**OOP IN DART**

Object-oriented programming (OOP) is a programming method that uses objects and their interactions to design and program applications. It is one of the most popular programming paradigms and is used in many programming languages, such as Dart, Java, C++, Python, etc.

In OOP, an object can be anything, such as a person, a bank account, a car, or a house. Each object has its attributes (or properties) and behavior (or methods). For example, a person object may have the attributes name, age and height, and the behavior walk and talk.

**Advantages**

1. It is easy to understand and use.
2. It increases reusability and decreases complexity.
3. The productivity of programmers increases.
4. It makes the code easier to maintain, modify and debug.
5. It promotes teamwork and collaboration.
6. It reduces the repetition of code.

**Features of OOP**

1. Class
2. Object
3. Encapsulation
4. Inheritance
5. Polymorphism
6. Abstraction

**Key Points**

* Object Oriented Programming (OOP) is a programming paradigm that uses objects and their interactions to design and program applications.
* OOP is based on objects, which are data structures containing data and methods.
* OOP is a way of thinking about programming that differs from traditional procedural programming.
* OOP can make code more modular, flexible, and extensible.
* OOP can help you to understand better and solve problems.

**Classes**

In object-oriented programming, a class is a blueprint for creating objects. A class defines the properties and methods that an object will have. For example, a class called Dog might have properties like breed, color and methods like bark, run.

**Declaring A class in Dart**

In Dart programming, you can declare a class using the class keyword followed by the class name. Here's a simple example:

class Dog {

  // Properties/attributes

  String breed;

  String color;

  String origin;

  // Constructor

  Dog(this.breed, this.color,this.origin);

  // Methods/behaviours

  void bark() {

    print('Woof!');

  }

  void run() {

    print('The dog is running.');

  }

}

void main() {

  // Creating an instance of the Dog class/object

  var myDog = Dog('Labrador', 'Golden','Siberian');

  // Accessing properties/attributes/characteristics

  print('Breed: ${myDog.breed}');

  print('Color: ${myDog.color}');

  print('Origin:${myDog.origin}');

  // Calling methods

  myDog.bark();

  myDog.run();

}

**Brief explanation**

Class Declaration (class Dog { ... }):

We declare a class named Dog using the class keyword.

Properties (String breed; String color;):

Inside the Dog class, we declare two properties: breed and color, both of type String. These properties will hold information about the breed and color of the dog.

Constructor (Dog(this.breed, this.color);):

We define a constructor for the Dog class.

The constructor takes two parameters, breed and color, and assigns them to the respective properties.

This shorthand constructor syntax Dog(this.breed, this.color); automatically initializes the properties with the provided values when a Dog object is created.

Methods (void bark() { ... }, void run() { ... }):

Inside the Dog class, we define two methods:

bark(): This method prints "Woof!" to simulate the sound of a dog barking.

run(): This method prints "The dog is running." to simulate the action of a dog running.

Main Function (void main() { ... }):

We define the main() function, which serves as the entry point of the Dart program.

Inside the main() function:

We create an instance of the Dog class using the constructor: var myDog = Dog('Labrador', 'Golden');

We access the properties of the myDog instance and print out its breed and color.

We call the bark() and run() methods on the myDog instance to simulate the behaviors of the dog.

**Example with a class Person and their details**

In this example below, there is class Person with four properties: name, phone, isMarried, and age. The class also has a method called displayInfo, which prints out the values of the four properties.

class Person {

  // Properties

  String name;

  String phone;

  bool isMarried;

  int age;

  // Constructor

  Person(this.name, this.phone, this.isMarried, this.age);

  // Method to display information

  void displayInfo() {

    print('Name: $name');

    print('Phone: $phone');

    print('Marital Status: ${isMarried ? 'Married' : 'Single'}');

    print('Age: $age');

  }

}

void main() {

  // Creating an instance of the Person class

  var person = Person('John Doe', '+1234567890', true, 30);

  // Calling the displayInfo method to print information

  person.displayInfo();

}

Example with class Area:

class Area {

  // Properties

  double length;

  double width;

  // Constructor

  Area(this.length, this.width);

  // Method to calculate area

  double calculateArea() {

    return length \* width;

  }

}

void main() {

  // Creating an instance of the Area class

  var rectangle = Area(5.0, 3.0);

  // Calculating and printing the area

  print('Area of the rectangle: ${rectangle.calculateArea()} square units');

}

In this example:

* We declare a class Area with properties length and width.
* We define a constructor to initialize these properties.
* We define a method calculateArea() to compute the area of the rectangle.
* In the main() function, we create an instance of Area called rectangle with length 5.0 units and width 3.0 units.
* We then call the calculateArea() method on the rectangle object to calculate its area and print the result.

**KEY POINTS ON CLASS**

1. The class keyword is used to declare the class.
2. The class serves as an object creation template.
3. Methods and properties make up the class body.
4. The terms fields, attributes, and data members are other terms for the properties.
5. The behaviors or member functions are another name for the methods.

**OBJECTS IN DART**

Programming in Dart is object-oriented; everything is handled as an object. A variable or instance of a class that is used to access the properties of the class is called an object. State and behavior are two characteristics of an object. Let's say a man is an item with a name, age, and level of health as well as a behavior (running, walking, and sleeping). Programming objects have states and behaviors, much like real-world things do in theory.

A class is a template from which an object is produced.

Example given:

// Define a class representing an Animal

class Animal {

  // Properties

  String name;

  String species;

  // Constructor

  Animal(this.name, this.species);

  // Method to make the animal sound

  void makeSound() {

    print('$name (${species}) makes a sound.');

  }

}

void main() {

  // Creating instances (objects) of the Animal class

  var cat = Animal('Whiskers', 'Cat');

  var dog = Animal('Buddy', 'Dog');

  // Accessing properties and behaviors

  print('${cat.name} is a ${cat.species}.');

  cat.makeSound();

  print('${dog.name} is a ${dog.species}.');

  dog.makeSound();

}

In this Dart code, we have a class called Animal, which acts as a blueprint for creating objects that represent different animals. Each Animal object has properties like name and species, and it can perform actions using methods like makeSound(). In our program, we create two instances (objects) of the Animal class: one representing a cat and another representing a dog. We set specific values for the properties of each object, such as their names and species. Then, we use the objects to call the makeSound() method, simulating the animals making sounds. This example demonstrates how classes and objects work together to model real-world entities in code.