**Data Class in Kotlin**

In Kotlin, a data class is a special type of class that is primarily used to hold data. It's concise and automatically generates useful methods like `equals()`, `hashCode()`, `toString()`, and `copy()`. These methods are generated based on the properties defined in the primary constructor of the data class.Data classes are a handy feature in Kotlin for creating classes whose main purpose is to hold data, reducing boilerplate code and making your code cleaner and more concise.

Here's a basic example of a data class in Kotlin:

data class Person(val name: String, val age: Int)

In this example, `Person` is a data class with two properties: `name` and `age`. Since it's a data class, Kotlin automatically generates the following methods for you:

- `equals()`: Compares the contents of two objects.

- `hashCode()`: Calculates a hash code based on the contents of the object.

- `toString()`: Generates a string representation of the object.

- `copy()`: Creates a copy of the object with optional modifications.

Data classes can also be used in destructuring declarations, making it easy to extract properties:

val person = Person("Alice", 30)

val (name, age) = person

println("Name: $name, Age: $age")

This will print "Name: Alice, Age: 30".

**Difference between class and data class**

In Kotlin, both `class` and `data class` are used to define custom types, but they serve slightly different purposes and have different behaviors.

1. **Class :**

- A regular class in Kotlin can have properties, methods, and other features like inheritance, interfaces, etc.

- By default, it doesn't provide implementations for `equals()`, `hashCode()`, `toString()`, or `copy()` methods unless explicitly overridden.

- You need to manually implement these methods if you want the class to support value-based equality checks, data copying, and string representation in a specific way.

Example:

class Person(val name: String, val age: Int) {

// Other methods and properties can be defined here

}

1. **Data class :**

- A data class is a special type of class specifically designed to hold data.

- It automatically generates `equals()`, `hashCode()`, `toString()`, and `copy()` methods based on the properties defined in the primary constructor.

- It's primarily used for immutable data-holding classes where equality checks, copying, and string representation are essential.

- Data classes cannot be declared as `open` (inheritable) by default.

Example:

data class Person(val name: String, val age: Int)