Types of Variables in Java

In Java, variables are classified into three main types:

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

Each type has its own scope, lifetime, and storage behaviour. Let's explore each one in detail with code examples and output.

1. Instance Variables

Definition:

- Declared **inside a class but outside any method or constructor**.
- Each object of the class has its own copy.
- Stored in **Heap Memory**.
- Accessible only through **object reference**.

When to Use:

- When you want each object to maintain its own state or data.

Example:

```
public class Student {
   // Instance variable
   String name;
   int age;

   public void display() {
      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
   }
```

```
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Alice";
        s1.age = 20;
        s1.display();

        Student s2 = new Student();
        s2.name = "Bob";
        s2.age = 22;
        s2.display();
    }
}
```

Output:

```
Name: Alice
Age: 20
Name: Bob
Age: 22
```

2. Static Variables (Class Variables)

Definition:

- Declared with the static keyword inside a class but outside any method or constructor.
- Belongs to the **class**, not to individual objects.
- Stored in **Method Area**.
- Loaded when the class is loaded.
- Can be accessed **without creating an object**.

When to Use:

- When a property is shared among all objects (e.g., a constant or counter).

Example:

```
public class Employee {
   int id;
   String name;
   static String company = "TechCorp"; // static variable

   public Employee(int id, String name) {
      this.id = id;
      this.name = name;
   }

   public void display() {
      System.out.println(id + " " + name + " " + company);
   }

   public static void main(String[] args) {
      Employee e1 = new Employee(101, "John");
      Employee e2 = new Employee(102, "Jane");

      e1.display();
      e2.display();
   }
}
```

Output:

```
101 John TechCorp
102 Jane TechCorp
```

3. Local Variables

Definition:

- Declared **inside a method, constructor, or block**.
- Stored in **Stack Memory**.
- Scope is **limited to the method or block**.
- Not accessible outside the method.
- **No default values** – must be initialized explicitly.

When to Use:

- For temporary storage and calculations inside methods.

Example:

```
public class Calculator {
   public void sum() {
      int a = 10; // local variable
      int b = 20; // local variable
      int result = a + b;
      System.out.println("Sum: " + result);
   }

   public static void main(String[] args) {
      Calculator calc = new Calculator();
      calc.sum();
   }
}
```

Output:

```
Sum: 30
```

Best Practices:

- Use instance variables for object-specific data.
- Use static variables for common/shared data.
- Always initialize local variables before use.
- Keep variable names meaningful and follow naming conventions.