Module 2: Object-Oriented Design: Classes, Inheritance, and Polymorphism

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus

premanand.s@vit.ac.in

February 16, 2026

Constructor & Love

Programming (Constructor)	**Real Life (Love)**
Runs automatically when object is created	Begins automatically when relationship starts
Initializes object variables	Sets trust, care, and understanding
Ensures object is ready to use	Prepares people for a happy life
Different constructors give different initial states	Different relationships have different beginnings
Without constructor, object is incomplete	Without love, life feels empty

Key Insight

Love is the constructor of a happy life.

Why Do We Need Constructors?

- Objects need initial values
- Prevent uninitialized or invalid states
- Automatic execution during object creation

Key Idea

Constructor initializes the object at birth.

What is a Constructor?

Definition

A constructor is a special method used to initialize objects.

- Name same as class name
- No return type (not even void)
- Automatically invoked when object is created

Basic Constructor Syntax

```
class Student {
 int id;

 Student() {
  id = 0;
 }
}
```

- Called automatically
- Executes once per object

Default Constructor

- Provided by Java if no constructor is written
- Initializes variables with default values

```
class Test {
 int x;
}
```

Equivalent to:

```
Test() {
 x = 0;
}
```

User-Defined Constructor

```
class Student {
 int id;

 Student() {
  id = 100;
 }
}
```

- Overrides default constructor
- Java will NOT provide default constructor now

Parameterized Constructor

```
class Student {
 int id;
 String name;

 Student(int i, String n) {
  id = i;
  name = n;
 }
}
```

- Allows custom initialization
- Improves object flexibility

Constructor Overloading

```
class Student {
 int id;
 String name;

 Student() {
  id = 0;
  name = "NA";
 }

 Student(int i, String n) {
  id = i;
  name = n;
 }  }

Student s1 = new Student();
Student s2 = new Student(101, "Arun");
```

Constructor vs Method

- Constructor name = class name
- No return type
- Called automatically
- Used only for initialization

Constructor and Array of Objects

```
Student[] arr = new Student[3];

for(int i = 0; i < arr.length; i++) {
 arr[i] = new Student(i+1, "Student");
}
```

- Constructor initializes each object
- Clean and safe coding style

Key Rules of Constructors

- Constructor cannot be static
- Constructor cannot be abstract
- Constructor is not inherited
- Constructor can be overloaded

Common Mistakes

- Writing return type for constructor
- Assuming constructor is inherited
- Forgetting object creation using new

What is `this`?

Definition

`this` is a reference variable that points to the current object.

- Used inside non-static methods
- Refers to the object that invoked the method

Problem Without this

```
class Student {
 int id;

 Student(int id) {
  id = id;   // Problem!
 }
}
```

- Local parameter hides instance variable
- Instance variable remains uninitialized

Solution Using this

```
class Student {
 int id;

 Student(int id) {
  this.id = id;
 }
}
```

Explanation

`this.id` → instance variable `id` → constructor parameter

Using `this` to Access Instance Variables

```
void setId(int id) {
 this.id = id;
}
```

- Improves code readability
- Standard industry practice

Constructor Chaining Using this()

```
class Student {
 int id;
 String name;

 Student() {
  this(0, "NA");
 }

 Student(int id, String name) {
  this.id = id;
  this.name = name;
 }
}
```

Rule

this() must be the first statement in constructor.

Passing Current Object Using this

```
class Test {
 void show(Test t) {
  System.out.println("Method called");
 }

 void call() {
  show(this);
 }
}
```

- Current object passed as argument
- Used in callbacks and frameworks

Returning Current Object

```
class Sample {
 Sample getObject() {
  return this;
 }
}
```

- Enables method chaining
- Common in builder pattern

`this` vs Object Reference

```
Student s1 = new Student(10);
```

Inside constructor:

- `this` $\rightarrow$ same object as `s1`
- Reference names may differ

Restrictions of `this`

- Cannot be used in static context
- Cannot refer to class itself
- Cannot be reassigned

Common Mistakes

- Using `this` inside static method
- Writing `this = obj`
- Calling `this()` after statements

Thank You!

Stay Connected

Premanand S

Email: premanand.s@vit.ac.in
Phone: +91-7358679961

LinkedIn: linkedin.com/in/premsanand
Instagram: instagram.com/premsanand
WhatsApp Channel: anandsDataX

Google Scholar: Google Scholar Profile
GitHub: github.com/anandprems