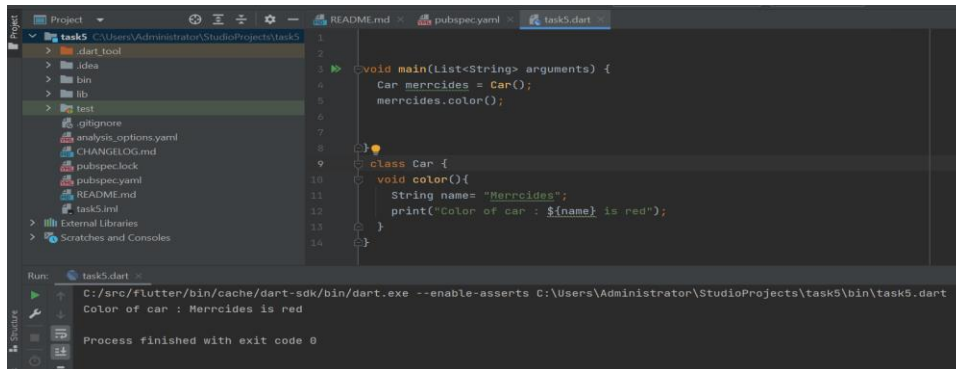


Abstract class :

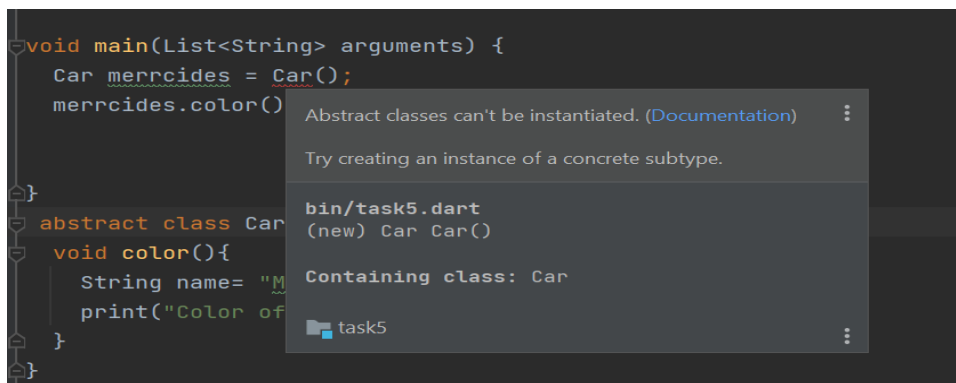
If we have Class Car, we can make instance of it as example :



```
1 void main(List<String> arguments) {
2   Car merrcides = Car();
3   merrcides.color();
4 }
5
6 class Car {
7   void color(){
8     String name= "Merrcides";
9     print("Color of car : ${name} is red");
10  }
11 }
```

Run: task5.dart
C:/src/flutter/bin/cache/dart-sdk/bin/dart.exe --enable-asserts C:\Users\Administrator\StudioProjects\task5\bin\task5.dart
Color of car : Merrcides is red
Process finished with exit code 0

But,if we change this class to abstract class, we show compile-time error as shown :



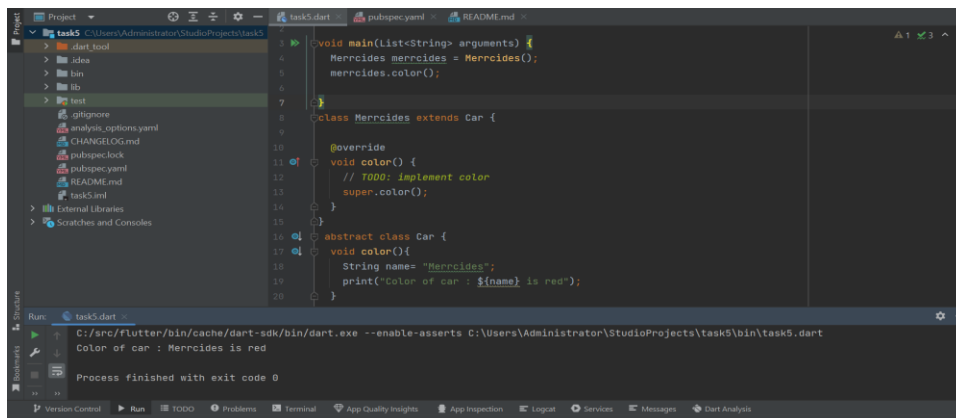
```
1 void main(List<String> arguments) {
2   Car merrcides = Car();
3   merrcides.color();
4 }
5
6 abstract class Car {
7   void color(){
8     String name= "Merrcides";
9     print("Color of car : ${name} is red");
10  }
11 }
```

Abstract classes can't be instantiated. (Documentation)
Try creating an instance of a concrete subtype.

bin/task5.dart
(new) Car Car()
Containing class: Car
task5

That mean we can't make instance of this class directly ,you should make child of it then can take instance of it.

Let's see :



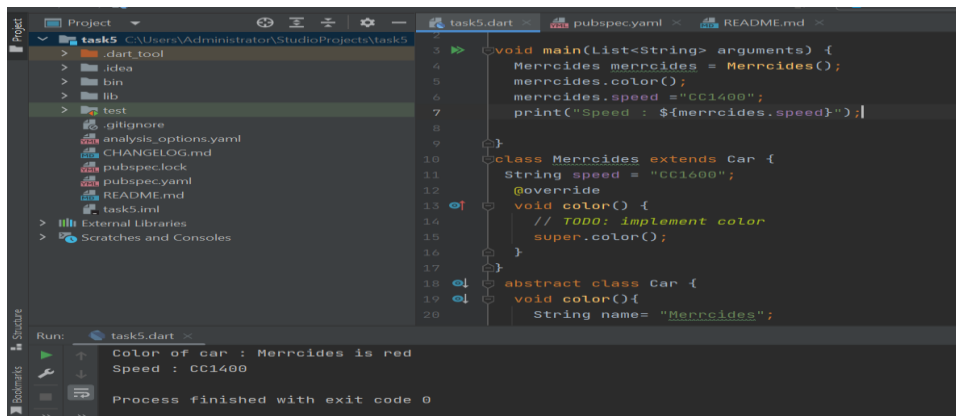
```
1 void main(List<String> arguments) {
2   Merrcides merrcides = Merrcides();
3   merrcides.color();
4 }
5
6 class Merrcides extends Car {
7   @override
8   void color() {
9     // TODO: implement color
10    super.color();
11  }
12 }
13
14 abstract class Car {
15   void color(){
16     String name= "Merrcides";
17     print("Color of car : ${name} is red");
18   }
19 }
```

Run: task5.dart
C:/src/flutter/bin/cache/dart-sdk/bin/dart.exe --enable-asserts C:\Users\Administrator\StudioProjects\task5\bin\task5.dart
Color of car : Merrcides is red
Process finished with exit code 0

- Definition: An abstract class is a template for subclasses that cannot be instantiated directly. It defines a common structure and ensures that subclasses implement specific methods.

Static:

From last example of class Merrcides ,we have variable called speed



```
void main(List<String> arguments) {
  Merrcides merrcides = Merrcides();
  merrcides.color();
  merrcides.speed = "CC1400";
  print("Speed : ${merrcides.speed}");
}

class Merrcides extends Car {
  String speed = "CC1600";
  @override
  void color() {
    // TODO: implement color
    super.color();
  }
}

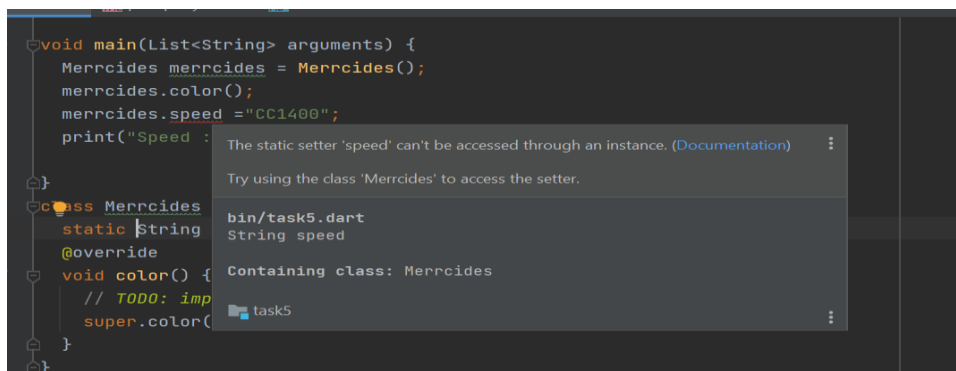
abstract class Car {
  void color();
  String name = "Merrcides";
}
```

Run: task5.dart

Color of car : Merrcides is red
Speed : CC1400

Process finished with exit code 0

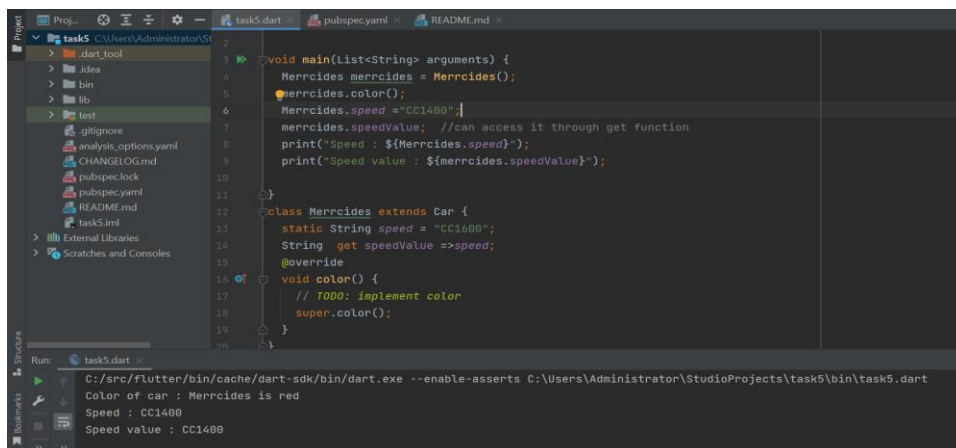
But if we change the speed into static variable ,what we show



```
void main(List<String> arguments) {
  Merrcides merrcides = Merrcides();
  merrcides.color();
  merrcides.speed = "CC1400";
  print("Speed : ");
}

class Merrcides {
  static String speed
  @override
  void color() {
    // TODO: imp
    super.color()
  }
}
```

We can't access it through instance you must access it through class or **get function**



```
void main(List<String> arguments) {
  Merrcides merrcides = Merrcides();
  merrcides.color();
  Merrcides.speed = "CC1400";
  merrcides.speedValue; //can access it through get function
  print("Speed : ${Merrcides.speed}");
  print("Speed value : ${merrcides.speedValue}");
}

class Merrcides extends Car {
  static String speed = "CC1600";
  String get speedValue => speed;
  @override
  void color() {
    // TODO: implement color
    super.color();
  }
}
```

Run: task5.dart

C:/src/Flutter/bin/cache/dart-sdk/bin/dart.exe --enable-asserts C:/Users/Administrator/StudioProjects/task5/bin/task5.dart

Color of car : Merrcides is red
Speed : CC1400
Speed value : CC1400

- Definition: The static keyword declares members that belong to the class itself, rather than individual instances. They are shared across all objects of the class.

Encapsulation

In Dart, **Encapsulation** means **hiding data** within a library, preventing it from outside factors. It helps you control your program and prevent it from becoming too complicated.

What Is Library In Dart?

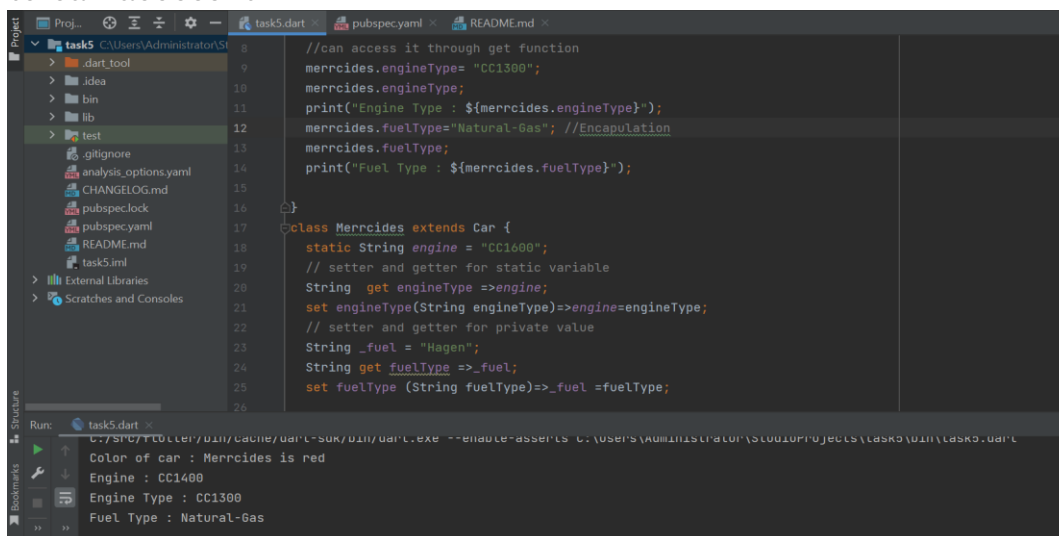
By default, every **.dart** file is a library. A library is a collection of functions and classes. A library can be imported into another library using the **import** keyword.

How To Achieve Encapsulation In Dart?

Encapsulation can be achieved by:

- Declaring the class properties as **private** by using **underscore(_)**.
- Providing public **getter** and **setter** methods to access and update the value of private property.

As shown before ,we used it speed we{ changename only to engine variable) to can access it

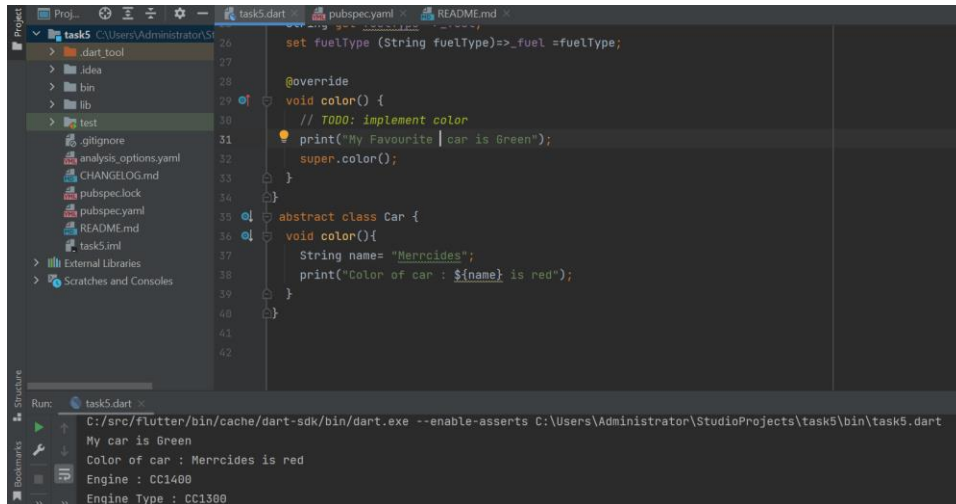


```
Project: Proj... task5.dart pubspec.yaml README.md
task5.dart
//can access it through get function
merrcides.engineType= "CC1300";
merrcides.engineType;
print("Engine Type : ${merrcides.engineType}");
merrcides.fuelType="Natural-Gas"; //Encapsulation
merrcides.fuelType;
print("Fuel Type : ${merrcides.fuelType}");
}
class Merrcides extends Car {
  static String engine = "CC1600";
  // setter and getter for static variable
  String get engineType =>engine;
  set engineType(String engineType)=>engine=engineType;
  // setter and getter for private value
  String _fuel = "Hagen";
  String get fuelType =>_fuel;
  set fuelType (String fuelType)=>_fuel =fuelType;
}
Run: task5.dart
C:/S/C/TC0Ller/U/11/cacine/Uar/L/SUK/U/11/Uar/L/EXE --enable-asserts C:/Users/Administrator/Desktop/Projects/task5/U/11/task5.Uar/L
Color of car : Merrcides is red
Engine : CC1400
Engine Type : CC1300
Fuel Type : Natural-Gas
```

- Definition: Encapsulation protects and controls access to an object's internal state by bundling data and methods within a single unit.

Polymorphism

Poly means **many** and morph means **forms**. Polymorphism is the ability of an object to take on many forms. As humans, we have the ability to take on many forms. We can be a student, a teacher, a parent, a friend, and so on. Similarly, in object-oriented programming, polymorphism is the ability of an object to take on many forms.



```
task5.dart
26 set fuelType (String fuelType)=>_fuel =fuelType;
27
28
29
30
31 // TODO: implement color
32 print("My Favourite | car is Green");
33 super.color();
34
35
36 abstract class Car {
37   void color(){
38     String name= "Mercedes";
39     print("Color of car : ${name} is red");
40   }
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Run: task5.dart

```
C:/src/flutter/bin/cache/dart-sdk/bin/dart.exe --enable-asserts C:\Users\Administrator\StudioProjects\task5\bin\task5.dart
My car is Green
Color of car : Mercedes is red
Engine : CC1400
Engine Type : CC1300
```

- Definition: Polymorphism allows objects of different classes to respond to the same method call in different ways.
- Types:
 - Runtime polymorphism (method overriding)

Advantage Of Polymorphism In Dart

- Subclasses can override the behavior of the parent class.
- It allows us to write code that is more flexible and reusable.