01/09

How and When to Use Record Classes in Java

SWIPE

02/09

What is a Record Class?

A record class in Java is a special type of class that is concise and designed to hold immutable data. Unlike traditional classes, records automatically generate many commonly used methods, such as:

- Getters for all fields
- equals()
- hashCode()
- toString()

KEEP SWIPING

03/09

Creating a Record Class

```
public record Person(String name, int age) {
}
```

This simple declaration:

- Defines a Person class with name and age fields.
- Automatically generates the constructor, getters (name() and age()), equals(), hashCode(), and toString() methods.

KEEP SWIPING

How to Use Record Classes

```java
Person person = new Person("Alice", 30);
System.out.println(person.name()); // Alice
System.out.println(person.age());  // 30
System.out.println(person);        //
Person[name=Alice, age=30]
```

Records come with pre-defined methods:

- Getters: name() and age()
- toString(): Provides a string representation of the record
- equals() and hashCode(): Implements value-based equality

KEEP SWIPING

05/09

When to Use Record Classes

1. Immutable Data Carriers
Records are ideal for classes that primarily serve to hold data:

- DTOs (Data Transfer Objects)
- Value Objects
- Configuration Settings

2. Simplifying Code
When you need a simple class to hold data and don't want to write boilerplate code, records are a great fit.

KEEP SWIPING

06/09

When Not to Use Record Classes

1. Mutable Data
If you need mutable fields, a traditional class is more suitable since record fields are implicitly final.

2. Complex Behavior
If your class requires complex logic, inheritance, or behavior beyond holding data, use a traditional class.

KEEP SWIPING

07/09

Examples of Record Classes in Action

```java
// Example 1: Simple Data Carrier

public record Point(int x, int y) {
}

Point point = new Point(10, 20);
System.out.println(point); // Point[x=10, y=20]
```

KEEP SWIPING

BFF

08/09

Examples of Record Classes in Action

```
// Example 2: DTO for API Response

public record ApiResponse(int statusCode, String
message) {
}

ApiResponse response = new ApiResponse(200, "Success");
System.out.println(response);
// ApiResponse[statusCode=200, message=Success]
```

KEEP SWIPING

09/09

Have you used record classes in Java?

COMMENT BELOW