

2^n -> 2^8 = 256 /2 = 128 -128 to 127

| **Type** | **Description** | **Default** | **Size** | **Example Literals** | **Range of values** |
| --- | --- | --- | --- | --- | --- |
| **boolean** | true or false | FALSE | 1 bit | true, false | true, false(1,0) |
| **byte** | two's-complement integer | 0 | 8 bits | (none) | -128 to 127 |
| **char** | Unicode character | \u0000 | 16 bits | ‘a’, ‘\u0041’, ‘\101’, ‘\\’, ‘\’, ‘\n’, ‘β’ | characters representation of ASCII values  0 to 255 |
| **short** | two's-complement integer | 0 | 16 bits | (none) | -32,768 to 32,767 |
| **int** | two's-complement intger | 0 | 32 bits | -2,-1,0,1,2 | -2,147,483,648  to  2,147,483,647 |
| **long** | twos-complement integer | 0 | 64 bits | -2L,-1L,0L,1L,2L | -9,223,372,036,854,775,808  to  9,223,372,036,854,775,807 |
| **float** | IEEE 754 floating point | 0 | 32 bits | 1.23e100f , -1.23e-100f , .3f ,3.14F | upto 7 decimal digits |
| **double** | IEEE 754 floating point | 0 | 64 bits | 1.23456e300d , -123456e-300d , 1e1d | upto 16 decimal digits |

## **What are Primitive Types in Java?**

* Primitive types are the basic building blocks of Java.
* They are not objects.
* Their values are stored directly in **stack memory**.

int x = 10;

int y = x;

y = 20;

System.out.println(x); // Output: 10

✅ x and y are independent. Changing y does not affect x.

## **What are Reference Types?**

* Reference types include everything other than primitives: arrays, classes, interfaces, etc.
* Variables of these types store a reference (memory address) to the object in **heap memory.**

### **📌 Common Reference Types:**

* Classes (e.g., String, Scanner, ArrayList)
* Arrays
* Custom objects
* Interfaces

**Keywords in Java:**

In Java, **keywords** are predefined, reserved words that have a special meaning to the compiler. They cannot be used as **identifiers** (such as variable names, function names, or class names).

### **🔑 List of Java Keywords:**

| **Keyword** | **Purpose** |
| --- | --- |
| **abstract** | Defines an abstract class or method (incomplete and needs to be implemented by subclasses). |
| **assert** | Used for debugging purposes to validate assumptions. |
| **boolean** | Defines a variable of type boolean (true/false). |
| **break** | Exits from a loop or switch statement. |
| **byte** | Defines a variable that can store an 8-bit integer. |
| **case** | Defines a branch in a switch statement. |
| **catch** | Catches exceptions in a try-catch block. |
| **char** | Defines a variable that can store a single 16-bit Unicode character. |
| **class** | Defines a class. |
| **const** | Reserved but not used in Java (historical). |
| **continue** | Skips the current iteration of a loop and continues with the next iteration. |
| **default** | Specifies the default case in a switch statement. |
| **do** | Defines the start of a do-while loop. |
| **double** | Defines a variable that can store a 64-bit floating-point number. |
| **else** | Defines the alternative block of code in an if-else statement. |
| **enum** | Defines a set of constant values (enumerated type). |
| **extends** | Indicates inheritance, allows a class to inherit methods and fields from a superclass. |
| **final** | Defines a constant or prevents modification of a variable, method, or class. |
| **finally** | Defines a block of code that will execute after a try-catch, whether an exception was thrown or not. |
| **float** | Defines a variable that can store a 32-bit floating-point number. |
| **for** | Defines the start of a for loop. |
| **goto** | Reserved but not used in Java (historical). |
| **if** | Defines the condition in an if statement. |
| **implements** | Indicates that a class implements an interface. |
| **import** | Imports other classes or entire packages. |
| **instanceof** | Checks if an object is an instance of a specified class or subclass. |
| **int** | Defines a variable that can store a 32-bit integer. |
| **interface** | Defines an interface (a contract of methods). |
| **long** | Defines a variable that can store a 64-bit integer. |
| **native** | Specifies that a method is implemented in native code (outside of Java). |
| **new** | Creates new objects or arrays. |
| **null** | Represents the null reference (points to no object). |
| **package** | Defines a package to group related classes and interfaces. |
| **private** | Specifies that a member is accessible only within its own class. |
| **protected** | Specifies that a member is accessible within its own package and by subclasses. |
| **public** | Specifies that a member is accessible from anywhere. |
| **return** | Exits from a method and optionally returns a value. |
| **short** | Defines a variable that can store a 16-bit integer. |
| **static** | Defines a class-level method or field, shared by all instances of the class. |
| **strictfp** | Ensures that floating-point calculations adhere strictly to the IEEE 754 standard. |
| **super** | Refers to the superclass (parent) class or calls its constructor or method. |
| **switch** | Defines a switch statement to choose between multiple options. |
| **synchronized** | Ensures that a method or block of code is accessed by only one thread at a time. |
| **this** | Refers to the current object instance. |
| **throw** | Used to throw an exception manually. |
| **throws** | Declares exceptions that a method may throw. |
| **transient** | Prevents a field from being serialized (saved when the object is saved). |
| **try** | Defines the start of a block of code to try catching exceptions. |
| **void** | Specifies that a method does not return a value. |
| **volatile** | Ensures that a field’s value is always read from and written to the main memory. |
| **while** | Defines the start of a while loop. |

### **🔍 Key Points:**

* **Reserved**: These words are reserved for Java’s syntax and cannot be used as names for variables, methods, classes, etc.
* **No functionality for some**: For example, const and goto are **reserved** but **not used** in Java.

### **✅ Commonly Used Keywords:**

* **class**, **public**, **private**, **static**, **final**, **int**, **for**, **if**, **return**, **try**, **catch**, **throw**, **throws**