Lab Report

Report No : 01

Report Title : Introduction to Dart Programming

Course Title: Mobile Application Design Lab

Course Code: CSE414

Submitted to,

Name: **Sadaf M. Anis**

Lecturer

Department of CSE

Daffodil International University

Submitted by,

Name: **Redwan Rahman**

ID: 0242220005101127

Section: 63_E1

Department of CSE

Daffodil International University

Date of Submission: 01/02/2026

Introduction:

This lab report documents the practical exercises performed during the second laboratory session of the Mobile Application Design Lab (CSE414). The tasks focused on fundamental Dart programming concepts including function definitions, data structures (Maps), control flow with exception handling, and basic object-oriented programming using classes. Each task was implemented and executed in the Dart environment to observe the expected outputs.

Objectives:

The objectives of this lab are:

- To understand the basic syntax and structure of the Dart programming language.
- To practice the uses of the Map data structure for storing key-value pairs.
- To implement fundamental functions for arithmetic operations.
- To utilize Collection types such as Maps and Lists.
- To apply Object-Oriented Programming (OOP) concepts by creating Classes and Methods.

Experimental Theory:

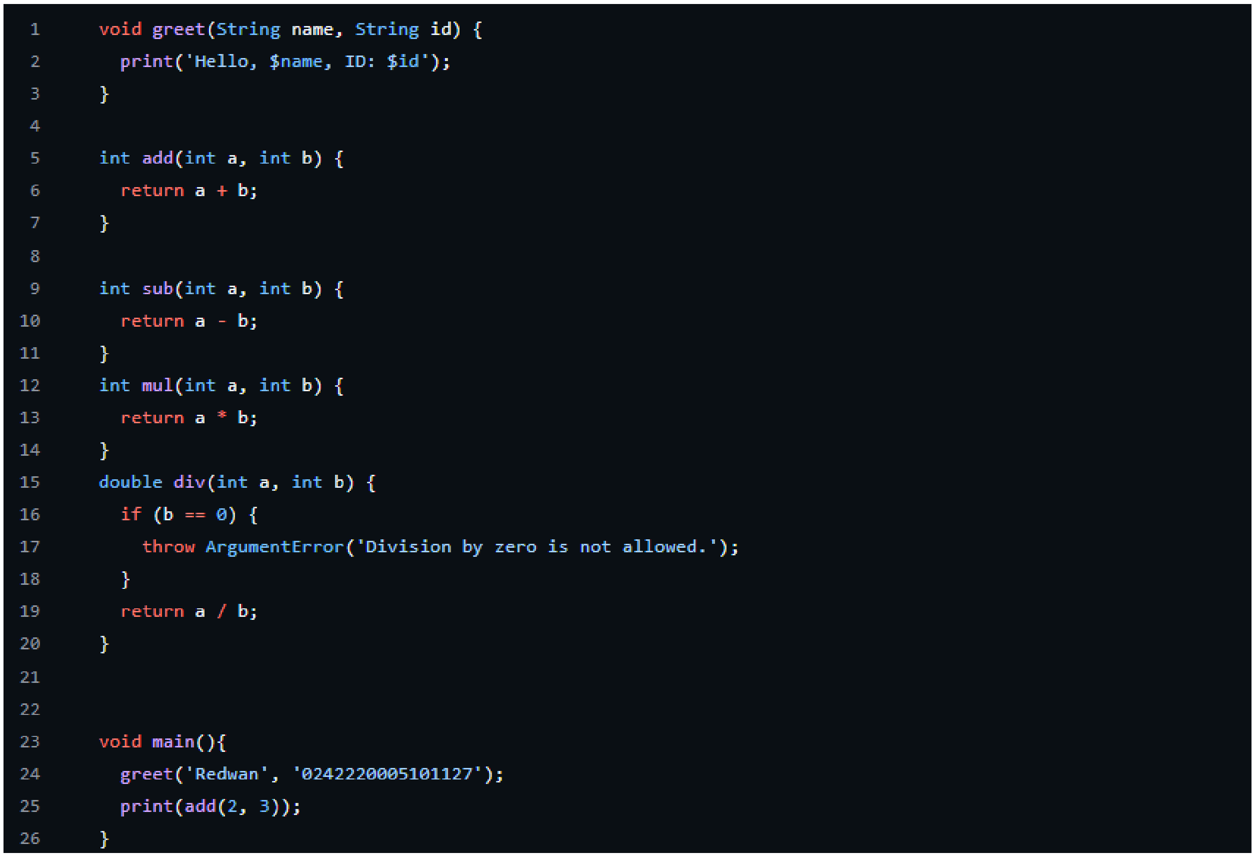
Dart is a client-optimized language for fast apps on any platform. In this lab, we explored:

- **Functions:** Reusable blocks of code (e.g., add, subtract).
- **Maps:** Key-value pair storage.
- **Lists:** Ordered groups of objects.
- **Classes:** Blueprints for creating objects, encapsulating data and methods.

Lab Tasks and Implementation:

Task 1: Basic Functions and Arithmetic

Problem: Create a program that defines functions for greeting a user and performing basic arithmetic operations (addition, subtraction, multiplication, division).



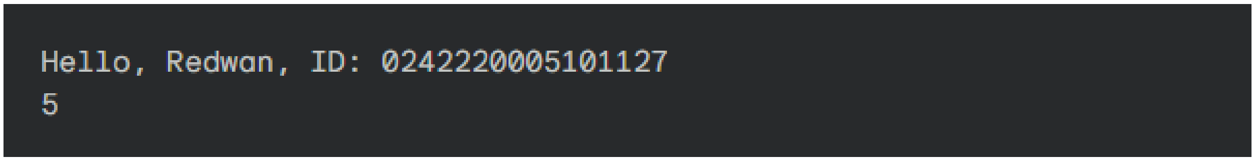
```
void greet(String name, String id) {
  print('Hello, $name, ID: $id');
}

int add(int a, int b) {
  return a + b;
}

int sub(int a, int b) {
  return a - b;
}
int mul(int a, int b) {
  return a * b;
}
double div(int a, int b) {
  if (b == 0) {
    throw ArgumentError('Division by zero is not allowed.');
  }
  return a / b;
}


void main(){
  greet('Redwan', '0242220005101127');
  print(add(2, 3));
}
```

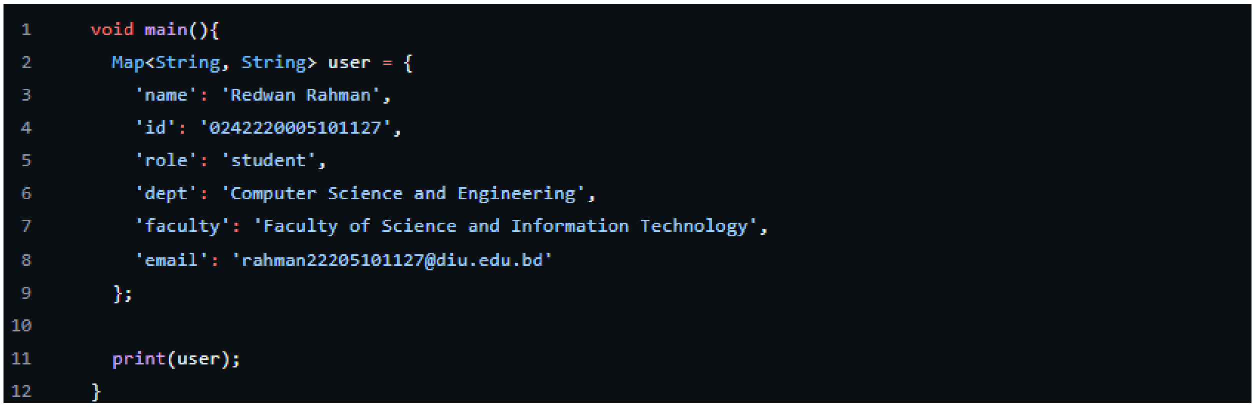
Output:



```
Hello, Redwan, ID: 0242220005101127
5
```

Task 2: Working with Maps

Problem: Implement a Dart Map to store and display student information, including name, ID, role, and contact details.



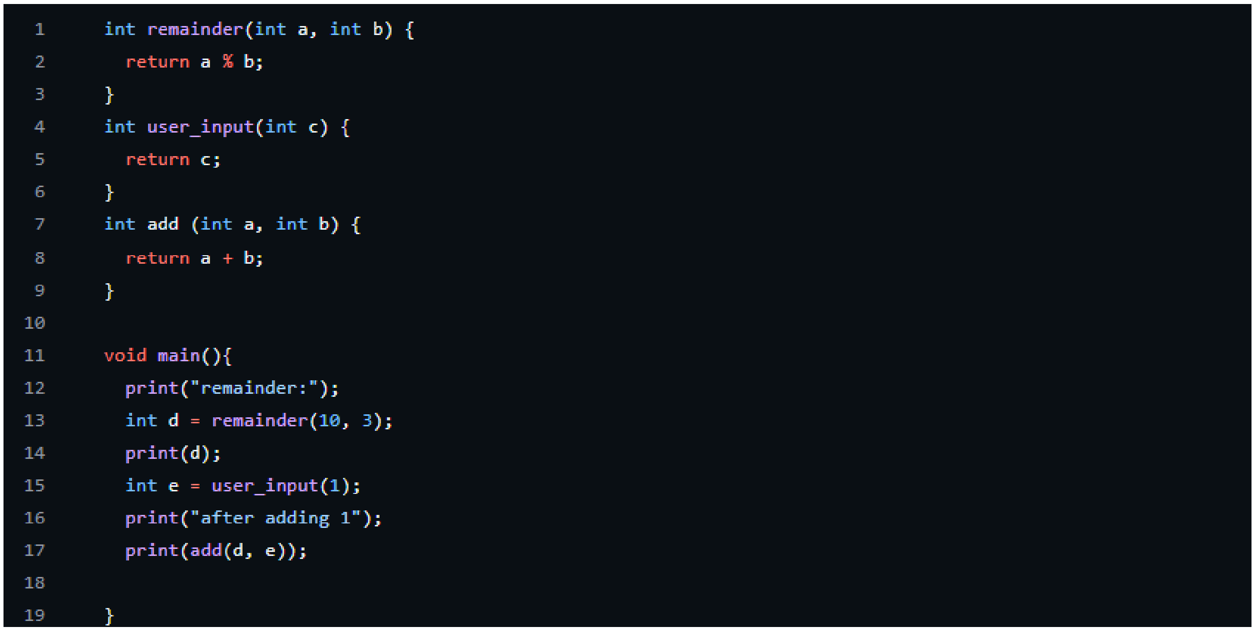
```
void main(){
  Map<String, String> user = {
    'name': 'Redwan Rahman',
    'id': '0242220005101127',
    'role': 'student',
    'dept': 'Computer Science and Engineering',
    'faculty': 'Faculty of Science and Information Technology',
    'email': 'rahman22205101127@diu.edu.bd'
  };

  print(user);
}
```

Output:
{name: Redwan Rahman, id: 0242220005101127, role: student, dept: Computer Science and Engineering, faculty: Faculty of Science and Information Technology, email: rahman22205101127@diu.edu.bd}

Task 3: Modulo Operator and Logic

Problem: Create functions to calculate the remainder of division and handle user input simulation.

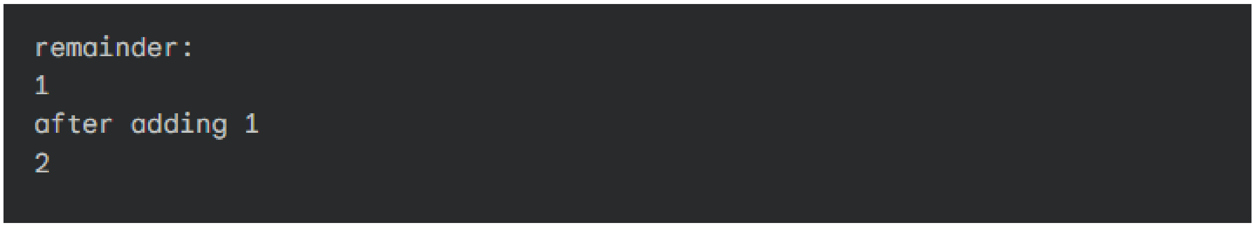


```
int remainder(int a, int b) {
  return a % b;
}
int user_input(int c) {
  return c;
}
int add (int a, int b) {
  return a + b;
}

void main(){
  print("remainder:");
  int d = remainder(10, 3);
  print(d);
  int e = user_input(1);
  print("after adding 1");
  print(add(d, e));

}
```

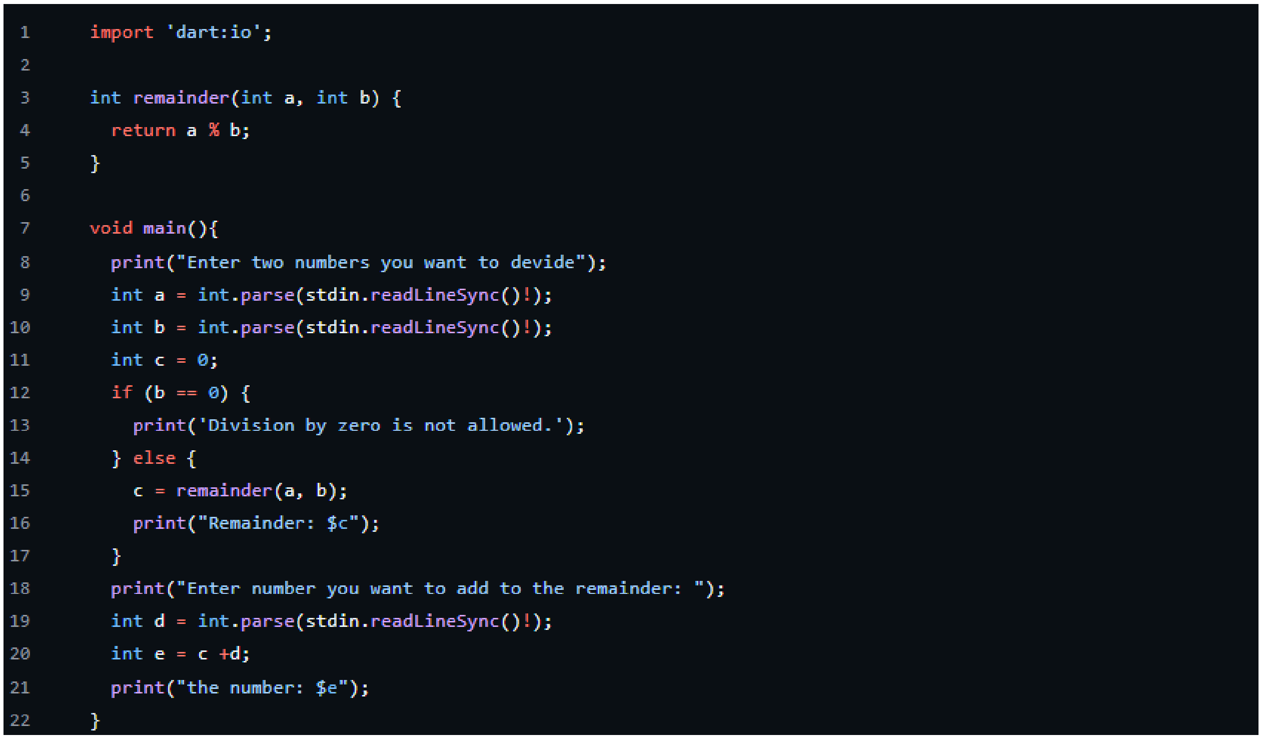
Output:



```
remainder:
1
after adding 1
2
```

Task 4: User Input and addition

Problem: Create a program that interacts with the user via the console to perform division and addition. The program should accept two numbers, calculate the remainder, and then accept a third number to add to that remainder.



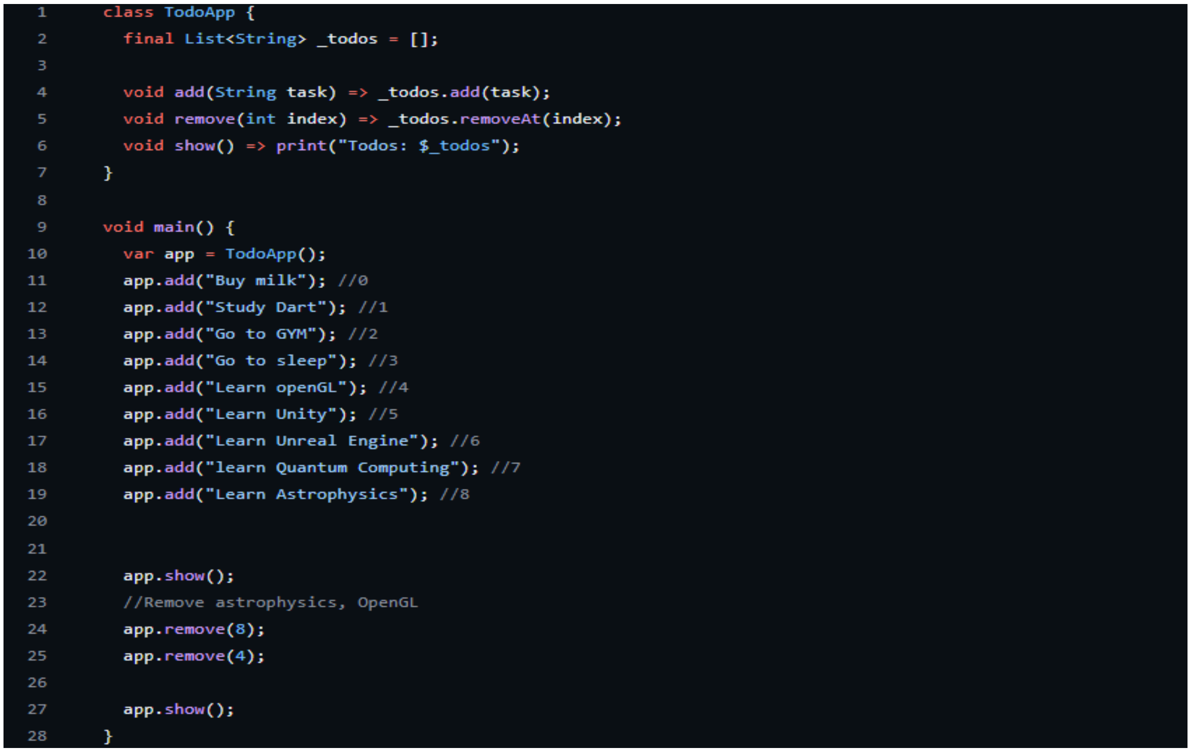
```
import 'dart:io';

int remainder(int a, int b) {
  return a % b;
}

void main(){
  print("Enter two numbers you want to devide");
  int a = int.parse(stdin.readLineSync()!);
  int b = int.parse(stdin.readLineSync()!);
  int c = 0;
  if (b == 0) {
    print('Division by zero is not allowed.');
  } else {
    c = remainder(a, b);
    print("Remainder: $c");
  }
  print("Enter number you want to add to the remainder: ");
  int d = int.parse(stdin.readLineSync()!);
  int e = c +d;
  print("the number: $e");
}
```

Task 5: To-Do App (Classes and Lists)

Problem: Implement a TodoApp class using a List to store tasks. The class should support adding tasks, removing specific tasks by index, and displaying the list.



```
class TodoApp {
  final List<String> _todos = [];

  void add(String task) => _todos.add(task);
  void remove(int index) => _todos.removeAt(index);
  void show() => print("Todos: $_todos");
}

void main() {
  var app = TodoApp();
  app.add("Buy milk"); //0
  app.add("Study Dart"); //1
  app.add("Go to GYM"); //2
  app.add("Go to sleep"); //3
  app.add("Learn openGL"); //4
  app.add("Learn Unity"); //5
  app.add("Learn Unreal Engine"); //6
  app.add("learn Quantum Computing"); //7
  app.add("Learn Astrophysics"); //8


  app.show();
  //Remove astrophysics, OpenGL
  app.remove(8);
  app.remove(4);

  app.show();
}
```

Output: Todos: [Buy milk, Study Dart, Go to GYM, Go to sleep, Learn openGL, Learn Unity, Learn Unreal Engine, learn Quantum Computing, Learn Astrophysics]
Todos: [Buy milk, Study Dart, Go to GYM, Go to sleep, Learn Unity, Learn Unreal Engine, learn Quantum Computing]

Discussion:

In this lab, we successfully set up the Dart environment and practiced writing basic Dart code. We learned how to define typed functions, handle string interpolation, and manage data using Maps. The implementation of the TodoApp class provided practical experience in manipulating Lists within an Object-Oriented structure. Additionally, **Task 4** introduced the dart:io library, allowing us to build interactive command-line applications that accept user input and process it dynamically. This comprehensive practice forms a solid foundation for mobile application development.

[Note: All codes are submitted to GitHub during lab sessions]

Conclusion:

The laboratory tasks were completed successfully and provided valuable hands-on experience with fundamental Dart programming concepts. By implementing arithmetic functions, data structures, object-oriented classes, and interactive console programs, we developed a strong foundation in writing structured and efficient Dart code.

This lab has prepared us for more advanced topics in mobile application development and increased our confidence in applying Dart concepts to real-world software design using Flutter and related technologies.