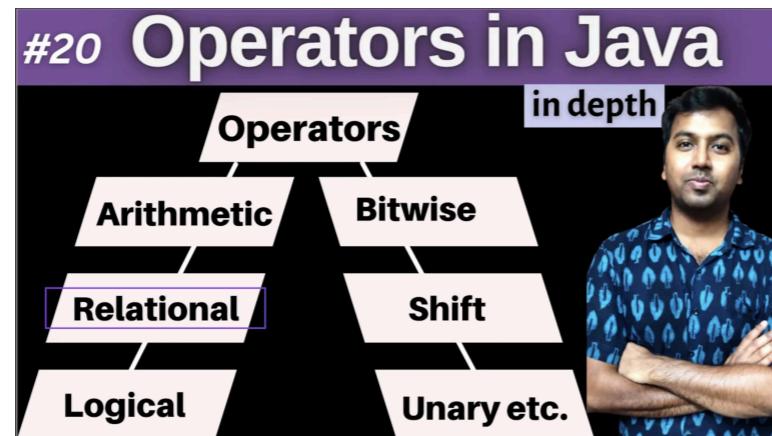


Java 16: Pattern Matching for instanceof operator

Thursday, 4 September 2025 12:20 PM

We already covered "*instanceof*" operator:



```
Object obj = "hello world";

if(obj instanceof String) {

    String s = (String) obj; //type casting
    System.out.println(s.toUpperCase());
}
```

With old way, there is an extra overhead of doing this typecasting.

Which is now resolved through this Java 16 pattern matching feature.

```
Object obj = "hello world";
```

```
if(obj instanceof String s) { //1. first comparison happens.  
| System.out.println(s.toUpperCase()); //2. if True, then s is automatically type cast and initialized  
}
```

- 's' only exist inside the block only.

```
Object obj = "hello world";  
  
if (obj instanceof String s) {  
| System.out.println(s.toUpperCase()); // it can further pass to different methods and valid till its within its scope  
}  
System.out.println(s); //invalid, s is not visible here
```

- Compiler ensures the type safety.

```
Object obj = "hello world";  
  
if (obj instanceof String s) {  
| System.out.println(s.toUpperCase());  
}  
else if (obj instanceof Integer i) {  
| System.out.println(i);  
}  
else if (obj instanceof User u) {  
| System.out.println(u.age());  
}
```

- We can combine it with '&&' operator

```
Object obj = 5;

if (obj instanceof Integer i && i < 10) {
    System.out.println(i);
}
```

- But pattern matching does not work in combination with '| |' operator

```
Object obj = 5;

if (obj instanceof Integer i || obj instanceof String s) { //i or s is not initialized because obj could be either Integer or String, there is no guarantee.
    System.out.println("obj could be either Integer or String: " + i);
}
```

- pattern matching can also works with Interfaces

```
public interface Vehicle {

    public void drive();
}
```



```
public class TwoWheeler implements Vehicle{

    @Override
    public void drive() {
        System.out.println("Two wheeler drive implementation");
    }
}
```

```
public class FourWheeler implements Vehicle{

    @Override
    public void drive() {
        System.out.println("four wheeler drive implementation");
    }
}
```

{
}{
}

Without Pattern Matching:

```
Object obj = new TwoWheeler();

if (obj instanceof TwoWheeler) {
    TwoWheeler twoWheelerObj = (TwoWheeler)obj;
    twoWheelerObj.drive();
}

else if (obj instanceof FourWheeler) {

    FourWheeler fourWheelerObj = (FourWheeler)obj;
    fourWheelerObj.drive();
}
```

With Pattern Matching:

```
Object obj = new TwoWheeler();

if(obj instanceof Vehicle vehicleObj) {
    vehicleObj.drive();
}
```

Output for both:

```
/Library/Java/JavaVirtualMachines/jdk-1
Two wheeler drive implementation

Process finished with exit code 0
```