**Dart – Day12**

**Emp-id : 4781**

## **Null Assertion Operator (!)**

The null assertion operator is used to force Dart to treat a nullable value as non-null.  
If the value is actually null, it will throw a runtime error.

**Example:**

class Data

{

String name;

String? city;

Data(this.name, this.city);

void displaydata()

{

if(city != null)

{

print("Hi ${name}, your city is ${city!.toUpperCase()}.");

}

else

{

print("Hi ${name}, your city is not mentioned.");

}

}

}

void main()

{

Data d1 = Data("Chandini","Bengaluru");

d1.displaydata();

Data d2 = Data("Sneha",null);

d2.displaydata();

}

## **Null-Aware Operators (?.)**

Null-aware operators help us work with nullable variables without throwing errors.  
If the variable is null, the expression returns null instead of throwing an error.

**Example:**

class Person

{

String? nickname;

void showNickname()

{

print("Your nickname in uppercase: ${nickname?.toUpperCase()}");

}

}

void main()

{

Person p1 = Person();

p1.nickname = "Chandu";

p1.showNickname();

Person p2 = Person();

p2.nickname = null;

p2.showNickname();

}

## **Late Variables**

In Dart, late means a variable will be initialized later, but before it is used.

* Without late, Dart forces you to initialize non-nullable variables immediately.
* With late, you *promise* the compiler you’ll assign a value before using it.

### **Example :**

class Student

{

late String name;

late int age;

void setDetails(String n, int a)

{

name = n;

age = a;

}

void display()

{

print("Name: $name, Age: $age");

}

}

void main()

{

Student s = Student();

s.setDetails("Chandini", 21);

s.display();

}

## **Private Constructor**

A private constructor in Dart is created by prefixing the constructor name with \_.  
This is usually used to restrict object creation from outside the class.

**Example :**

class Student

{

String? name;

int? age;

Student.\_(this.name, this.age);

static Student register(String name, int age)

{

if (age < 5)

{

throw Exception("Age must be at least 5 to register");

}

return Student.\_(name, age);

}

void display()

{

print("Student: $name, Age: $age");

}

}

void main()

{

var s1 = Student.register("Chandini", 21);

s1.display();

}

## **Factory Constructor**

A factory constructor in Dart does not always create a new instance.  
It can return existing objects, private constructor or apply custom logic before object creation.

**Example :**

class Employee

{

String name;

double salary;

Employee.\_(this.name, this.salary);

factory Employee(String name, String type)

{

if (type == "fulltime") {

return Employee.\_(name, 50000);

} else if (type == "parttime") {

return Employee.\_(name, 20000);

} else {

throw Exception("Invalid employee type");

}

}

void showDetails()

{

print("Employee: $name, Salary: $salary");

}

}

void main() {

var e1 = Employee("Chandini", "fulltime");

e1.showDetails();

var e2 = Employee("Sneha", "parttime");

e2.showDetails();

}

* **this keyword**

this refers to the current object of the class.

Used to access instance variables, methods, and to enable method chaining.

## **1. Using this to Differentiate Instance & Local Variables**

When local variables shadow instance variables, use this to clarify.

**Example :**

class Student

{

String name = "";

int age = 0;

Student(String name, int age)

{

this.name = name; // 'this' refers to instance variables

this.age = age;

}

void display()

{

print("Name: $name, Age: $age");

}

}

void main()

{

var s1 = Student("Chandini", 21);

s1.display();

}

## **2. Using this in Constructor Short Syntax**

Dart allows you to skip explicit assignments:

**Example :**

class Employee

{

String name;

int id;

Employee(this.name, this.id);

}

void main()

{

var e = Employee("Anil", 101);

print("${e.name}, ${e.id}");

}

## **3. Using this to Call Current Class Methods**

**Example :**

class Car

{

void start()

{

print("Car started");

}

void run()

{

this.start();

print("Car is running");

}

}

void main()

{

var c = Car();

c.run();

}

## **4. Method Chaining with this**

Return this from methods to chain multiple calls.

**Example :**

class Calculator

{

int value = 0;

Calculator add(int n)

{

value += n;

return this;

}

Calculator multiply(int n)

{

value \*= n;

return this;

}

}

void main()

{

var calc = Calculator();

calc.add(10).multiply(2);

print(calc.value);

}

**5. Named constructor using this keyword**

In Dart, this inside a constructor is used to assign parameters to instance variables, while : this() in a named constructor redirects to another constructor of the same class.

**Example :**

class Student

{

String name;

int age;

Student(this.name, this.age);

Student.named(String name) : this(name, 18);

void show()

{

print("Name: $name, Age: $age");

}

}

void main() {

var s1 = Student("Chandini", 21);

s1.show();

var s2 = Student.named("Sneha");

s2.show();

}