Dart – Day14

Emp-id : 4781

- **Mixin**

A mixin is like reusable code you can add to multiple classes. It is like a class that contain a set of methods and properties that can be added to another class.

→ Adding the functionality of one class to another class(without using traditional inheritance).
→ Mixin is defined using with keyword and to declare a class as mixin we use mixin keyword.

Example :

```
mixin Walk
{
  void walk() => print("Walking...");
}
mixin Swim
{
  void swim() => print("Swimming...");
}
class Human with Walk, Swim {}
class Fish with Swim {}
void main()
{
  var h = Human();
  h.walk();
  h.swim();
```

```
  var f = Fish();
  f.swim();
}
```

- **on Keyword**

The on keyword restricts a mixin so it can only be applied to specific classes.

→ The mixin can be applied to the class which extends that class(subclass).
→ The particular mixin will be applied only to particular classes.

Example :

```
class Vehicle
{
  void move() => print("Vehicle is moving");
}
mixin Electric on Vehicle
{
  void charge() => print("Charging battery...");
}
class Car extends Vehicle with Electric {}
class Bike {}
void main()
{
  var c = Car();
  c.move();
  c.charge();
```

```
}
```

- **Mixin using on keyword :**

Example :

```
abstract class Performer
{
  void perform();
}
abstract class Hero
{
  void action();
}
mixin Dancer on Performer
{
  @override
  void perform()
  {
    print("dance");
  }
}
mixin Singer on Hero
{
  void perform()
  {
    print("Singing");
  }
```

```
  void action()
  {
    print("Acting");
  }
}
mixin Producer on Hero
{
  void produce()
  {
    print("Producing");
  }
}
mixin Director
{
  void direct()
  {
    print("Directing");
  }
}
class Actor extends Hero with Singer, Producer, Director
{
  void display()
  {
    perform();
  }
}
class Actress extends Hero
```

```
{
  void action()
  {
    print("Actress acting");
  }
}
void main()
{
  Actor a = Actor();
  a.perform();
  a.action();
  a.display();
  a.produce();
  a.direct();
  Actress s = Actress();
  s.action();
}
```

- **Static Variables**

A static variable belongs to the class, not to individual objects. Shared among all instances of the class.

→ Accessed using ClassName.variableName.
→ Memory allocation is done only once, regardless of how many instances are created.

Example :

```
class Library
{
```

```
  String bookName;
  static int totalBooks = 0;
  Library(this.bookName)
  {
    totalBooks++;
  }
  void showBook()
  {
    print("Book: $bookName");
  }
}
void main()
{
  var b1 = Library("Dart Programming");
  var b2 = Library("Flutter Development");
  b1.showBook();
  b2.showBook();
  print("Total Books in Library: ${Library.totalBooks}");
}
```

- **Static Method**

A static method can be called without creating an object. Belongs to class, not instances.

→ Accessed using ClassName.methodName().

→ Can access only static variables and static methods.
→ Cannot use this keyword.

Example :

```
class Bank
{
  String accountHolder;
  double balance;
  static double interestRate = 0.05;
  Bank(this.accountHolder, this.balance);
  static void updateInterestRate(double newRate)
  {
    interestRate = newRate;
    print("Updated Interest Rate: $interestRate");
  }
  void showDetails()
  {
    print("$accountHolder - Balance: $balance, Interest Rate: $interestRate");
  }
}
void main()
{
  var acc1 = Bank("Chandini", 10000);
  var acc2 = Bank("Sneha", 20000);
```

```
  acc1.showDetails();
  acc2.showDetails();
  Bank.updateInterestRate(0.07);
  acc1.showDetails();
  acc2.showDetails();
}
```