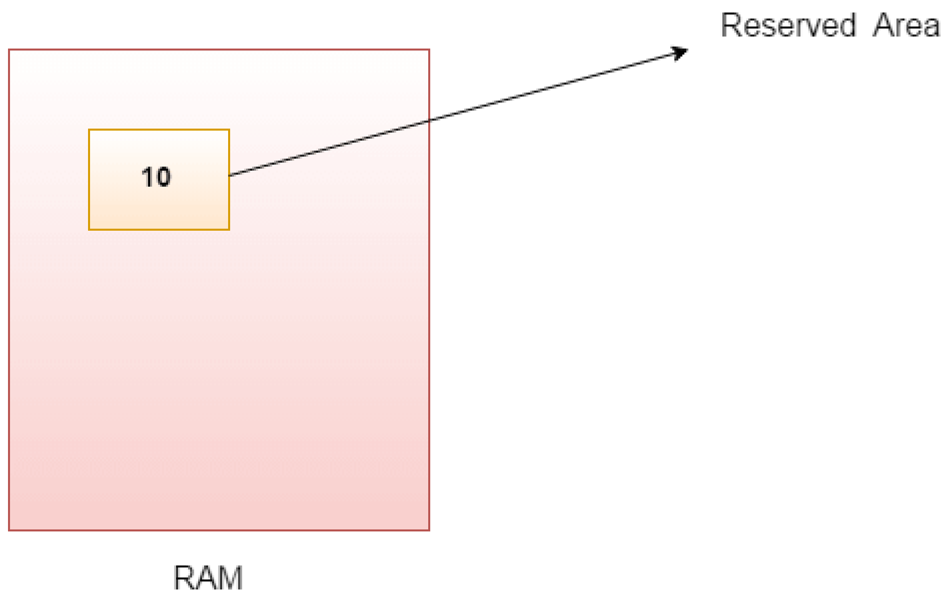
Last Updated : 2 Jan 2026

In Java, a variable is the name of memory location that are used like a container to store values during Java program execution. Every variable must be assigned a data type to define the kind of data it can hold. In Java, variables are categorized into three types: local, instance, and static variables. In this chapter, we will learn about the Java variables and types with the help of examples.

Before understanding Java variables and their types, you must be familiar with what a variable is and how it is named.

What is a Variable?

A variable is the name of a reserved area allocated in memory. In other words, it is a name of the memory location. It is a combination of "vary + able" which means its value can be changed.

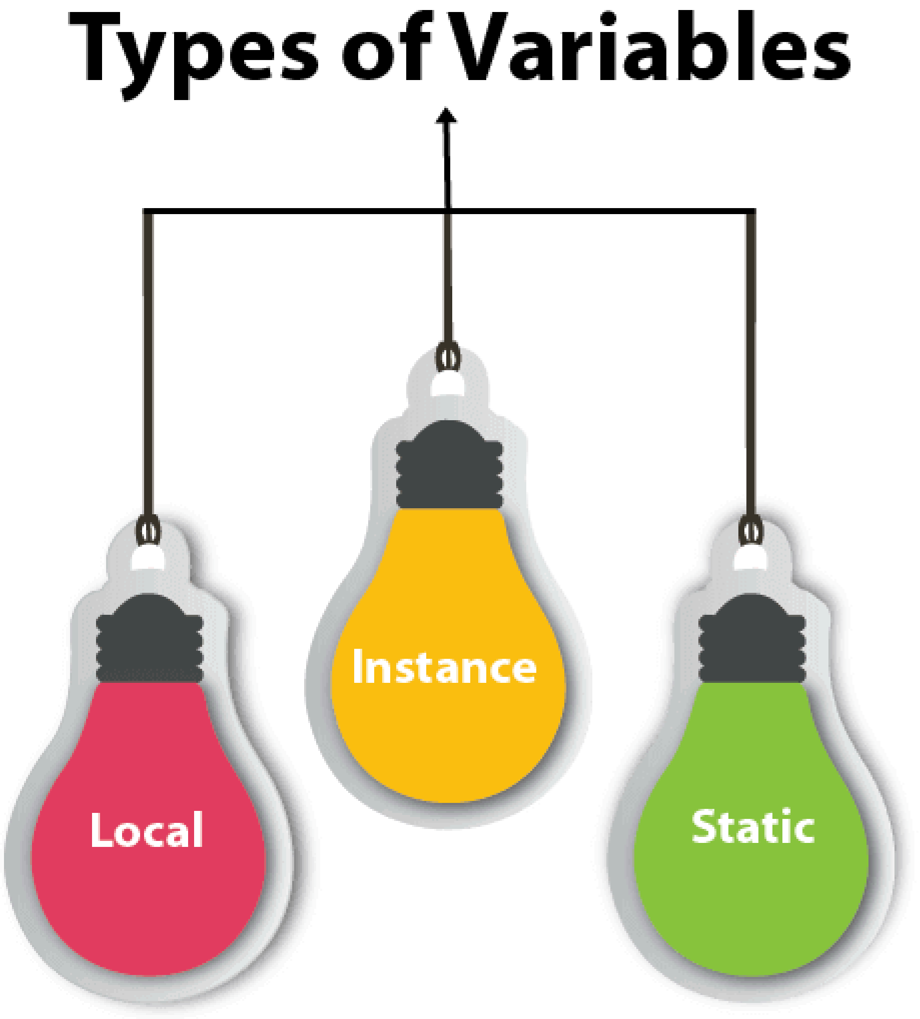


```
int data=50;//Here data is variable
```

Types of Java Variables

There are three types of variables in Java:

- Java Local Variable
- Java Instance Variable
- Java Static Variable



1) Java Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

Example of Local Variable

Let us take an example to demonstrate the Local Variable in Java.

Example

```
//defining a Local Variable
int num = 10;
System.out.println(" Variable: " + num);
```

Output:

```
Variable: 10
```

2) Java Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

Example of Instance Variable

Let us take an example to demonstrate the Instance Variable in Java.

Example

```
public class InstanceVariableDemo {
   //Defining Instance Variables
   public String name;
   public int age=19;
 //Creadting a default Constructor initializing Instance Variable
   public InstanceVariableDemo()
   {
      this.name = "Deepak";
   }
}
public class Main{
   public static void main(String[] args)
   {
      // Object Creation
      InstanceVariableDemo obj = new InstanceVariableDemo();
      System.out.println("Student Name is: " + obj.name);
      System.out.println("Age: "+ obj.age);
   }
}
```

Output:

```
Student Name is: Deepak
Age: 19
```

3) Java Static Variable

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Example Static Variable

Let us take an example to demonstrate the Static Variable in Java.

Example

```
class Student{
   //static variable
  static int age;
}
public class Main{
  public static void main(String args[]){
     Student s1 = new Student();
     Student s2 = new Student();
     s1.age = 24;
     s2.age = 21;
     Student.age = 23;
     System.out.println("S1\'s age is: " + s1.age);
     System.out.println("S2\'s age is: " + s2.age);
  }
```

```
    }
```

Output:

```
S1's age is: 23
S2's age is: 23
```

More Examples on Java Variables

Let’s see some other examples for better understanding about the Java variables:

Variables Example 1: Add Two Numbers

In the following program, we use variables to store values and perform an addition operation. The variables a and b hold values, c stores their sum, and the result is printed.

Example

```
int a=10;
int b=10;
int c=a+b;
System.out.println(c);
```

Output:

```
20
```

Variables Example 2: Widening Type Casting

This example demonstrates implicit type casting in Java. The integer variable a stores the value 10, which is automatically converted to a float and assigned to f. Both values are then printed, showing that Java safely converts an int to a float without data loss.

Example

```
int a=10;
float f=a;
System.out.println(a);
System.out.println(f);
```

Output:

```
10
10.0
```

Variables Example 3: Narrowing Type Casting

This example demonstrates explicit type casting from float to int in Java. Directly assigning f to a causes a compile-time error because a float cannot be automatically converted to an int. By using (int)f, the float value 10.5 is cast to 10, truncating the decimal part, and both values are printed.

Example

```
float f=10.5f;
//int a=f;//Compile time error
int a=(int)f;
System.out.println(f);
System.out.println(a);
```

Output:

```
10.5
10
```

Variables Example 4: Overflow

This example demonstrates overflow during explicit type casting in Java. The integer value 130 is explicitly cast to a byte, but since the byte range is from -128 to 127, the value overflows and results in -126.

Example

```
//Overflow
int a=130;
byte b=(byte)a;
System.out.println(a);
System.out.println(b);
```

Output:

```
130
-126
```

Variables Example 5: Adding Lower Type

This example shows type promotion in Java. Although a and b are bytes, the expression a + b is automatically promoted to int, causing a compile-time error when assigned to a byte. By explicitly casting the result to byte, the value is stored in c and printed successfully.

Example

```java
byte a=10;
byte b=10;
//byte c=a+b;//Compile Time Error: because a+b=20 will be int
byte c=(byte)(a+b);
System.out.println(c);
```

Output:

```
20
```