**Var Keyword in Java**

**1. Introduction**

* The var keyword was introduced in **Java 10** as part of **local-variable type inference**.
* It allows the compiler to **automatically infer the type** of a local variable from the value used to initialize it.
* This makes code cleaner, shorter, and reduces redundancy.

**2. Why var?**

Before Java 10, you had to explicitly declare types:

String name = "Anju";

int age = 25;

With var:

var name = "Anju"; // compiler infers String

var age = 25; // compiler infers int

**3. How it Works?**

* The **compiler infers** the type at **compile-time**, not runtime.
* Once a type is inferred, it **cannot change**.
* You **must initialize** the variable when declaring with var.

Example:

var x = 10; // int

x = 20; // ✅ allowed (still int)

x = "text"; // ❌ error (type mismatch, cannot change int to String)

**4. Rules for Using var**

Allowed:

* Local variables inside methods
* Index variables in loops (for, foreach)
* Try-with-resources statement

❌ Not Allowed:

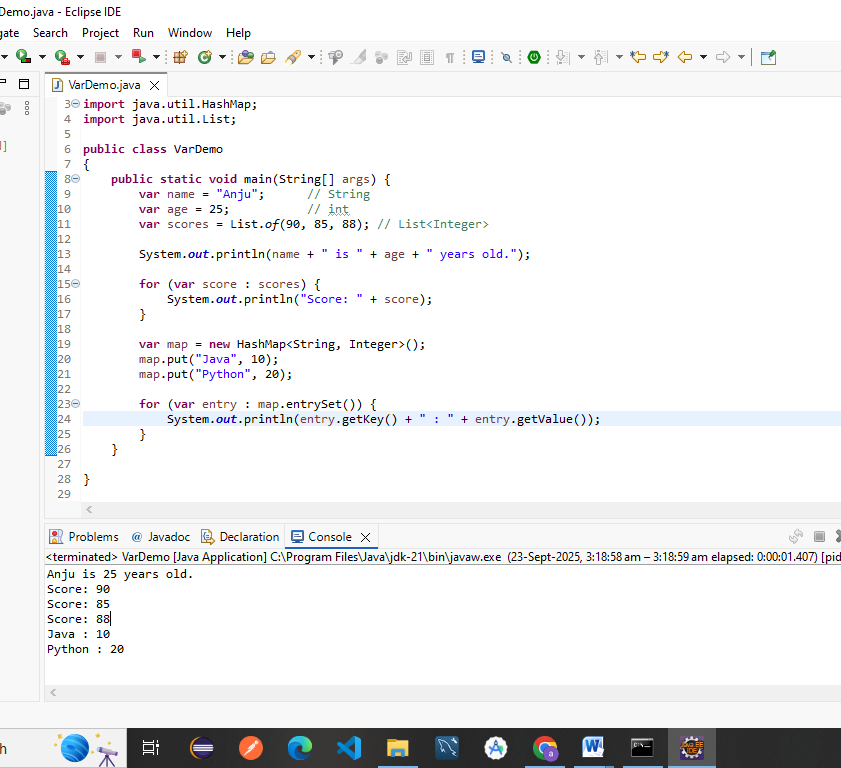
* Class fields (instance/static variables)
* Method parameters
* Method return types

**6. Advantages**

* Reduces boilerplate code
* Improves readability (especially with generics)
* Makes code cleaner in local scopes

**7. Disadvantages**

* Can reduce readability if used carelessly
* Not suitable for **public APIs** (return types, parameters)
* Sometimes it makes code confusing when type is not obvious:



## Code Explanation

### ****Imports****

import java.util.HashMap;

import java.util.List;

* HashMap → stores key–value pairs.
* List → stores ordered collections of elements.

### ****Class & Main Method****

public class VarDemo {

public static void main(String[] args) {

* Standard Java class with main() entry point.

### ****Using**** var ****for Variables****

var name = "Anju"; // String inferred

var age = 25; // int inferred

var scores = List.of(90, 85, 88); // List<Integer> inferred

* var lets the compiler **infer the type automatically** based on the value assigned.
* So:
  + name → String
  + age → int
  + scores → List<Integer>

### ****Printing Values****

System.out.println(name + " is " + age + " years old.");

➡ Prints:

Anju is 25 years old.

### ****Loop through Scores****

for (var score : scores) {

System.out.println("Score: " + score);

}

* Enhanced **for-each loop**.
* Iterates through scores list → prints each score.  
  ➡ Output:

Score: 90

Score: 85

Score: 88

### ****Using**** var ****with HashMap****

var map = new HashMap<String, Integer>();

map.put("Java", 10);

map.put("Python", 20);

* Here, map → HashMap<String, Integer>.
* Keys are programming language names (String).
* Values are numbers (Integer).

### ****Iterating Map Entries****

for (var entry : map.entrySet()) {

System.out.println(entry.getKey() + " : " + entry.getValue());

}

* map.entrySet() → set of key–value pairs.
* Each entry is of type Map.Entry<String, Integer>.
* Prints key and value.  
  ➡ Output:

Java : 10

Python : 20

## 📌 Final Program Output

Anju is 25 years old.

Score: 90

Score: 85

Score: 88

Java : 10

Python : 20