Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 1

по курсу «Разработка мобильных приложений»

«Графический пользовательский интерфейс в Dart»

Студент группы ИУ9-71Б Яровикова А. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель

Научиться создавать приложения с графическим пользовательским интерфейсом с использованием фреймворка Flutter на языке программирования Dart

2 Задание

В течение лабораторной работы нужно разработать программу, рисующую на экране мобильного устройства ромб, диагонали которого имеют величины a и b и рисуются по желанию пользователя выбранным пользователем цветом. Программа должна иметь графический пользовательский интерфейс, через который пользователь может задавать параметры изображения. Изображение должно перерисовываться автоматически при изменении любого параметра. Значения параметров представляют собой неотрицательные целые числа. Когда в описании изображения говорится о выборе цвета, подразумевается выбор из нескольких предопределённых альтернатив (например, красный, зелёный или синий).

3 Реализация

Исходный код представлен в листинге 1.

Github Gist

DartPad

Листинг 1: main.dart

```
1  import 'package:flutter/material.dart';
2  import 'dart:ui';
3
4  void main() {
5    runApp(MyApp());
6  }
7
8  class MyApp extends StatelessWidget {
9    @override
10   Widget build(BuildContext context) {
11     return MaterialApp(
12       debugShowCheckedModeBanner: false,
13       home: Scaffold(
14         appBar: AppBar(
```

```
15            title: Text('lab1 - Flutter Custom Drawer'),
16          ),
17          body: Center(
18            child: DiamondPainter(),
19          ),
20        ),
21      );
22    }
23  }
24
25  class DiamondPainter extends StatefulWidget {
26    @override
27    _DiamondPainterState createState() => _DiamondPainterState();
28  }
29
30  class _DiamondPainterState extends State<DiamondPainter> {
31    // TextEditingController _diagonalAController = TextEditingController
        ();
32    // TextEditingController _diagonalBController = TextEditingController
        ();
33    var _diagA = 100.0;
34    var _diagB = 100.0;
35    Color _selectedColor = Colors.black;
36
37    @override
38    Widget build(BuildContext context) {
39      return Column(
40        mainAxisAlignment: MainAxisAlignment.center,
41        children: <Widget>[
42          Text('Diagonal A'),
43          Slider(
44            value: _diagA,
45            min: 100.0,
46            max: 700.0,
47            label: _diagA.toInt().toString(),
48            divisions: 10,
49            onChanged: (value) {
50              setState(() {
51                _diagA = value;
52              });
53            },
54          ),
55          SizedBox(height:10),
56          Text('Diagonal B'),
57          Slider(
58            value: _diagB,
```

```
59            min: 100.0,
60            max: 700.0,
61            label: _diagB.toInt().toString(),
62            divisions: 10,
63            onChanged: (value) {
64              setState(() {
65                _diagB = value;
66              });
67            },
68          ),
69          SizedBox(height: 20),
70
71          Row(
72            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
73            children: [
74              ElevatedButton(
75                  style: ElevatedButton.styleFrom(backgroundColor: Colors.
    red,),
76                  onPressed: () {
77                    setState(() {
78                      _selectedColor = Colors.red;
79                    });
80                  },
81                child: const Text("red"),
82              ),
83              ElevatedButton(
84                style: ElevatedButton.styleFrom(backgroundColor: Colors.
    blue,),
85                onPressed: () {
86                  setState(() {
87                    _selectedColor = Colors.blue;
88                  });
89                },
90                child: const Text("blue"),
91              ),
92              ElevatedButton(
93                style: ElevatedButton.styleFrom(backgroundColor: Colors.
    lightGreenAccent,),
94                onPressed: () {
95                  setState(() {
96                    _selectedColor = Colors.lightGreenAccent;
97                  });
98                },
99                child: const Text("green"),
100             ),
101             ElevatedButton(
```

```
102                 style: ElevatedButton.styleFrom(backgroundColor: Colors.
    yellow,),
103                 onPressed: () {
104                   setState(() {
105                     _selectedColor = Colors.yellow;
106                   });
107                 },
108                 child: const Text("yellow"),
109               ),
110               ElevatedButton(
111                 style: ElevatedButton.styleFrom(backgroundColor: Colors.
    pink.shade200,),
112                 onPressed: () {
113                   setState(() {
114                     _selectedColor = Colors.pink.shade200;
115                   });
116                 },
117                 child: const Text("pink"),
118               ),
119             ],
120           ),
121
122
123           SizedBox(height: 20),
124           CustomPaint(
125             size: Size(200, 200),
126             painter: DiamondCustomPainter(
127               _diagA,
128               _diagB,
129               _selectedColor,
130             ),
131           ),
132         ],
133       );
134   }
135
136   void _updateColor(Color color) {
137     setState(() {
138       _selectedColor = color;
139     });
140   }
141 }
142
143 class DiamondCustomPainter extends CustomPainter {
144   final double diagonalA;
145   final double diagonalB;
```

```
146   final Color color;
147
148   DiamondCustomPainter(this.diagonalA, this.diagonalB, this.color);
149
150   @override
151   void paint(Canvas canvas, Size size) {
152     Paint paint = Paint()
153       ..color = color
154       ..strokeWidth = 3
155       ..style = PaintingStyle.stroke;
156
157     var path = Path();
158
159     Offset center = Offset(size.width / 2, size.height / 2);
160     // diag A left corner point
161     Offset start = Offset(center.dx - diagonalA / 2.0, center.dy);
162
163
164     path.moveTo(start.dx, start.dy);
165
166     // diag B right corner point
167     path.lineTo(center.dx, center.dy + diagonalB / 2.0);
168     //   diag A right  corner point
169     path.lineTo(center.dx + diagonalA / 2.0, center.dy);
170     //  diag B left  corner point
171     path.lineTo(center.dx, center.dy - diagonalB / 2.0);
172     // diag A left corner point
173     path.lineTo(start.dx, start.dy);
174
175     // diagonals
176     //  A
177     path.lineTo(center.dx + diagonalA / 2.0, center.dy);
178     //  B
179     path.lineTo(center.dx, center.dy - diagonalB / 2.0);
180     path.lineTo(center.dx, center.dy + diagonalB / 2.0);
181
182     path.close();
183     canvas.drawPath(path, paint);
184   }
185
186   @override
187   bool shouldRepaint(covariant CustomPainter oldDelegate) {
188     return false;
189   }
190 }
```

4 Результаты

Результат представлен на рисунках 1 - 2.

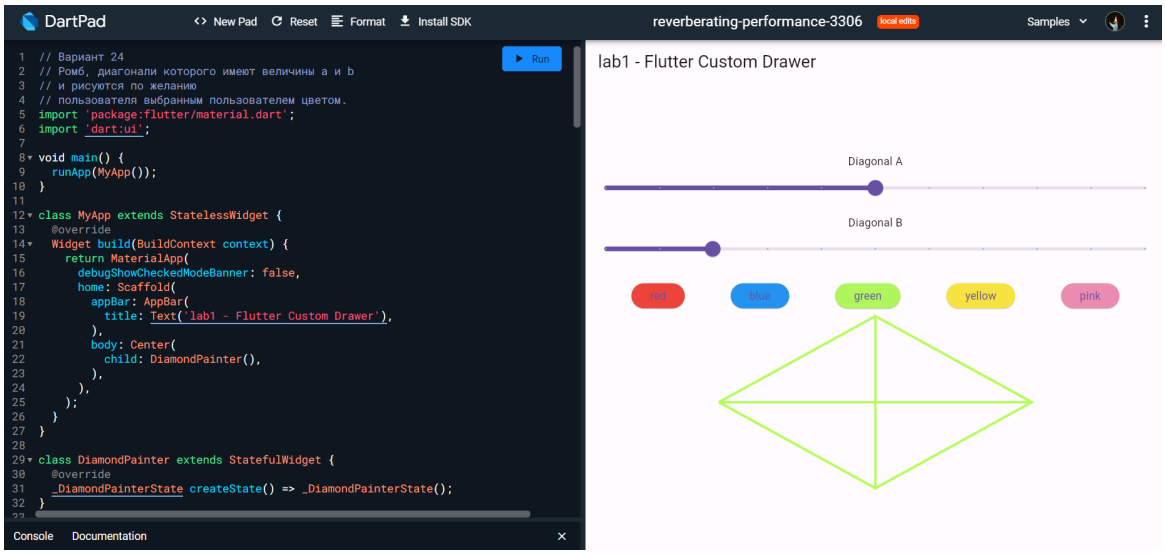

Рис. 1 — Заданные параметры: диагональ b больше диагонали a, цвет зеленый

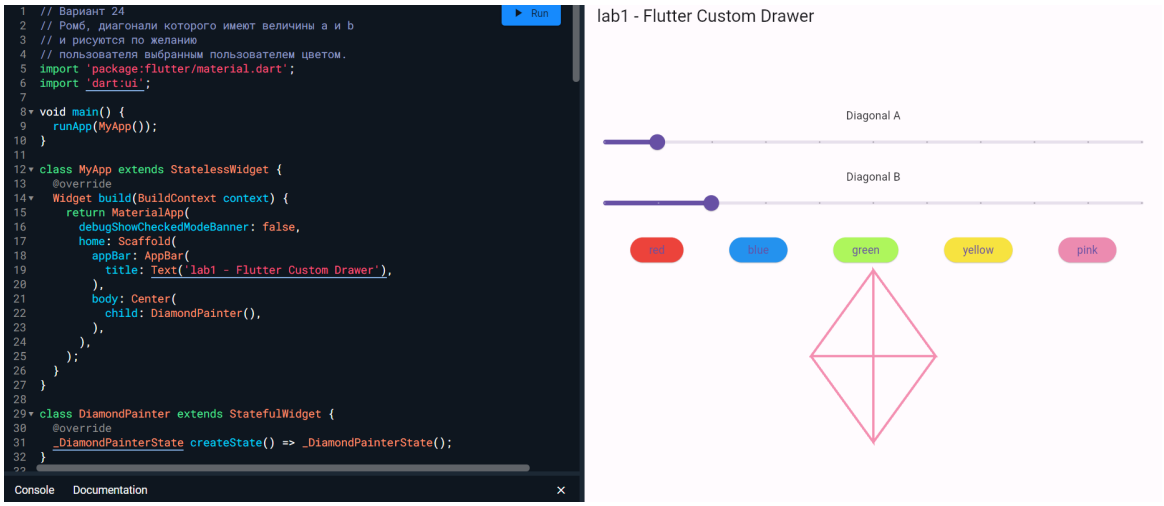

Рис. 2 — Заданные параметры: диагональ a больше диагонали b, цвет розовый

5 Выводы

В результате лабораторной работы было создано приложение с графическим пользовательским интерфейсом с использованием фреймворка Flutter на языке программирования Dart. Приложение позволяет изменять параметры рисунка, получая различные изображения.