

UNIVERSITY OF RWANDA
COLLEGE OF SCIENCE AND TECHNOLOGY
GAKO CAMPUS
COMPUTER ENGINEERING
MOBILE APPLICATIONS
DATE ON 15th feb,2026

Group members :-Abdoul SIBO 223026681
-JUSTINE IRAFASHA 223026693
-FILS NIYINZIMA 223026876

DART LAB REPORT (lab 1)

Part 1: Functions

Q1: Write a Dart function named `welcomeMessage` that prints a welcome message for the school system.

Answer screenshot 1:



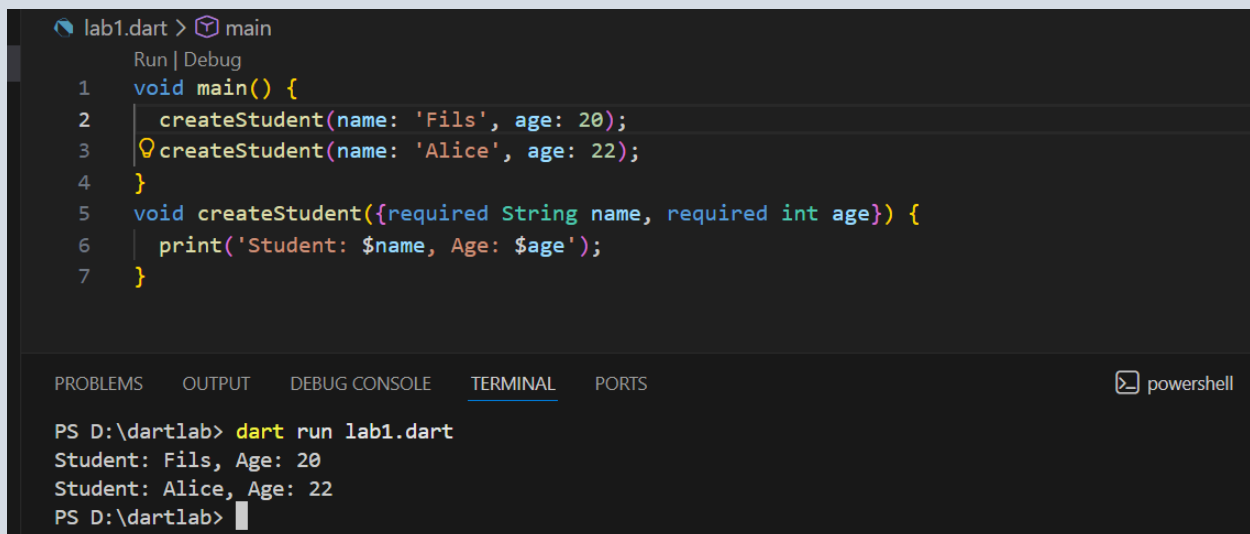
```
lab1.dart > welcomeMessage
1 void main() {
2   welcomeMessage();
3 }
4 void welcomeMessage() {
5   print('Welcome to the Mobo!');
6 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\dartlab> dart run lab1.dart
Welcome to the Mobo!
PS D:\dartlab>
```

Q2: Write a function named `createStudent` that uses named parameters (`name` and `age`) and prints the student details.

Answer screenshot 2:



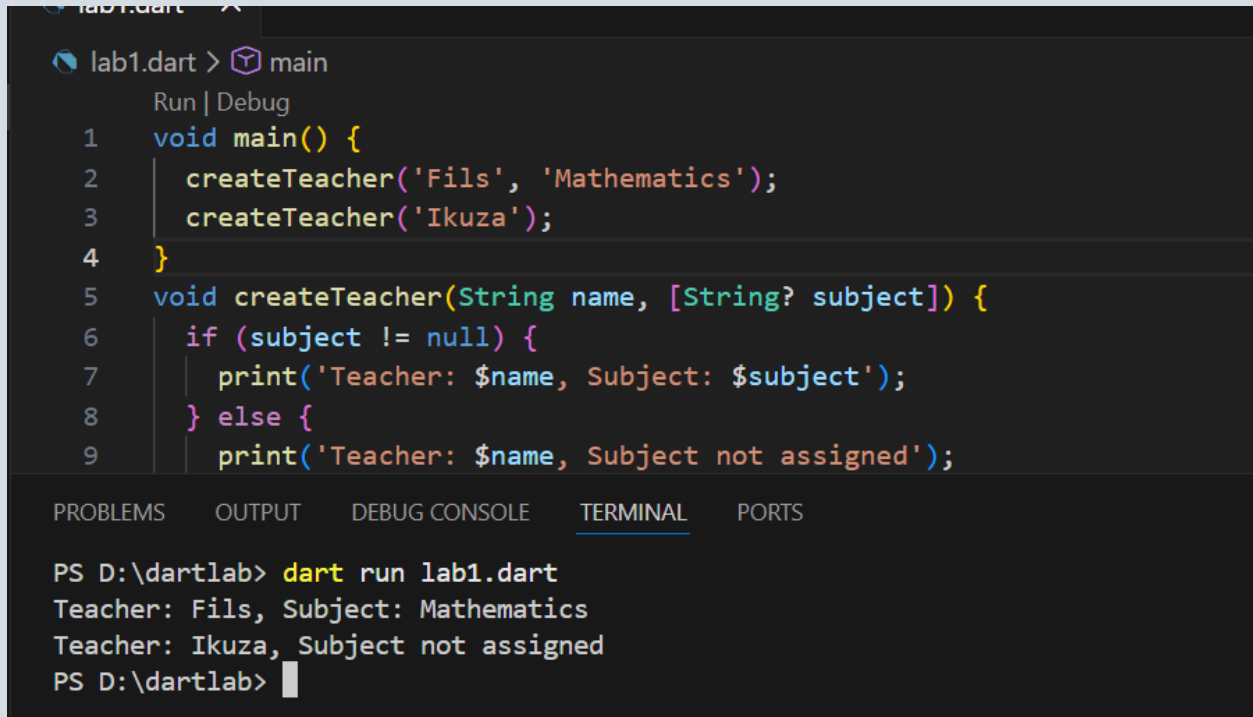
```
lab1.dart > main
Run | Debug
1 void main() {
2   createStudent(name: 'Fils', age: 20);
3   createStudent(name: 'Alice', age: 22);
4 }
5 void createStudent({required String name, required int age}) {
6   print('Student: $name, Age: $age');
7 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

PS D:\dartlab> dart run lab1.dart
Student: Fils, Age: 20
Student: Alice, Age: 22
PS D:\dartlab>
```

Q3: Write a function named `createTeacher` with one required parameter `name` and one optional parameter `subject`. If subject is not provided, print 'Subject not assigned'.

Answer screenshot 3:



The screenshot shows an IDE window with a file named 'lab1.dart'. The code defines a `main` function and a `createTeacher` function. The `main` function calls `createTeacher` twice: first with 'Fils' and 'Mathematics', and then with 'Ikuza' and no subject. The `createTeacher` function uses an `if` statement to check if the subject is not null. The output pane shows the results of these calls: 'Teacher: Fils, Subject: Mathematics' and 'Teacher: Ikuza, Subject not assigned'.

```
lab1.dart > main
Run | Debug
1 void main() {
2   createTeacher('Fils', 'Mathematics');
3   createTeacher('Ikuza');
4 }
5 void createTeacher(String name, [String? subject]) {
6   if (subject != null) {
7     print('Teacher: $name, Subject: $subject');
8   } else {
9     print('Teacher: $name, Subject not assigned');
10 }

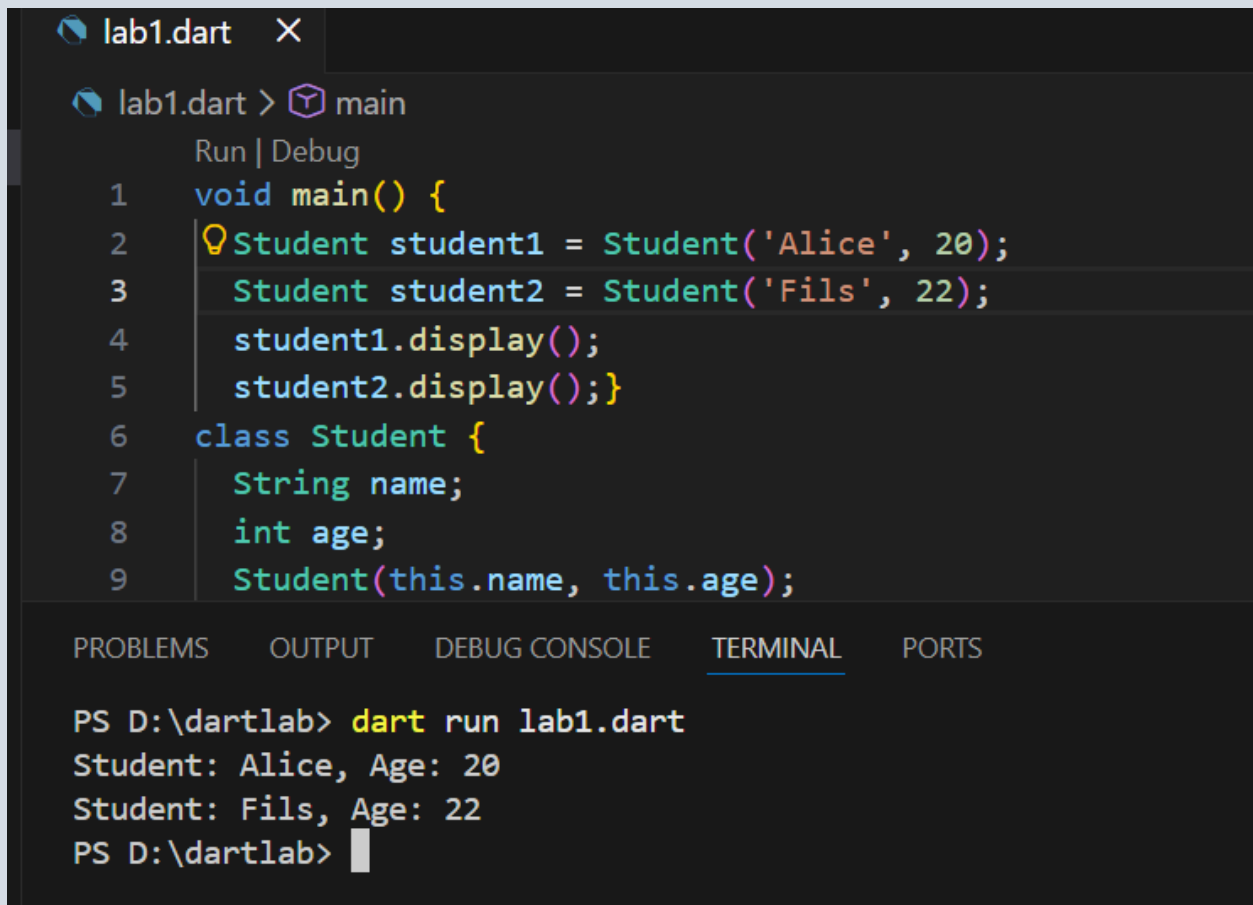
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\dartlab> dart run lab1.dart
Teacher: Fils, Subject: Mathematics
Teacher: Ikuza, Subject not assigned
PS D:\dartlab>
```

Part 2: Constructors and Classes

Q4: Create a class named `Student` with `name` and `age`, and a constructor to initialize these values. **Together with** Q5: Create an object of `Student` and print

the student's name and age [.Screenshot](#)



```
lab1.dart X
lab1.dart > main
Run | Debug
1 void main() {
2   Student student1 = Student('Alice', 20);
3   Student student2 = Student('Fils', 22);
4   student1.display();
5   student2.display();}
6 class Student {
7   String name;
8   int age;
9   Student(this.name, this.age);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

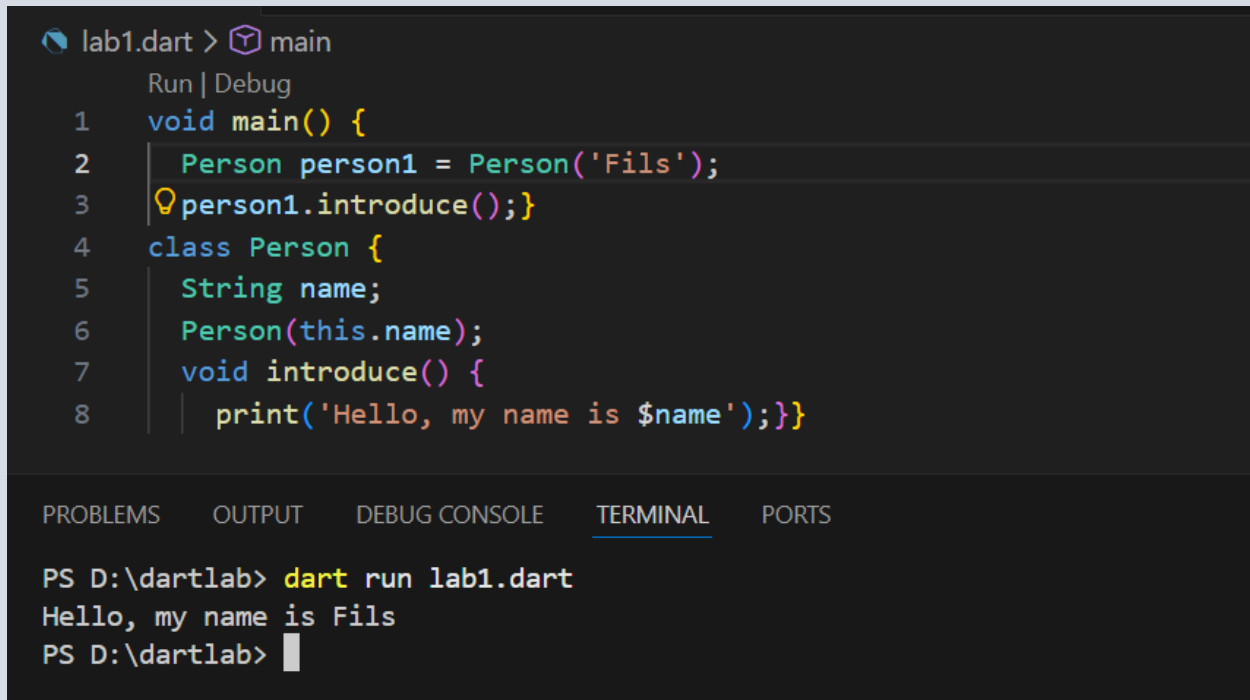
PS D:\dartlab> dart run lab1.dart
Student: Alice, Age: 20
Student: Fils, Age: 22
PS D:\dartlab> 
```

Part 3: Inheritance

Q6: Create a class `Person` with a variable `name` and a function `introduce()` that prints

the name.

screenshot for Answer 6:



The screenshot shows an IDE window with a Dart file named 'lab1.dart'. The code defines a 'Person' class with a 'name' property and an 'introduce()' method that prints 'Hello, my name is \$name'. In the 'main()' function, a 'Person' object named 'person1' is created with the name 'Fils', and its 'introduce()' method is called. Below the code editor, the 'TERMINAL' tab is active, showing the command 'dart run lab1.dart' being executed, which results in the output 'Hello, my name is Fils'.

```
lab1.dart > main
Run | Debug
1 void main() {
2   Person person1 = Person('Fils');
3   person1.introduce();
4 }
class Person {
5   String name;
6   Person(this.name);
7   void introduce() {
8     print('Hello, my name is $name');
9   }
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\dartlab> dart run lab1.dart
Hello, my name is Fils
PS D:\dartlab>
```

Q7: Make `Student` inherit from `Person` and call `introduce()` from a `Student` object.

Answer screenshot 7:

```
lab1.dart > Person > introduce
Run | Debug
1 void main() {
2   Student student1 = Student(' Fils', 20);
3   student1.introduce();
4 }
5 class Person {
6   String name;
7   Person(this.name);
8   void introduce() {
9     print('Hello, myn inherited name is $name');}}
10 class Student extends Person {
11   int age;
12   Student(String name, this.age) : super(name);
13   void display() {
14     print('Student: $name, Age: $age');}}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

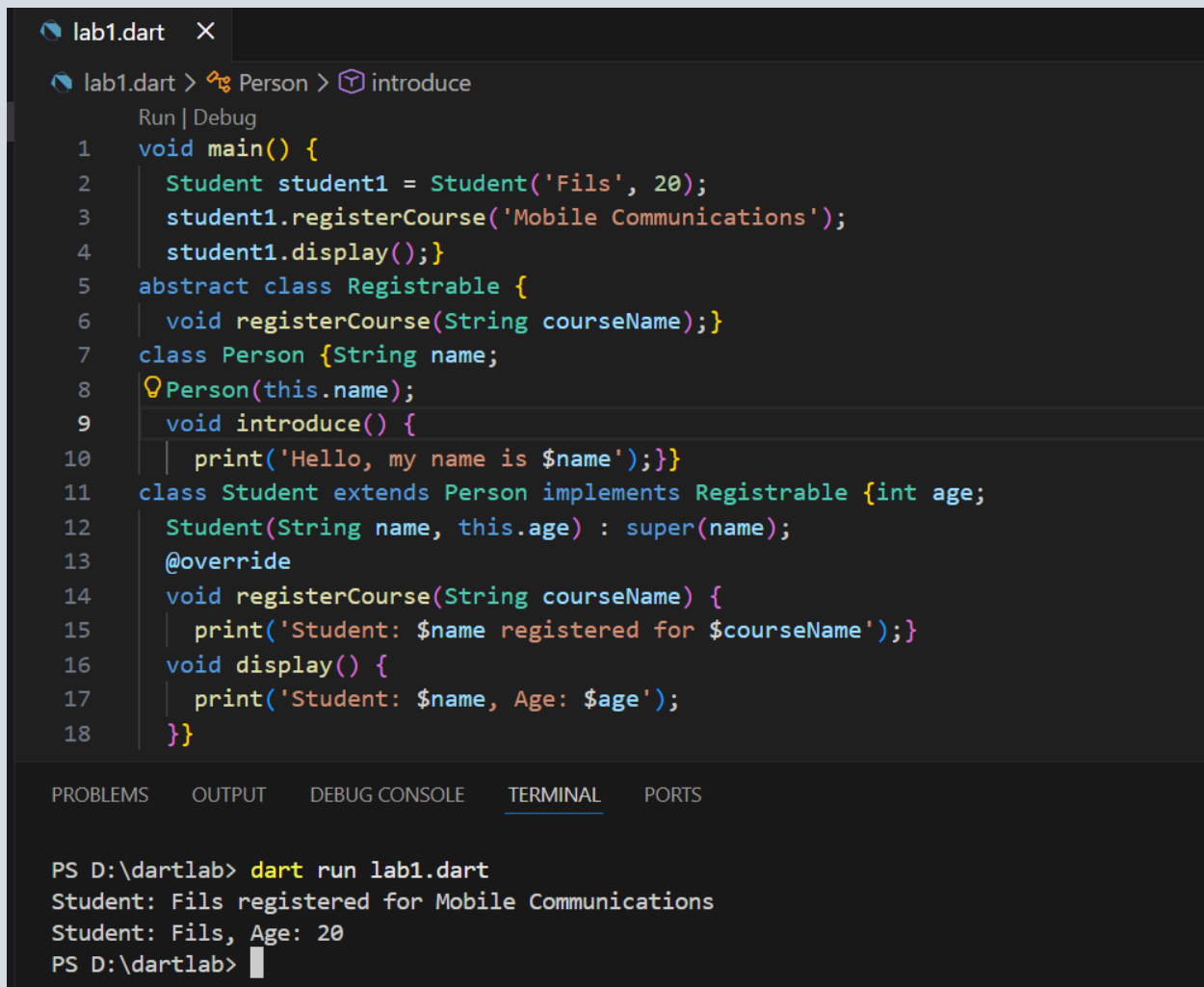
```
PS D:\dartlab> dart run lab1.dart
Hello, myn inherited name is Fils
PS D:\dartlab>
```

Part 4: Interfaces

Q8: Create an abstract class `Registrable` with a function `registerCourse(String courseName)`. **Together combined with**

Q9: Make `Student` implement `Registrable` and implement `registerCourse` to print the student name and course.

Answer screenshot for q8 & q9:



The screenshot shows an IDE window titled 'lab1.dart'. The code defines a `Person` class with an `introduce` method and a `Student` class that extends `Person` and implements the `Registrable` interface. The `main` function creates a `Student` object, registers a course, and displays its details. The terminal output shows the execution results: 'Student: Fils registered for Mobile Communications' and 'Student: Fils, Age: 20'.

```
lab1.dart > Person > introduce
Run | Debug
1 void main() {
2   Student student1 = Student('Fils', 20);
3   student1.registerCourse('Mobile Communications');
4   student1.display();}
5 abstract class Registrable {
6   void registerCourse(String courseName);}
7 class Person {String name;
8   Person(this.name);
9   void introduce() {
10    print('Hello, my name is $name');}}
11 class Student extends Person implements Registrable {int age;
12   Student(String name, this.age) : super(name);
13   @override
14   void registerCourse(String courseName) {
15     print('Student: $name registered for $courseName');}
16   void display() {
17     print('Student: $name, Age: $age');
18   }}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\dartlab> dart run lab1.dart
Student: Fils registered for Mobile Communications
Student: Fils, Age: 20
PS D:\dartlab>
```

Part 5: Mixins

Q10: Create a mixin ``AttendanceMixin`` that stores an attendance counter and has a function ``markAttendance()`` to increase attendance.

Q11: Apply ``AttendanceMixin`` to ``Student``. Mark attendance 3 times and print the attendance.

Answer screenshot q10 & q11:

```
lab1.dart > AttendanceMixin > markAttendance
1  void main() {
4      student1.markAttendance();
5      student1.markAttendance();
6      student1.display();}
7  mixin AttendanceMixin {
8      int attendanceCount = 0;
9      void markAttendance() {
10         attendanceCount++;
11         print('Attendance marked!: $attendanceCount');}
12     int getAttendanceCount() {
13         return attendanceCount;}
14     class Person {String name;
15         Person(this.name);
16         void introduce() {
17             print('Hello, my name is $name');}}
18     class Student extends Person with AttendanceMixin {int age;
19         Student(String name, this.age) : super(name);
20         void display() {
21             print('Student: $name, Age: $age, Attendance: $attendanceCount');}}
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell +


```
PS D:\dartlab> dart run lab1.dart
Attendance marked!: 1
Attendance marked!: 2
Attendance marked!: 3
Student: Alice, Age: 20, Attendance: 3
PS D:\dartlab>
```

Part 6: Collections

Q12: Create a List storing multiple `Student` objects. Add 3 students.

Answer screenshot 12:

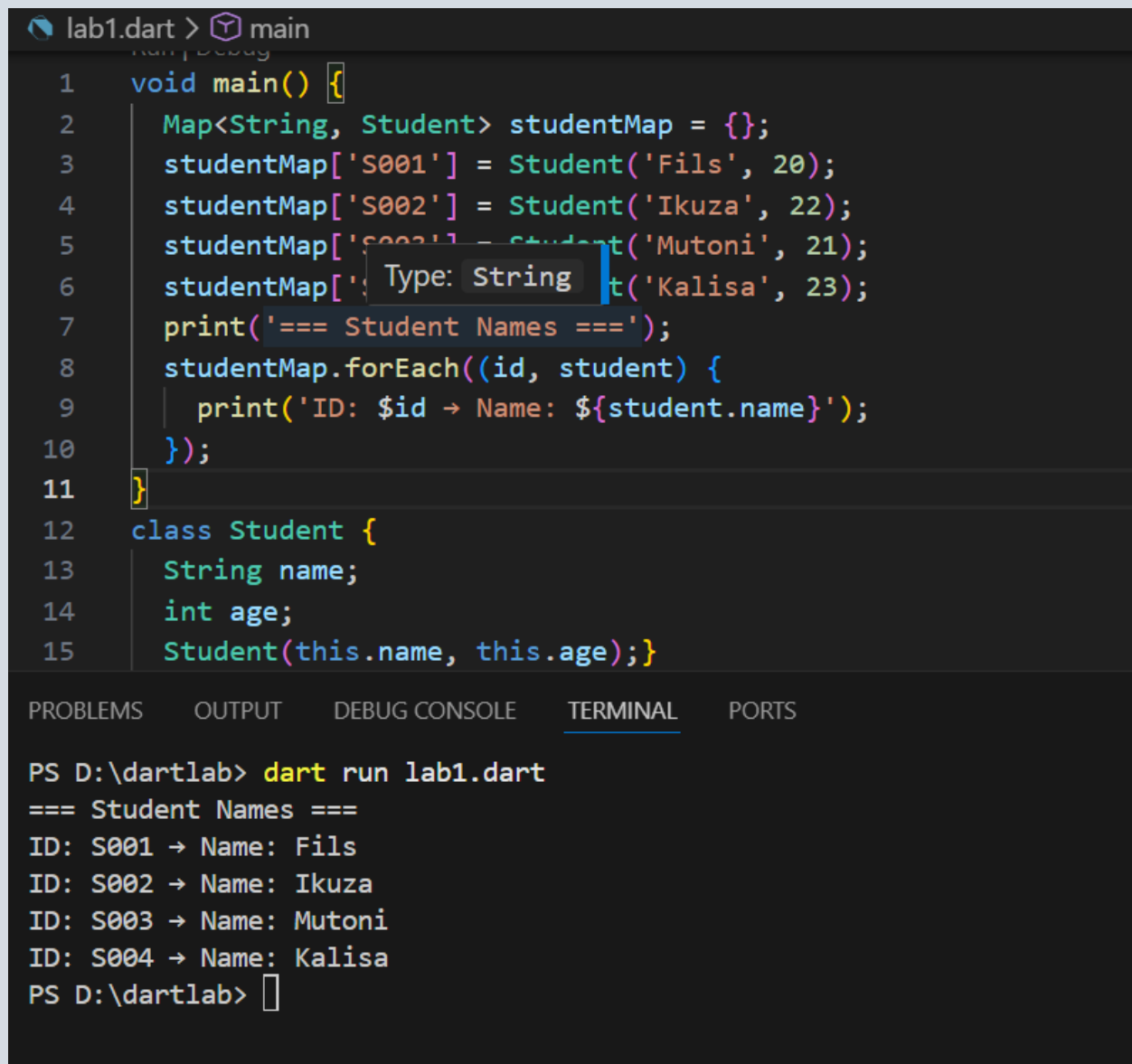

```
lab1.dart >  Student
Run | Debug
1 void main() {
2     List<Student> students = [];
3     students.add(Student('Alice', 20));
4     students.add(Student('Bob', 22));
5     students.add(Student('Charlie', 19));
6     print('=== Student List ===');
7     for (int i = 0; i < students.length; i++) {
8         print('${i + 1}. ${students[i].name}, Age: ${students[i].age}');
9     }
10    print(' Using for-in loop');
11    for (Student student in students) {
12        print('Student: ${student.name}, Age: ${student.age}');
13    }
14    class Student {
15        String name;
16        int age;
17        Student(this.name, this.age);
18    }
19 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  powerline

```
PS D:\dartlab> dart run lab1.dart
=== Student List ===
1. Alice, Age: 20
2. Bob, Age: 22
3. Charlie, Age: 19
 Using for-in loop
Student: Alice, Age: 20
Student: Bob, Age: 22
Student: Charlie, Age: 19
PS D:\dartlab>
```

Q13: Create a Map where the key is student ID and value is a `Student`. Print all student names.

Answer screenshot 13:



The screenshot shows an IDE window with a Dart file named `lab1.dart`. The code defines a `main` function and a `Student` class. The `main` function creates a `Map` of `String` to `Student` objects, iterates over it, and prints the names. The `Student` class has `name` and `age` properties and a constructor. The terminal output shows the execution of the code, displaying the names of the students.

```
lab1.dart > main
1 void main() {
2   Map<String, Student> studentMap = {};
3   studentMap['S001'] = Student('Fils', 20);
4   studentMap['S002'] = Student('Ikuza', 22);
5   studentMap['S003'] = Student('Mutoni', 21);
6   studentMap['S004'] = Student('Kalisa', 23);
7   print('=== Student Names ===');
8   studentMap.forEach((id, student) {
9     print('ID: $id → Name: ${student.name}');
10  });
11 }
12 class Student {
13   String name;
14   int age;
15   Student(this.name, this.age);}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\dartlab> dart run lab1.dart
=== Student Names ===
ID: S001 → Name: Fils
ID: S002 → Name: Ikuza
ID: S003 → Name: Mutoni
ID: S004 → Name: Kalisa
PS D:\dartlab>
```

Part 7: Anonymous and Arrow Functions

Q14: Use an anonymous function to print all student names from the list.

Answer screenshot 14:

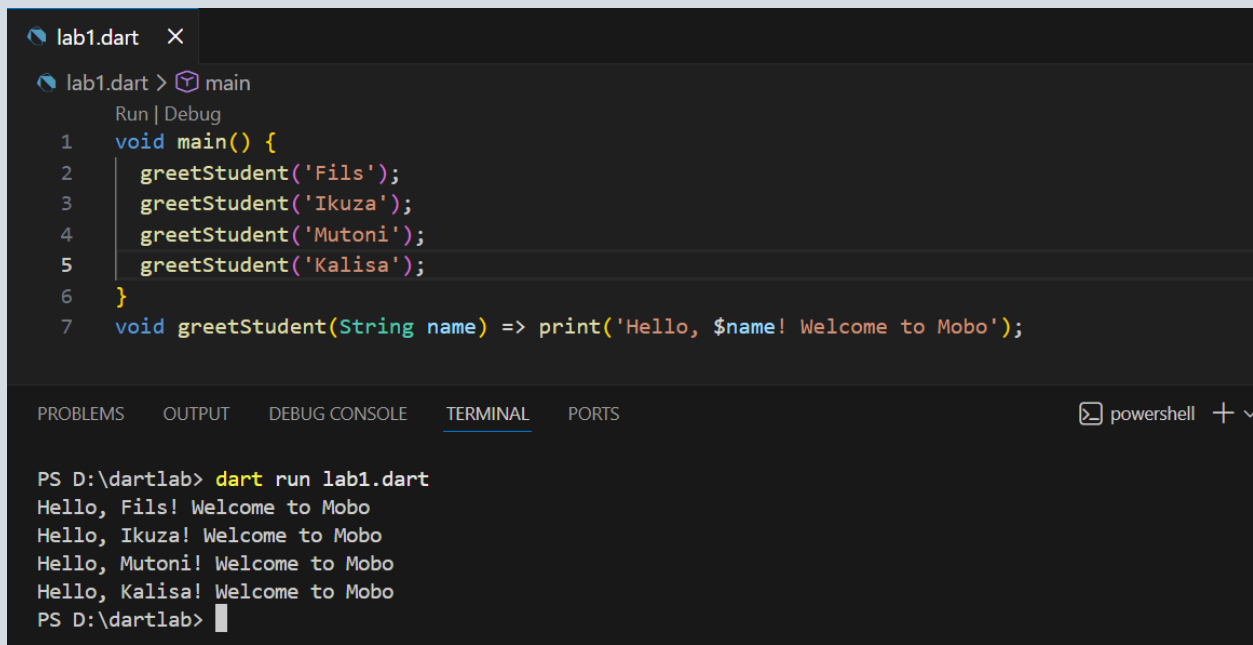
```
lab1.dart > main
Run | Debug
1 void main() {
2     List<Student> students = [
3         Student('Fils', 20),
4         Student('Ikuza', 22),
5         Student('Mutoni', 21),
6         Student('Kalisa', 23),];
7     print('Student Names Anonymouslyare:');
8     students.forEach((Student student) {
9         print(student.name);
10    });}
11 class Student {
12     String name;
13     int age;
14
15     Student(this.name, this.age);
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\dartlab> dart run lab1.dart
Student Names Anonymouslyare:
Fils
Ikuza
Mutoni
Kalisa
PS D:\dartlab> 
```

Q15: Write an arrow function that takes a student name and prints a greeting message.

Answer screenshot15 :



The screenshot shows an IDE window titled 'lab1.dart'. The editor contains the following Dart code:

```
lab1.dart > main
Run | Debug
1 void main() {
2   greetStudent('Fils');
3   greetStudent('Ikuza');
4   greetStudent('Mutoni');
5   greetStudent('Kalisa');
6 }
7 void greetStudent(String name) => print('Hello, $name! Welcome to Mobo');
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS D:\dartlab> dart run lab1.dart
Hello, Fils! Welcome to Mobo
Hello, Ikuza! Welcome to Mobo
Hello, Mutoni! Welcome to Mobo
Hello, Kalisa! Welcome to Mobo
PS D:\dartlab>
```

Part 8: Asynchronous Programming

Q16: Write an async function `loadStudents()` that waits 2 seconds and returns the list of students.

Answer screenshot 16:

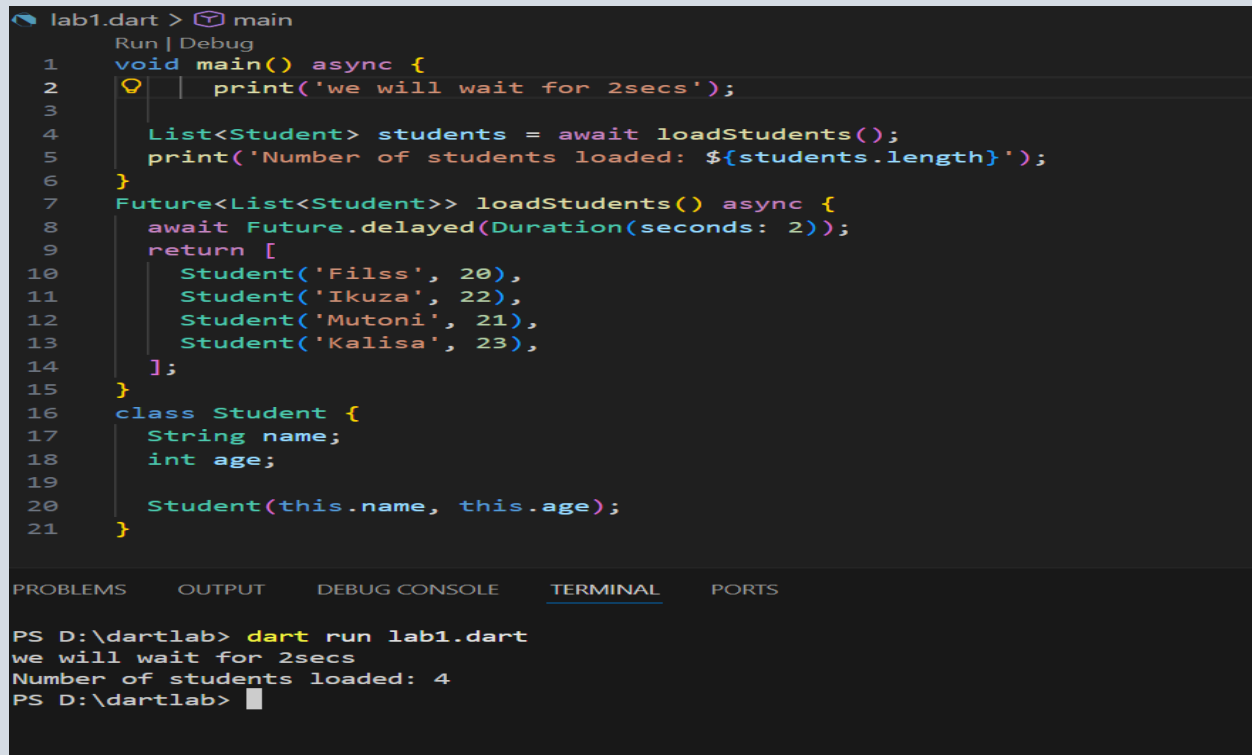
```
lab1.dart > main
Run | Debug
1 void main() async {
2   print('Loading students');
3   List<Student> students = await loadStudents();
4   print(' Students sre loaded now!');
5   for (Student student in students) {
6     print('• ${student.name}, Age: ${student.age}');}}
7 Future<List<Student>> loadStudents() async {
8   await Future.delayed(Duration(seconds: 2));
9   return [
10    Student('Fils', 20),
11    Student('Ikuza', 22),
12    Student('Mutoni', 21),
13    Student('Kalisa', 23),,];}
14 class Student {
15   String name;
16   int age;
17   Student(this.name, this.age);
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + -

```
PS D:\dartlab> dart run lab1.dart
Loading students
 Students sre loaded now!
• Fils, Age: 20
• Ikuza, Age: 22
• Mutoni, Age: 21
• Kalisa, Age: 23
PS D:\dartlab> 
```

Q17: In main(), call `loadStudents()`, use await, and print the number of students loaded.

Answer screenshot 17 :



The screenshot shows an IDE with a Dart file named `lab1.dart`. The code defines a `main` function that prints a message, waits for 2 seconds, and then loads a list of students. A `Student` class is also defined. The terminal output shows the execution of the code, which prints "we will wait for 2secs" and "Number of students loaded: 4".

```
lab1.dart > main
Run | Debug
1 void main() async {
2   print('we will wait for 2secs');
3
4   List<Student> students = await loadStudents();
5   print('Number of students loaded: ${students.length}');
6 }
7 Future<List<Student>> loadStudents() async {
8   await Future.delayed(Duration(seconds: 2));
9   return [
10    Student('Filss', 20),
11    Student('Ikuza', 22),
12    Student('Mutoni', 21),
13    Student('Kalisa', 23),
14  ];
15 }
16 class Student {
17   String name;
18   int age;
19
20   Student(this.name, this.age);
21 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\dartlab> dart run lab1.dart
we will wait for 2secs
Number of students loaded: 4
PS D:\dartlab>
```

Part 9: Integration Challenge

Q19: Create a new mixin ``NotificationMixin`` that prints a message when a student is registered to a course. Apply it to ``Student``.

Answer screenshot 19 :

```
lab1.dart > Person
1 void main() {
2   Student student1 = Student('Fils', 20);
3   student1.registerCourse('Mobo');
4   student1.registerCourse('NetSec');
5   print(' Student Summary ');
6   student1.display();}
7 mixin NotificationMixin {
8   void sendNotification(String studentName, String courseName) {
9     print(' NOTIFICATION: $studentName has successfully registered for $courseName!');
10    print(' Registration confirmed. Welcome to the course!');}}
11 abstract class Registrable {
12   void registerCourse(String courseName);}
13 class Person {String name;
14   Person(this.name);
15   void introduce() {
16     print('Hello, my name is $name!');}}
17 class Student extends Person with NotificationMixin implements Registrable { int age;
18   Student(String name, this.age) : super(name);
19   @override
20   void registerCourse(String courseName) {
21     sendNotification(name, courseName);}
22   void display() {
23     print('Student: $name, Age: $age!');}}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [] [X] ... |

```
PS D:\dartlab> dart run lab1.dart
NOTIFICATION: Fils has successfully registered for Mobo!
Registration confirmed. Welcome to the course!
NOTIFICATION: Fils has successfully registered for NetSec!
Registration confirmed. Welcome to the course!
Student Summary
Student: Fils, Age: 20
```