Dart Object-Oriented Concepts

- Dart is an object-oriented programming language, and it supports all the concepts of object-oriented programming such as classes, object, inheritance, mixin, and abstract classes.

- The Object-oriented programming approach is used to implement the concept like polymorphism, data-hiding, etc. **The main goal of oops is to reduce programming complexity and do several tasks simultaneously.**

Class in Dart Langauge

- Dart classes are defined as the blueprint of the associated objects. A Class is a user-defined data type that describes the characteristics and behavior of it. To get all properties of the class, we must create an object of that class. The syntax of the class is given below.

-
  ```
  class ClassName {
  //variables
  //constructor
  //named constructors
  //methods
  }
  ```

where ClassName is the name by which this class is referenced.

Dart Class with variables and methods

```
  class Car
{
  String name="";
  int miles=1;
  void printDetails()
  { print(name+' has gone '+miles.toString()+' miles.');  }

}
void main()
{
  Car car1 = Car();
  car1.name = 'Ford Mustang';
  car1.miles = 22000;
  car1.printDetails();
}
```

Constructor

- A constructor shares the syntax of a method, with same name as that of class and without any return type.
- When you create an object of a class type, you can provide the values for parameters given in a constructor.

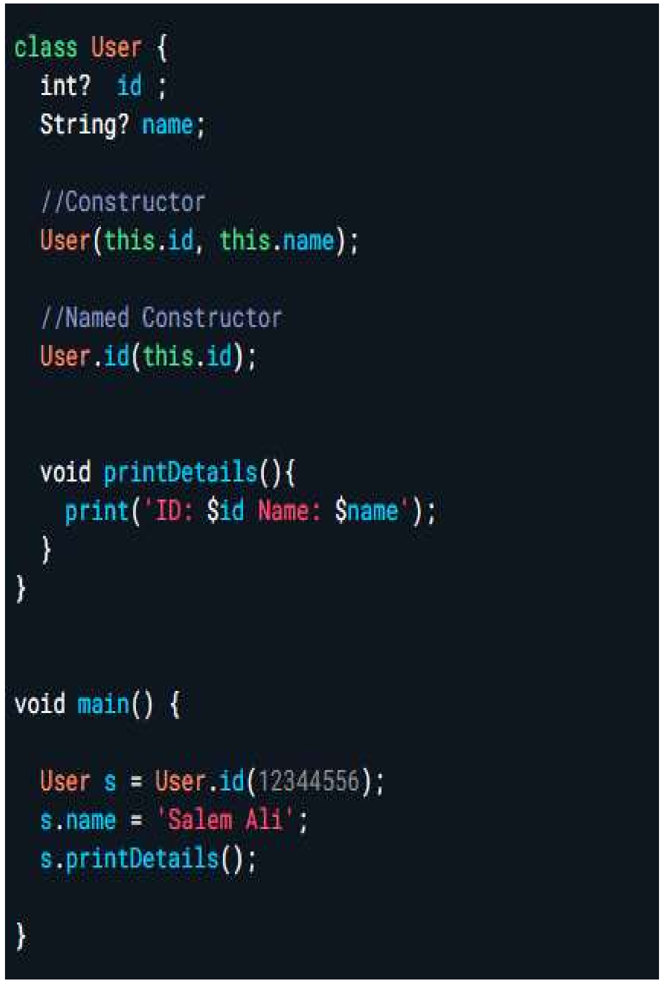
```
class Car{
String name="";    int miles=1;
  Car(name, miles) {
    this.name = name;
    this.miles = miles;
  }
void printDetails() {
    print(name+' has gone '+miles.toString()+' miles.');  }
}
void main(){
  Car car = Car('Ford Mustang', 225555320);
car.printDetails();
}
```

Named Constructors

- You can define special type of constructors called named constructors in a class.

- Named Constructor: As you can't define multiple constructors with the same name, this type of constructor is the solution to the problem. They allow the user to make multiple constructors with a different name.

```
class_name.constructor_name(parameters)
   {
   // Body of Constructor
}
```



```
class User {
  int?  id ;
  String? name;

  //Constructor
  User(this.id, this.name);

  //Named Constructor
  User.id(this.id);


  void printDetails(){
    print('ID: $id Name: $name');
  }
}


void main() {

  User s = User.id(12344556);
  s.name = 'Salem Ali';
  s.printDetails();

}
```

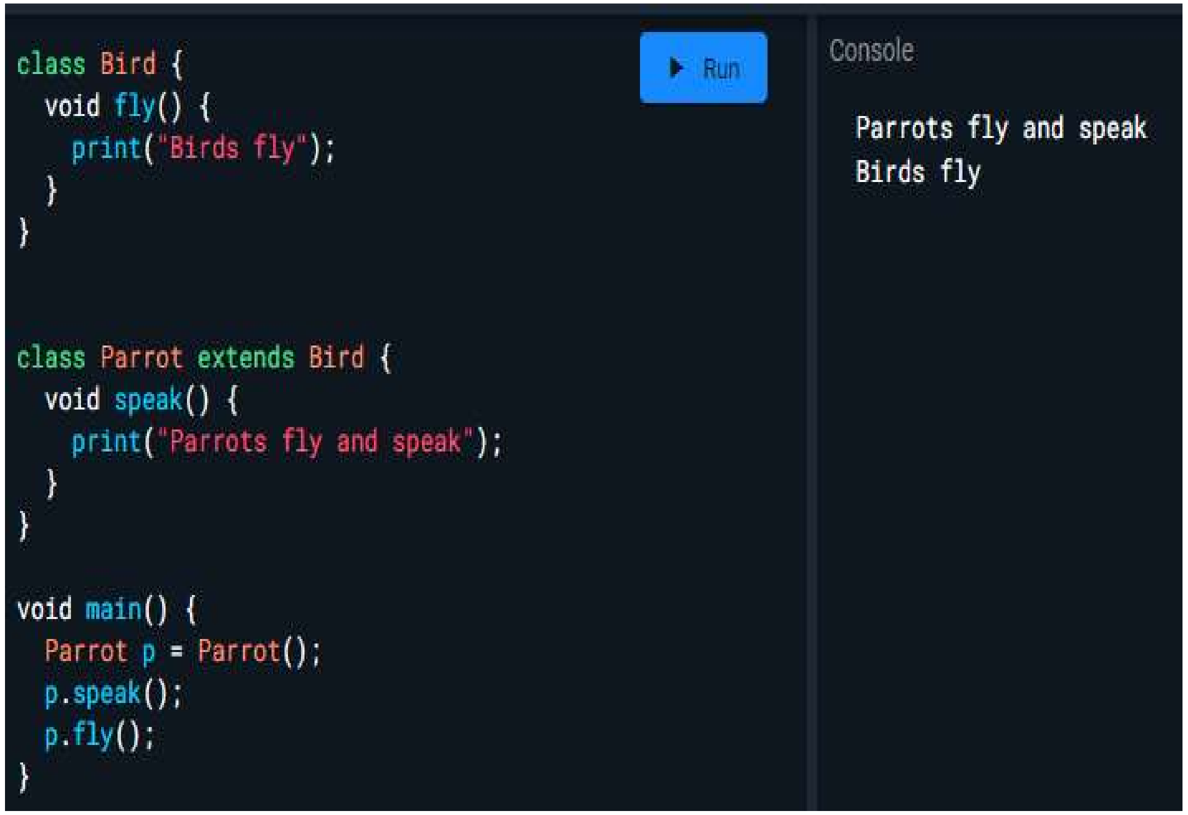
Getter and Setter Methods in Dart

Getter and setter methods are the class methods used to manipulate the data of the class fields. Getter is used to read or get the data of the class field whereas setter is used to set the data of the class field to some variable.

```
class CCC {
  String geekName = "";
  String get getName {
    return geekName;
  }

  set setName(String name) {
    geekName = name;
  }
}

void main() {
  CCC geek = CCC();
  geek.setName = "ITMC323 ";
  print("Welcome to ${geek.getName}");
}
```

Dart - Inheritance

In Dart, one class can inherit another class i.e dart can create a new CLass from an existing class. We make use of **extend** keyword to do so ..



```
class Bird {
  void fly() {
    print("Birds fly");
  }
}


class Parrot extends Bird {
  void speak() {
    print("Parrots fly and speak");
  }
}

void main() {
  Parrot p = Parrot();
  p.speak();
  p.fly();
}
```

```
Parrots fly and speak
Birds fly
```

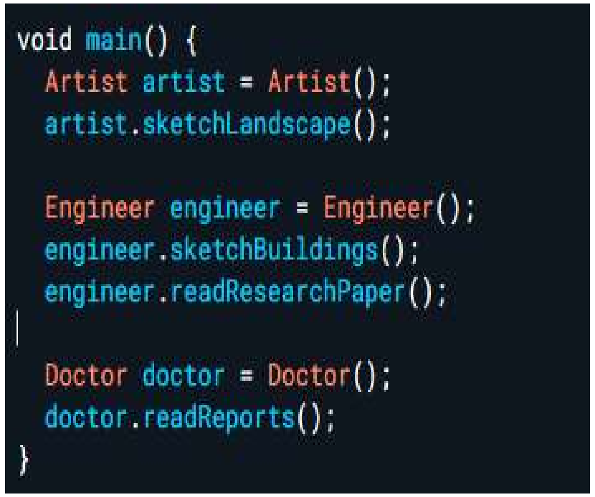
Types of Inheritance

- **Single Inheritance:** When a class inherits a single parent class than this inheritance occurs.
- **Multiple Inheritance:** When a class inherits more than one parent class than this inheritance occurs. **Dart doesn't support this.**
- **Multi-Level Inheritance:** When a class inherits another child class than this inheritance occurs.
- **Hierarchical Inheritance:** More than one classes have the same parent class.

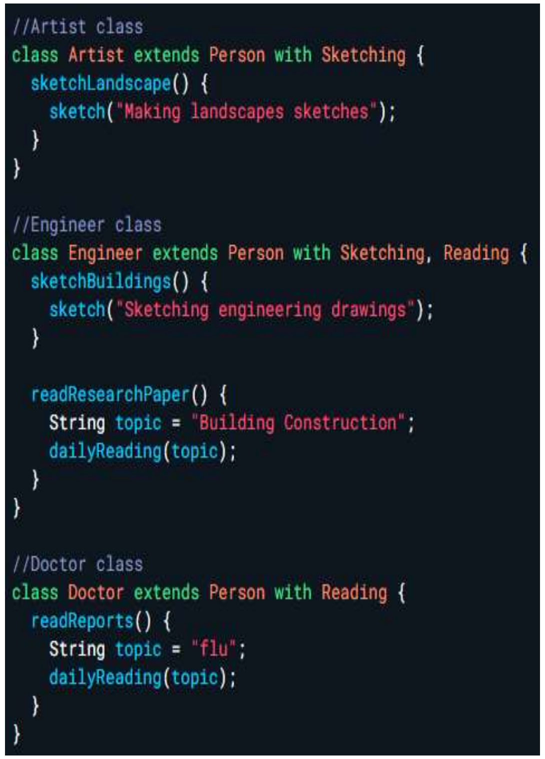
- **Important Points:**
  - Child classes inherit all properties and methods except constructors of the parent class.
  - Unlike Java, Dart also doesn't support multiple inheritance.

Mixins

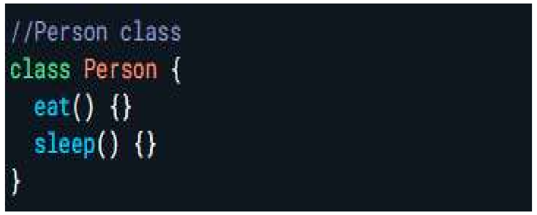
- Mixins keep Dart code reusable across separate classes. It's the most efficient way to reuse common code from multiple classes that share common behaviors.



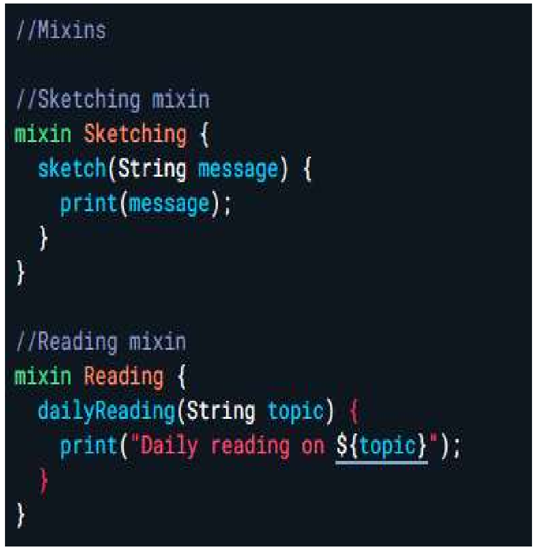
```
void main() {
  Artist artist = Artist();
  artist.sketchLandscape();

  Engineer engineer = Engineer();
  engineer.sketchBuildings();
  engineer.readResearchPaper();

  Doctor doctor = Doctor();
  doctor.readReports();
}
```



```
//Artist class
class Artist extends Person with Sketching {
  sketchLandscape() {
    sketch("Making landscapes sketches");
  }
}

//Engineer class
class Engineer extends Person with Sketching, Reading {
  sketchBuildings() {
    sketch("Sketching engineering drawings");
  }

  readResearchPaper() {
    String topic = "Building Construction";
    dailyReading(topic);
  }
}

//Doctor class
class Doctor extends Person with Reading {
  readReports() {
    String topic = "flu";
    dailyReading(topic);
  }
}
```



```
//Person class
class Person {
  eat() {}
  sleep() {}
}
```

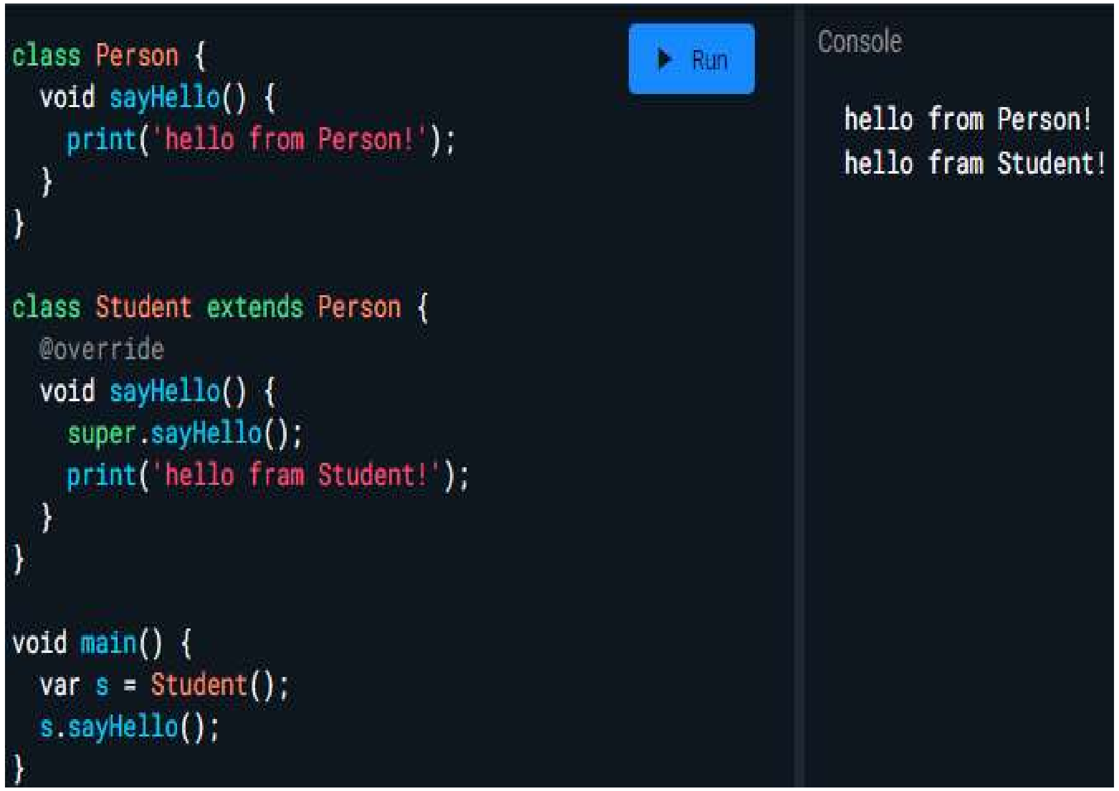


```
//Mixins

//Sketching mixin
mixin Sketching {
  sketch(String message) {
    print(message);
  }
}

//Reading mixin
mixin Reading {
  dailyReading(String topic) {
    print("Daily reading on ${topic}");
  }
}
```

Method Overriding in Dart

Method overriding occurs in dart when a child class tries to override the parent class's method. When a child class extends a parent class, it gets full access to the methods of the parent class and thus it overrides the methods of the parent class. It is achieved by re-defining the same method present in the parent class.

This method is helpful when you have to perform different functions for a different child class, so we can simply re-define the content by overriding it.



```dart
class Person {
  void sayHello() {
    print('hello from Person!');
  }
}

class Student extends Person {
  @override
  void sayHello() {
    super.sayHello();
    print('hello fram Student!');
  }
}

void main() {
  var s = Student();
  s.sayHello();
}
```

```
hello from Person!
hello fram Student!
```

Method Overriding in Dart

- When a *Child class* redefines the method of its *Parent class* then it is called Method Overriding.

- Overriding is done so that a *child class* can give its own implementation to the method which is already provided by the *parent class*.

- In this case, the method in the parent class is called the ***overridden method*** and the method in the child class is called ***overriding method***.

- The ***number*** and ***data type*** of function's argument ***must match*** while overriding the method.

- The main advantage of method overriding is that the individual class can give its own specific implementation to the inherited method **without even modifying the parent class code**.
- So now you don't need to create the same method again and again in different classes. If you want to put the different implementation in a different class with the same method you can do it by method overriding.