📘 Constructor in Java — Complete Theory Notes

1. A **constructor** in Java is a **special type of method** that is used to **initialize objects** when they are created.
It is automatically executed when you create an object using the **new** keyword.
Key points:

- Constructor name must be the same as class name
- Constructor has no return type (not even void)
- Constructor is called **automatically** at object creation
- Used to initialize instance variables

2. Rules for Creating a Java Constructor

The constructor name must be exactly the same as the class name.
Constructors cannot be declared inside methods or other constructors — they must be declared directly inside the class body.

1.Constructors must not specify a return type (not even void).
2.If a return type is specified, it becomes a regular method, not a constructor.
3.Only access modifiers are allowed — public, protected, private, or package-private (no modifier).
4.Not allowed: static, final, abstract, or native.
5.You can declare more than one constructor in a class (constructor overloading).
6.Each constructor must have a different parameter list.

3. Types of Constructors in Java
Default Constructor
User-defined Constructor

1. Default Constructor
Definition:

If a constructor is not explicitly declared in a Java class, the compiler automatically provides one during compilation.
This automatically provided constructor is called the default constructor.

Rules of the Default Constructor:

1.It is always a zero-argument (no-parameter) constructor.
2.The access modifier of the default constructor is the same as that of the class.
3.It contains an implicit call to super(), which invokes the parent class constructor.
4.It initializes the object with default values:

Numeric types → 0
Boolean → false
Object references → null

The default constructor is provided only if no other constructor is declared by the programmer.

2. User-defined Constructor

If a programmer explicitly declares a constructor in a Java class, it is called a user-defined constructor.
User-defined constructors can be of two types:

Zero-parameter constructor (also called no-argument constructor)
→ Declared explicitly by the programmer without parameters.

Parameterized constructor
→ Declared with one or more parameters to initialize object variables with custom values.

3.Constructor Chaining in Java

Definition:

Constructor Chaining in Java is the process of calling one constructor from another constructor within
the same class or from the parent class.
It helps to reuse constructor code and avoid duplication when multiple constructors perform similar initialization tasks.

4. Purpose of Constructor Chaining

To reuse code between multiple constructors.
To initialize an object in multiple ways using constructor overloading.
To avoid redundancy by centralizing common initialization logic.

5.Types of Constructor Chaining

Constructor chaining can occur in two ways:
Within the same class (using this())
From parent class to child class (using super())

Constructor	Method
A constructor is used to **initialize the object**.	A method is used to **perform actions / business logic**.
Constructor **must have the same name as class**.	Method can have **any name**.
Constructor has **no return type** (not even void).	Method **must have a return type** (void or any
Constructor is called **automatically** when an object is created.	Methods are called **manually** using object or class name.
Constructor **cannot be static, final, abstract**.	Methods **can be static, final, abstract**, etc.

Constructor **cannot be overridden**.	Methods **can be overridden** (in inheritance).
Constructor is used **once per object creation**.	Methods can be used **multiple times** for same object.
Constructor is mainly used to **assign initial values to instance variables**.	Method is used to **perform tasks or return output**.
Constructor **cannot return any value**.	Method can **return values** using `return`
If no constructor is defined, Java provides a **default constructor**.	If no method is defined, Java does **not** provide any default method.