**Variable:**

A **variable** is a container that holds data. It has a **name**, a **type**, and a **value**.

**Real-Life Example of Variables: Coffee Machine ☕**

Imagine you have a **coffee machine**. It has different components like:

* **Water Tank** (stores water)
* **Coffee Beans Container** (stores coffee beans)
* **Sugar Container** (stores sugar)

Each of these components **stores some values**, just like variables in Java

**Syntax of Variable Declaration**

dataType variableName = value;

Example :-

int age = 20; // Stores an integer value

double price = 99.99; // Stores decimal values

char grade = 'A'; // Stores a single character

String name = "Gauri"; // Stores a sequence of characters

boolean isJavaFun = true; // Stores true or false

**🔴 Types of Variables in Java (With Range & Size)**

In Java, variables are classified into **3 types** based on **scope** and **usage**:

1. **Local Variables**
2. **Instance Variables (Non-Static Variables)**
3. **Static Variables (Class Variables)**

Additionally, based on **data type**, variables are divided into **Primitive** and **Non-Primitive**.

**1️. Local Variables**

* **Defined inside a method, constructor, or block**.
* Only accessible within that **method/block** (temporary storage).
* Must be **initialized before use**.

public class Example {

public static void main(String[] args) {

int age = 25; // Local variable (inside method)

System.out.println("Age: " + age);

}

}

**Key Point:** Cannot access age outside main() method.

**2️.Instance Variables (Non-Static Variables)**

* Declared **inside a class but outside methods**.
* Each object gets its **own copy** of instance variables.
* Initialized to **default values** if not assigned.

class Student {

String name; // Instance variable

int age; // Instance variable

public void display() {

System.out.println("Name: " + name + ", Age: " + age);

}

}

public class Main {

public static void main(String[] args) {

Student s1 = new Student();

s1.name = "Gauri";

s1.age = 22;

s1.display();

}

}

**🔹 Key Point:** Each Student object will have its own name and age.

**3️.Static Variables (Class Variables)**

* Declared using static keyword.
* Shared among **all objects** of the class (single memory allocation).
* Can be accessed using **class name**.

✅ **Example:**

class Company {

static String companyName = "Google"; // Static variable

}

public class Main {

public static void main(String[] args) {

System.out.println("Company: " + Company.companyName);

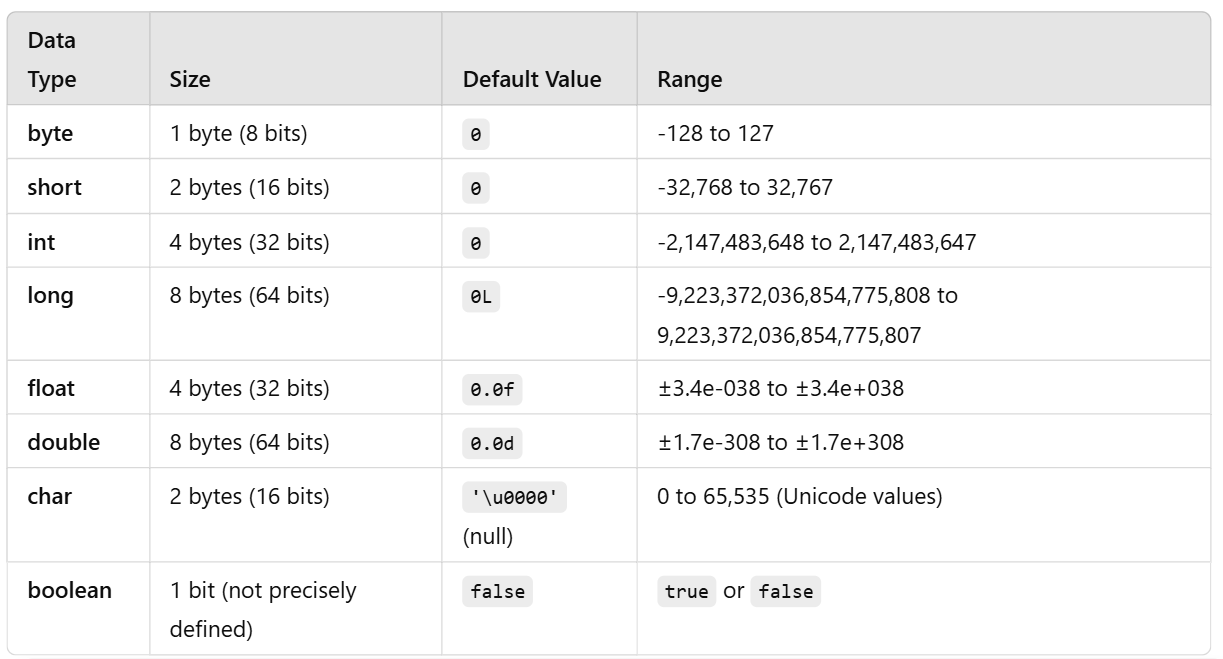
}

}

**🔹 Key Point:** Same companyName is shared across all instances.

**🔷 Primitive Data Types in Java (Size & Range)**

Java has 8 primitive data types, each with a specific size and range.



public class DataTypesExample {

public static void main(String[] args) {

byte smallNum = 100;

int largeNum = 50000;

long bigNum = 10000000000L;

float decimalNum = 3.14f;

double preciseDecimal = 3.1415926535;

char grade = 'A';

boolean isJavaFun = true;

System.out.println("Byte: " + smallNum);

System.out.println("Integer: " + largeNum);

System.out.println("Long: " + bigNum);

System.out.println("Float: " + decimalNum);

System.out.println("Double: " + preciseDecimal);

System.out.println("Character: " + grade);

System.out.println("Boolean: " + isJavaFun);

}

}

**🔷 Non-Primitive Data Types**

These are more complex than primitive types and **can store multiple values**.



✅ **Example Using Non-Primitive Data Types:**

public class NonPrimitiveExample {

public static void main(String[] args) {

String name = "Java"; // String

int[] numbers = {10, 20, 30}; // Array

System.out.println("String: " + name);

System.out.println("First Number: " + numbers[0]);

}

}

Sure! Let's break down **public static void main(String args[])** in the **simplest way** using **real-life examples** 🚀

**💡 Think of it Like This:**

Imagine you are starting a **new project in a company**. You need a **main manager** to begin the work. In Java, the main method is that **main manager** that starts the program.

**🔎 Breaking it Down - Word by Word**

| **Java Keyword** | **Real-Life Meaning** | **Simple Explanation** |
| --- | --- | --- |
| public | 🔓 **Public Door** | Anyone can enter (accessible by everyone). |
| static | 🏢 **Company Rule** | No need to create an employee (object), anyone can follow the rule. |
| void | 📬 **No Return Parcel** | The function does not return anything back. |
| main | 🏁 **Main Manager** | The starting point of the project (program). |
| String args[] | 📦 **Command Line Inputs** | Information given to the program when starting (not always needed). |

**📌 Real-Life Example**

**Imagine a Company Starting Work**

1️⃣ **public** → The **company building** is **open to everyone** (anyone can enter).  
2️⃣ **static** → There is a **company rule** that applies to **everyone** (no need to hire a special person).  
3️⃣ **void** → The company **doesn’t send** any **parcel (return value)** after starting work.  
4️⃣ **main** → The **main manager** who starts the **daily work**.  
5️⃣ **String args[]** → If the manager wants to receive **extra instructions** from the **CEO** (command-line arguments).

**📌 Java Code + Real-Life Connection**

public class Company {

public static void main(String args[]) {

System.out.println("Company Work Started!");

}

}

✅ **Output:**

Company Work Started!

🔹 main() starts the work like a **manager** in a company.

**🔹 Deep Dive Into Each Word (with Example)**

**🔴 1️.public → Open for Everyone**

public class Bank {

public static void main(String args[]) {

System.out.println("Bank is Open to Everyone");

}

}

✅ **Output:**

Bank is Open to Everyone

📌 public means **anyone can access** the main method.

**🔵 2️.static → No Need to Create an Object**

public class Calculator {

public static void main(String args[]) {

System.out.println("Calculator is working!");

}

}

✅ **Output:**

Calculator is working!

📌 static means we **don't need to create an object** to run main().

**🟢 3️.void → No Return Value**

public class Message {

public static void main(String args[]) {

System.out.println("This method does not return anything.");

}

}

✅ **Output:**

This method does not return anything.

📌 void means this method **does not return any value**.

**🟣 4️.main → The Starting Point**

public class Start {

public static void main(String args[]) {

System.out.println("Main Method Started!");

}

}

✅ **Output:**

Main Method Started!

📌 main is the **starting point of the program**.

**🟠 5️.String args[] → Command Line Arguments**

public class Greeting {

public static void main(String args[]) {

System.out.println("Hello, " + args[0] + "!");

}

}

📌 Run the program with:

java Greeting Gauri

✅ **Output:**

Hello, Gauri!

📌 args[] takes **extra inputs** from the user when running the program.

**🎯 Summary Table**

|  |  |  |
| --- | --- | --- |
| Java Keyword | Meaning | Real-Life Example |
| public | **Accessible by everyone** | A **shop** that is open for all customers. |
| static | **Can be used without creating an object** | **Electricity** is available in every house **without a special request**. |
| void | **Does not return anything** | A **robot** that only works but **does not give anything back**. |
| main | **Starting point** | The **CEO** who **starts the company** daily. |
| String args[] | **Command-line inputs** | Giving **extra instructions** to a **manager** before starting work. |

**🚀 Final Thoughts**

* public static void main(String args[]) is **mandatory** in Java programs.
* It is the **starting point** where execution **begins**.
* Each word has **special meaning** and **importance**.