CASE CLASS
&
PATTERN MATCHING

Normal Class

- **class Person(val name: String, var age: Int)**

In simple words, it's a way to group related information and functions into one structure that can be used to create multiple objects of the same type.

Case Class

- **case class** Person**(name: String, age: Int)**

A **case class** in Scala is a special type of class that comes with some built-in features that make it more convenient for working with immutable data.

Difference Between Normal Class and Case Class

Feature	Normal Class	Case Class
Immutability	Data is mutable by default (you can change values).	Data is immutable by default (values cannot be changed).
Methods	You need to write custom methods for equals, hashCode, etc.	Automatically provides methods like equals, hashCode, toString.
Copying	No built-in method to copy and change fields.	Provides a copy() method to create modified copies easily.
Pattern Matching	Cannot be used directly in pattern matching.	Designed for pattern matching and destructuring data.
Default Constructor	You have to define the constructor manually.	Automatically generates a constructor.
Inheritance	Can inherit from other classes or traits.	Case classes can inherit from traits, but not from other case classes.

Pattern Matching

- Pattern matching is a powerful feature in Scala that allows you to match complex data structures, such as case classes, tuples, or lists, against patterns. It’s similar to switch/case statements in other languages but more flexible and expressive.

Benefits of pattern matching:

- Cleaner Code
- Easy Data Extraction
- Multiple Conditions
- Works with Complex Data
- Default Case Handling

General Syntax:

- **value match {**

```
    case pattern1 => result1
    case pattern2 => result2
    case _ => defaultResult // Wildcard to match anything
}
```

Let's see some examples of pattern matching in scala.

In **Scala**, pattern matching is commonly used with **case classes**, but it works with many other types too.

THANKYOU…