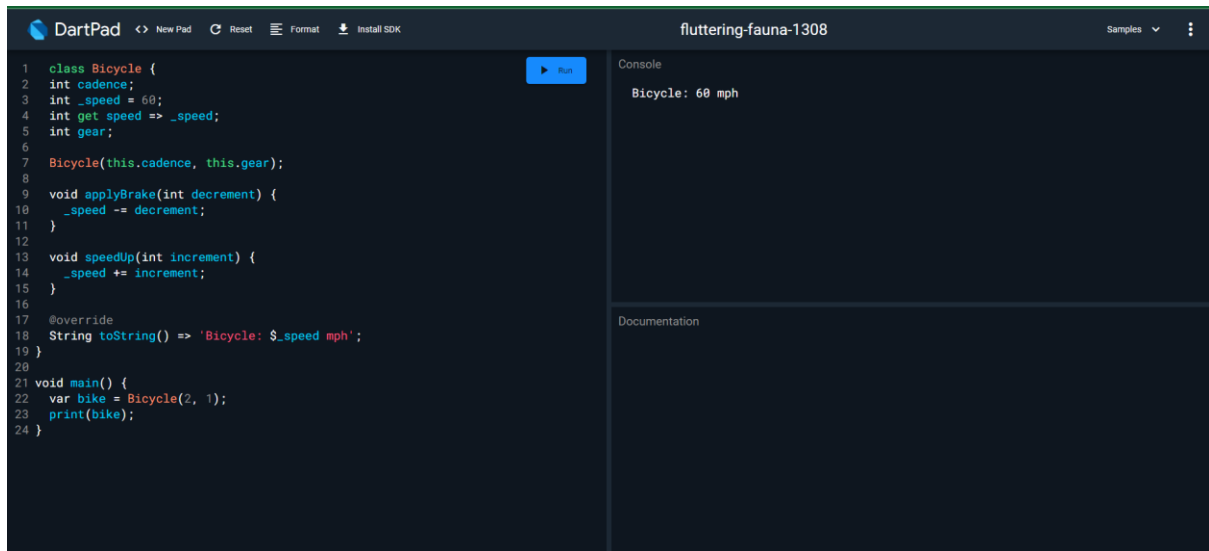


NAMA : RIJAL AHMAD JANANI

NPM : 065119062

1. MEMBUAT CLASS DART SEDERHANA



The screenshot shows the DartPad interface with a file named 'fluttering-fauna-1308'. The code defines a `Bicycle` class with attributes `cadence`, `speed`, and `gear`. It includes methods `applyBrake` and `speedUp`. The `toString` method is overridden to return a string like 'Bicycle: 60 mph'. In the `main` function, a `Bicycle` object is created with `cadence: 2` and `gear: 1`, and its string representation is printed. The console output shows 'Bicycle: 60 mph'.

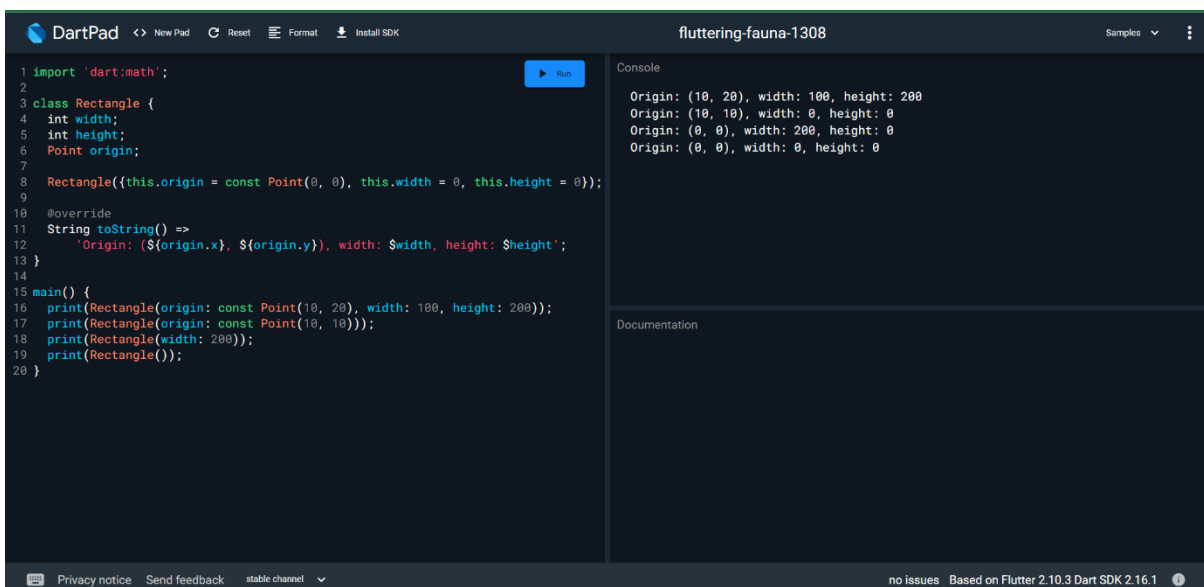
```
1 class Bicycle {
2   int cadence;
3   int _speed = 60;
4   int get speed => _speed;
5   int gear;
6
7   Bicycle(this.cadence, this.gear);
8
9   void applyBrake(int decrement) {
10    _speed -= decrement;
11  }
12
13  void speedUp(int increment) {
14    _speed += increment;
15  }
16
17  @override
18  String toString() => 'Bicycle: $_speed mph';
19 }
20
21 void main() {
22   var bike = Bicycle(2, 1);
23   print(bike);
24 }
```

Console

Bicycle: 60 mph

Documentation

2. PARAMETER OPERASIONAL



The screenshot shows the DartPad interface with a file named 'fluttering-fauna-1308'. The code defines a `Rectangle` class with positional parameters `width` and `height`, and a named parameter `origin` of type `Point`. The `toString` method is overridden to return a string like 'Origin: (10, 20), width: 100, height: 200'. In the `main` function, four `Rectangle` objects are created and printed: one with all parameters, one with `origin` and `height`, one with `origin` and `width`, and one with only `width` and `height`. The console output shows the string representation for each object.

```
1 import 'dart:math';
2
3 class Rectangle {
4   int width;
5   int height;
6   Point origin;
7
8   Rectangle({this.origin = const Point(0, 0), this.width = 0, this.height = 0});
9
10  @override
11  String toString() =>
12    'Origin: (${origin.x}, ${origin.y}), width: $width, height: $height';
13 }
14
15 main() {
16   print(Rectangle(origin: const Point(10, 20), width: 100, height: 200));
17   print(Rectangle(origin: const Point(10, 10)));
18   print(Rectangle(width: 200));
19   print(Rectangle());
20 }
```

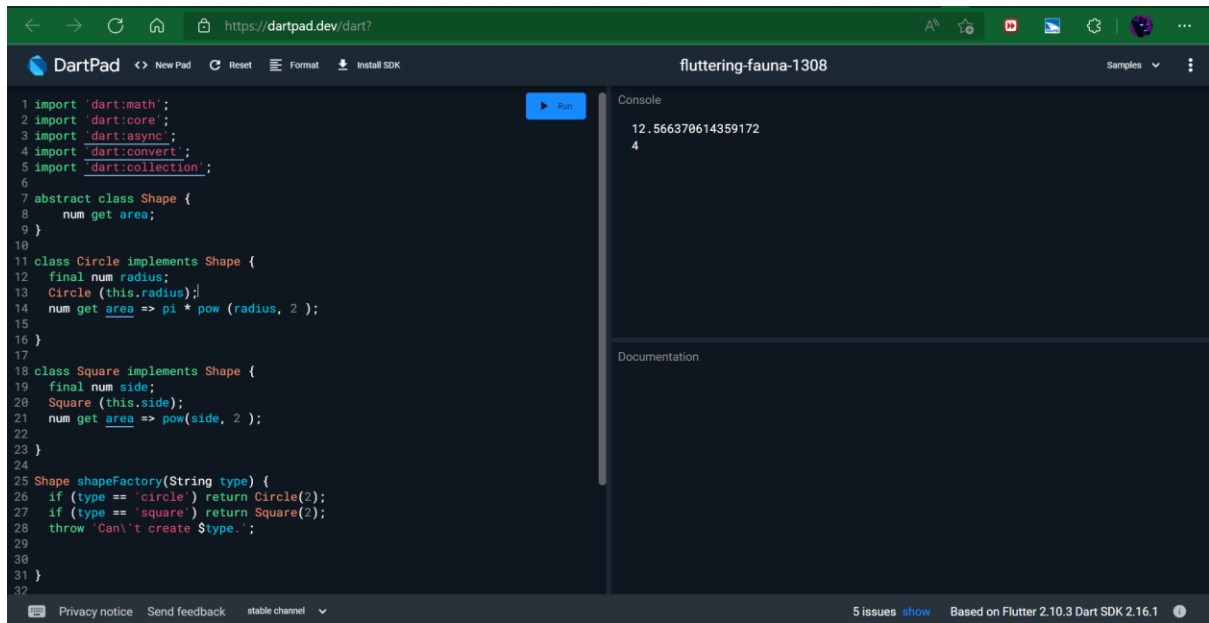
Console

Origin: (10, 20), width: 100, height: 200
Origin: (10, 10), width: 0, height: 0
Origin: (0, 0), width: 200, height: 0
Origin: (0, 0), width: 0, height: 0

Documentation

Privacy notice Send feedback stable channel no issues Based on Flutter 2.10.3 Dart SDK 2.16.1

3. MEMBUAT PABRIK



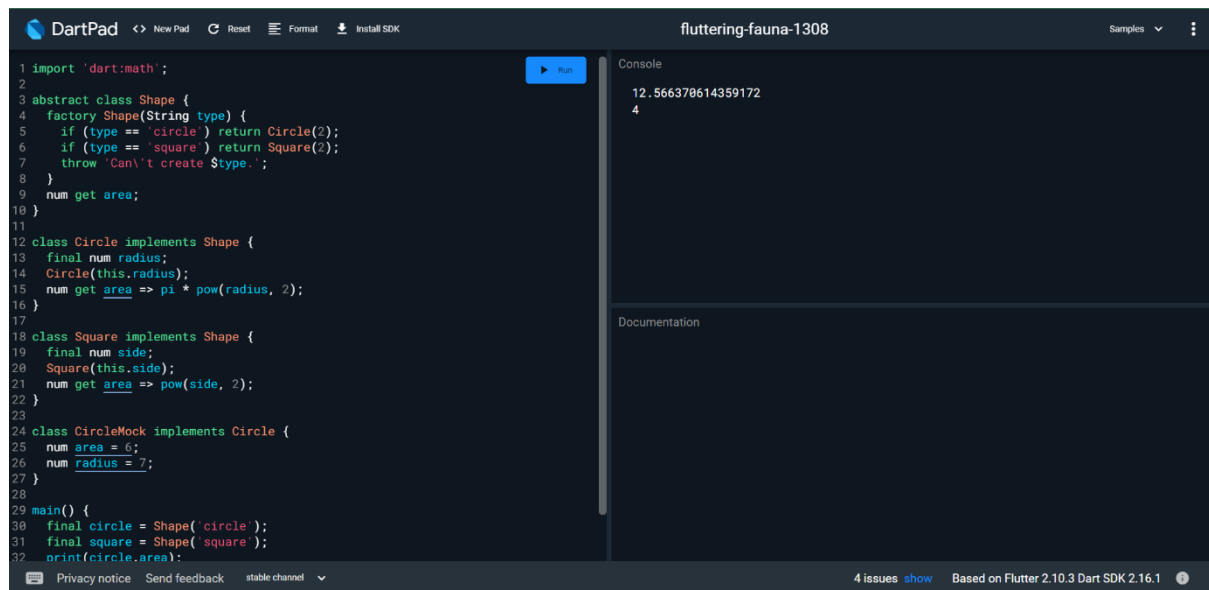
The screenshot shows the DartPad interface with a Dart code snippet implementing a factory method for a Shape class. The code defines an abstract Shape class with an area property, and two concrete classes, Circle and Square, that implement the Shape interface. A ShapeFactory class is also defined, which uses the factory method to create Shape objects based on a string input. The console output shows the result of running the code, which is a double value representing the area of a circle with radius 2.

```
1 import 'dart:math';
2 import 'dart:core';
3 import 'dart:async';
4 import 'dart:convert';
5 import 'dart:collection';
6
7 abstract class Shape {
8   num get area;
9 }
10
11 class Circle implements Shape {
12   final num radius;
13   Circle(this.radius);
14   num get area => pi * pow(radius, 2);
15 }
16
17 class Square implements Shape {
18   final num side;
19   Square(this.side);
20   num get area => pow(side, 2);
21 }
22
23
24 Shape shapeFactory(String type) {
25   if (type == 'circle') return Circle(2);
26   if (type == 'square') return Square(2);
27   throw 'Can\'t create $type.';
28 }
29
30
31 }
```

Console output:

```
12.566370614359172
4
```

4. MENETAPKAN ANTAR MUKA



The screenshot shows the DartPad interface with a Dart code snippet implementing a factory method for a Shape class. The code defines an abstract Shape class with an area property, and two concrete classes, Circle and Square, that implement the Shape interface. A ShapeFactory class is also defined, which uses the factory method to create Shape objects based on a string input. The console output shows the result of running the code, which is a double value representing the area of a circle with radius 2.

```
1 import 'dart:math';
2
3 abstract class Shape {
4   factory Shape(String type) {
5     if (type == 'circle') return Circle(2);
6     if (type == 'square') return Square(2);
7     throw 'Can\'t create $type.';
8   }
9   num get area;
10 }
11
12 class Circle implements Shape {
13   final num radius;
14   Circle(this.radius);
15   num get area => pi * pow(radius, 2);
16 }
17
18 class Square implements Shape {
19   final num side;
20   Square(this.side);
21   num get area => pow(side, 2);
22 }
23
24 class CircleMock implements Circle {
25   num area = 6;
26   num radius = 7;
27 }
28
29 main() {
30   final circle = Shape('circle');
31   final square = Shape('square');
32   print(circle.area);
33 }
```

Console output:

```
12.566370614359172
4
```

5. PEMEGRORAMAN FUNGSIONAL

