# Lec 16 Constructor:

This lesson explains how constructors work in Java, focusing on their role in initializing objects, the difference between default, no-argument, and parameterized constructors, and the concept of constructor chaining. Understanding these concepts is essential for managing object creation and data handling in Java programs.

**Setter and Getter update the value of the object because the default constructor is the one initialising them with default value(it invokes OBJECT class's constructor).**

**That's why constructor is used.**

If no constructor in class then JVM will provide default constructor and if we define constructor with no arg, its no args constructor.

this keyword is used to provide current class's object and it also differentiates between local variable and instance variable.

this() → constructor call (Chaining)

- only allowed inside the constructor

- used to call one constructor to another from same class
- this() must be the first statement inside the constructor
- recursion not allowed

## Encapsulation and Data Hiding

Encapsulation means keeping data (fields or variables) private within a class and providing controlled access using setter (to modify) and getter (to retrieve) methods, also called accessors and mutators. This protects data from unintended changes by outside classes.

## Data Binding and Object Passing

Data binding involves grouping all related data and methods (like setters and getters) into a single object, making it easier to pass information between parts of a program by sharing just the object reference instead of many separate variables.

## Effective Learning Habits for Java

Consistent practice, note-taking, and revision are crucial for mastering Java. Individual learning speed varies, so maintaining and regularly reviewing notes

helps reinforce concepts and retain knowledge over time.

### Clarifying Setter Usage and Reference Variables

Setters can receive values directly or via variables, and reference variables must have unique names within the same scope. You cannot declare two variables with the same name in the same block, even if they refer to objects from different classes.

### Local Variables and Access Modifiers

Local variables (declared inside methods) cannot have access modifiers like private or static in Java; these are only for class-level (instance) variables. Attempting to use them will cause errors.

### Introduction to Constructors

A constructor is a special block of code in a class, similar to a method but used specifically to initialize new objects. Its main job is to assign initial values to object fields when the object is created.

### Object Creation and Default Initialization

When using the new keyword to create an object, Java automatically allocates memory and sets default values (like zero for integers) for instance variables. If no initial value is provided, defaults are used.

### Updating vs. Initializing Object Data

Using setters after object creation updates the values of fields, but does not constitute true initialization. Initialization means assigning values at the time of object creation, which is best done using constructors.

### Creating Constructors: Rules and Types

Constructors must have the same name as the class and no return type. You can create multiple constructors in a class (constructor overloading) by changing their parameters. The default (no-argument) constructor is provided by Java if none is written, but disappears if any constructor is defined.

### Constructor Invocation and Overloading

Constructors are automatically called by the Java Virtual Machine (JVM) when an object is created. The constructor that matches the arguments provided in

the new statement is chosen. If no constructor is defined, the JVM supplies a default one.

### Default Constructor and Object Class

The default constructor initializes fields with default values and internally calls the constructor of the parent class (usually Object). If a programmer defines any constructor, the default one is not automatically provided.

### Parameterized Constructors and Custom Initialization

Parameterized constructors allow you to set custom values for object fields at creation time, ensuring objects start with meaningful data instead of defaults. Setters can still be used to update these values later.

### Constructor Chaining and 'this' Keyword

Constructor chaining allows one constructor to call another within the same class using this(), enabling step-by-step initialization. The this keyword refers to the current object, and this() must be the first statement in a constructor when used for chaining.

## Rules and Pitfalls of Constructor Chaining

Constructor chaining cannot form loops (a constructor cannot call itself directly or indirectly). Chaining helps avoid code duplication and ensures consistent initialization, but must follow rules like being the first statement and only within constructors.

## Summary and Next Steps

Mastering constructors and their chaining is key for robust Java class design. Advanced topics like copy constructors, deep and shallow copies, and inheritance will build upon this foundation.