Pattern Matching in Java

Cleaner conditionals, safer code
(Java 16 → Java 21)

ALAN BIJU

The Problem

Old-style type checks are ugly

```
if (obj instanceof String) {
    String s = (String) obj;
    ...
}
```

❌ Manual casting
❌ Verbose
❌ Easy to break

Pattern Matching with instanceof

Casting is automatic

```
if (obj instanceof String s) {
    System.out.println(s.toUpperCase());
}
```

✔️ No explicit cast
✔️ Type-safe
✔️ More readable

Pattern Matching + Conditions

Scoped & safe variables

```
if (obj instanceof Integer i && i > 0) {
    System.out.println("Positive: " + i);
}
```

💡 Variable i is only available when condition is true

Pattern Matching with switch (Java 21)

Match by type

```
static String process(Object obj) {
    return switch (obj) {
        case String s -> "Text: " + s;
        case Integer i -> "Number: " + i;
        case null -> "No value";
        default -> "Unknown";
    };
}
```

✔ No casting
✔ Handles null
✔ Cleaner branching

Spring Boot: Real-World Example

Request processing logic

```
public String handle(Object request) {
    return switch (request) {
        case CreateUserRequest r -> "Create user";
        case UpdateUserRequest r -> "Update user";
        case DeleteUserRequest r -> "Delete user";
        default -> "Invalid request";
    };
}
```

✔ Perfect for command handlers
✔ Replaces long if-else

Pattern Matching + Sealed Classes

No default needed

```
sealed interface Command
    permits CreateCmd, UpdateCmd {}

return switch (cmd) {
    case CreateCmd c -> "Create";
    case UpdateCmd u -> "Update";
};
```

💡 Variable i is only available when condition is true

Pattern Matching helps you:

✔️ Remove boilerplate
✔️ Improve readability
✔️ Write safer business logic

> 👉 If you’re checking types — use pattern matching

Stop checking types the old way. Java can do it better.

IF YOU FIND THIS HELPFUL, LIKE AND SHARE IT WITH YOUR FRIENDS

ALAN BIJU | @alanbiju98