Dart – Day8

Emp-id : 4781

- **Mixin**

A mixin is a way to reuse code in multiple classes without using inheritance. In Dart, mixins are created using the mixin keyword or a normal class without a constructor. Classes can use a mixin with the with keyword.

Example:

```
mixin Logger
{
  void log(String message)
  {
    print("Log: $message");
  }
}

class FileManager with Logger
{
  void saveFile()
  {
    log("File saved successfully");
  }
}

void main()
{
  var fm = FileManager();
  fm.saveFile();
}
```

- ## on Keyword in Mixins

The on keyword is used in a mixin to restrict which classes can use it. It ensures the mixin can only be applied to certain types of classes.

Example:

```
class Database
{
  void connect()
  {
    print("Database connected");
  }
}

mixin Query on Database
{
  void runQuery()
  {
    connect(); // safe because mixin is restricted to Database
    print("Query executed");
  }
}

class MyDatabase extends Database with Query {}

void main()
{
  var db = MyDatabase();
  db.runQuery();
}
```

- ## Deadly Diamond of Death

The Deadly Diamond of Death is a problem in multiple inheritance when a class inherits from two classes that both inherit from the same base class. Dart avoids this issue because it does not allow multiple inheritance — instead, it uses mixins to share code safely.

Example (concept only, not valid in Dart):

```
 A
/ \
B   C
\ /
 D
```

- If B and C both override a method from A, class D won't know which version to use.
- Dart prevents this by not allowing multiple extends. Instead, you use mixins.

- **Interface Segregation Principle (ISP)**

The Interface Segregation Principle says that a class should not be forced to implement methods it does not use. Instead of having one large interface, we split it into smaller, more specific ones.

Example:

```
class Printer
{
  void printDoc();
}

class Scanner
{
  void scanDoc();
}

// Good design: separate interfaces
class OfficeMachine implements Printer, Scanner
{
  @override
  void printDoc()
  {
    print("Printing document...");
  }

  @override
  void scanDoc()
{
    print("Scanning document...");
  }
}
```

Here, Printer and Scanner are small interfaces, so a class only implements what it needs.