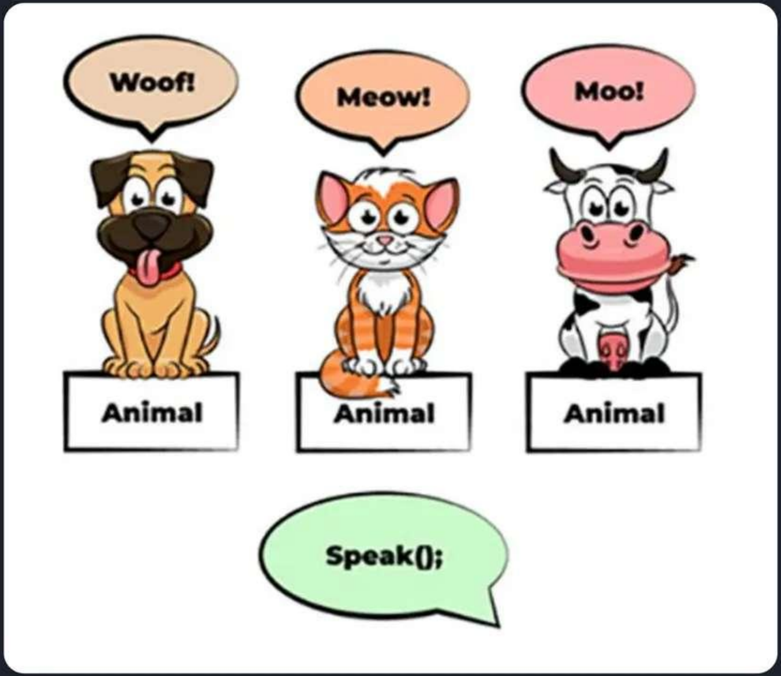
Dart Programming Language 2

By : Asst.Lect Mohammad Baqer Haleem

Polymorphism

Polymorphism is a concept in object-oriented programming that allows objects of different classes to be treated as if they were of the same type. In other words, it is the ability of an object to take on many forms.



Polymorphism

```
class Animal {
 void makeSound() {
   print('The animal makes a sound');
 }
}

class Dog extends Animal {
 void makeSound() {
   print('The dog barks');
 }
}

class Cat extends Animal {
 void makeSound() {
   print('The cat meows');
 }
}

void main() {
 Animal animal = new Animal();
 Animal dog = new Dog();
 Animal cat = new Cat();

 animal.makeSound();
 dog.makeSound();
 cat.makeSound();
}
```

Abstract

In programming, the term "abstract" refers to the concept of defining common interface or behavior without implementing it in detail. In object-oriented programming, an abstract class is a class that cannot be instantiated directly, but can be used as a base class for other classes.

Abstract

```
abstract class Shape {
 double area();
}
class Circle extends Shape {
 double radius;

 Circle(this.radius);

 double area() {
   return 3.14 * radius * radius;
 }
}

class Rectangle extends Shape {
 double width, height;

 Rectangle(this.width, this.height);

 double area() {
   return width * height;
 }
}

void main() {
 Circle circle = Circle(5);
 Rectangle rectangle = Rectangle(5, 10);

 print(circle.area());
 print(rectangle.area());
}
```

Interface

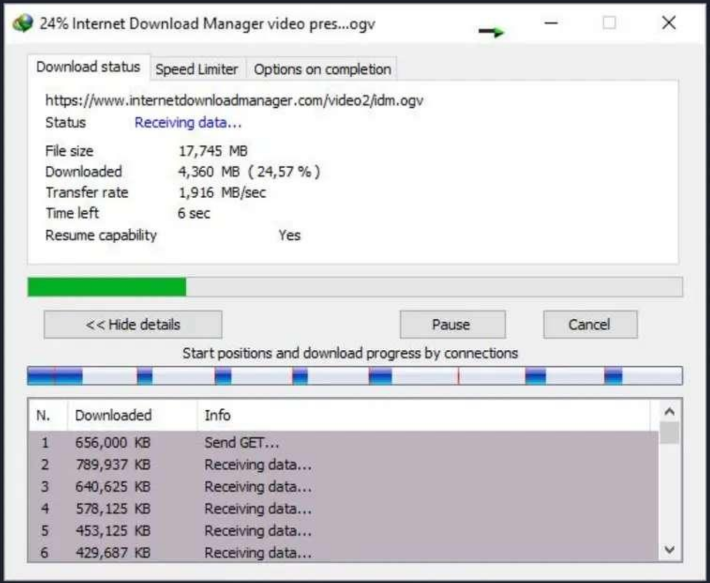
In programming, an interface is a specification of a set of methods and properties that a class must implement. An interface does not provide any implementation details, but rather defines a contract or agreement between different classes that share a common set of behaviors.

Interface

```
class CountryRule {
 void noGun() {
   print('No gun');
 }
}

class SchoolRule {
 void noLongHair() {
   print('No long hair');
 }
}

class Student implements CountryRule, SchoolRule {
 @override
 void noGun() {
   print('No gun');
 }

 @override
 void noLongHair() {
   print('No long hair');
 }
}

void main() {
 Student student = Student();
 student.noGun();
 student.noLongHair();
}
```

I. Asynchronous

Asynchronous programming is a programming paradigm that allows tasks to be executed concurrently and independently of each other, without having to wait for each task to complete before moving on to the next one. In other words, it enables a program to perform multiple operations at the same time, which can greatly improve its performance and responsiveness.



24% Internet Download Manager video pres...ogv

Download status | Speed Limiter | Options on completion

```
https://www.internetdownloadmanager.com/video2/idm.ogv
Status          Receiving data...

File size          17,745  MB
Downloaded         4,360  MB  ( 24,57 % )
Transfer rate      1,916  MB/sec
Time left          6 sec
Resume capability              Yes
```

<< Hide details | Pause | Cancel

Start positions and download progress by connections

N.	Downloaded	Info
1	656,000 KB	Send GET...
2	789,937 KB	Receiving data...
3	640,625 KB	Receiving data...
4	578,125 KB	Receiving data...
5	453,125 KB	Receiving data...
6	429,687 KB	Receiving data...

VIII. Asynchronous

```
Future<void> printNumbers() async {
 print('Starting to print numbers...');

 // wait for 1 second before printing each number
 for (int i = 1; i <= 5; i++) {
   await Future.delayed(Duration(seconds: 1));
   print(i);
 }

 print('Finished printing numbers.');
}

void main() {
 printNumbers();
 print('Continuing to execute other tasks...');
}
```

THANK YOU