**Dart – Day7**

**Emp-id : 4781**

### **Abstract Class**

An abstract class is a class that cannot be instantiated directly. It is used when you want to provide common functionality along with some methods that must be implemented by child classes.

* Can have abstract methods (without body).
* Can also have concrete methods (with body).
* Classes extend an abstract class.

**Example:**

abstract class Payment

{  
 void pay(); // abstract method  
  
 void receipt()

{ // concrete method  
 print("Payment receipt generated");  
 }  
}  
  
class UpiPayment extends Payment

{  
 @override  
 void pay()

{  
 print("Payment done using UPI");  
 }  
}  
  
void main()

{  
 var upi = UpiPayment();  
 upi.pay();  
 upi.receipt();  
}

### **Interface**

In Dart, any class can be used as an interface. A class uses implements to follow the structure of another class.

* All methods of the interface must be overridden.
* Cannot have default implementation carried over.
* Used when you just need a contract (rules to follow).

**Example:**

class Printer

{  
 void printData() {}  
}  
  
class Scanner

{  
 void scanData() {}  
}  
  
class OfficeMachine implements Printer, Scanner

{  
 @override  
 void printData()

{  
 print("Office Machine is printing...");  
 }  
  
 @override  
 void scanData()

{  
 print("Office Machine is scanning...");  
 }  
}  
  
void main()

{  
 var machine = OfficeMachine();  
 machine.printData();  
 machine.scanData();  
}

### **Example: Abstract Class + Interface in One Program**

// Abstract class  
abstract class Employee

{  
 String name;  
 Employee(this.name);  
  
 void work(); // abstract method  
  
 void showDetails()

{  
 print("Employee Name: $name"); // concrete method  
 }  
}  
  
// Interface (in Dart, any class can act as an interface)  
class Report

{  
 void generateReport() {}  
}  
  
// Class using both abstract class and interface  
class Manager extends Employee implements Report

{  
 String department;  
  
 Manager(String name, this.department) : super(name);  
  
 @override  
 void work()

{  
 print("$name manages the $department department");  
 }  
  
 @override  
 void generateReport()

{  
 print("$name is generating the performance report");  
 }  
}  
  
void main()

{  
 var m = Manager("Chandini", "HR");  
 m.showDetails(); // from abstract class  
 m.work(); // abstract method implemented  
 m.generateReport(); // interface method implemented  
}

### **Difference Between Abstract Class and Interface**

1. **Purpose**
   1. **Abstract Class**: Used when you want to provide base functionality plus enforce some rules.
      1. Example: Employee has showDetails() (concrete) + work() (abstract).
   2. **Interface**: Used only to enforce a contract that a class must follow.
      1. Example: Report forces generateReport() to be implemented.
2. **Methods**
   1. **Abstract Class**: Can have both abstract methods and concrete methods.
   2. **Interface**: All methods must be implemented (no default implementation).
3. **Inheritance vs Implementation**
   1. **Abstract Class**: Classes extend an abstract class (only one).
   2. **Interface**: Classes implement interfaces (multiple can be implemented).
4. **Reusability**
   1. **Abstract Class**: Promotes reusability since subclasses can reuse concrete methods.
      1. Example: Manager reused showDetails() from Employee.
   2. **Interface**: No reusability — every implementing class must write its own method.
      1. Example: Manager had to define its own generateReport().