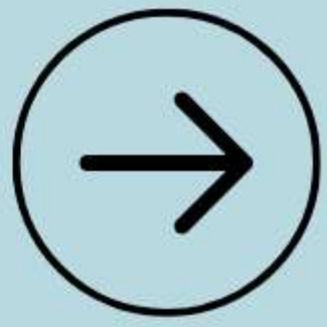
Dasari Vamsi

Java Developer

# Java Sealed Classes

Controlling Inheritance with Precision

dasarivamsi.netlify.app



Dasari Vamsi

Java Developer

# Too Many Extensions! 😕

Traditional classes can be extended by anyone (unless marked final).

This can lead to:

- Unwanted hierarchies
- Fragile designs
- Security concerns

01

dasarivamsi.netlify.app

Dasari Vamsi

Java Developer

# Enter Sealed Classes (Java 17) ✨

```
public sealed class Shape
    permits Circle, Rectangle {}
```

Only the permitted classes can extend Shape.

02

dasarivamsi.netlify.app

Dasari Vamsi

Java Developer

# Sealed Class Example 📦

```
sealed class Vehicle
    permits Car, Bike {}

final class Car extends Vehicle {}
final class Bike extends Vehicle {}
```

👉 No other class can extend Vehicle

Dasari Vamsi

Java Developer

# Why Use Sealed Classes? ⭐

- Restrict inheritance to known types
- Improve API safety
- Enhance domain modeling
- Works great with pattern matching

Dasari Vamsi

Java Developer

# Final vs Abstract vs Sealed ⚖️

- final: No one can extend
- abstract: Anyone can extend
- sealed: Only specific classes can extend

# ✓ Quick Recap

- Sealed = controlled inheritance
- Improves design & security
- Best for domain-driven models

💬 Have you tried Sealed Classes in your projects yet?

Dasari Vamsi

Java Developer

# If you find this helpful, like and share it with your friends

**Let's Connected**

dasarivamsi.netlify.app