



**UNIVERSITY OF RWANDA
COLLEGE OF SCIENCE AND TECHNOLOGY
COMPUTER ENGINEERING
YEAR 3
GAKO CAMPUS
DART FOR FLUTTER MOBILE PROMMING**

Reg n°: **223026685**

223027476

223026674

Names:

Jerome HAKIZIMANA

Clement NZABAMWITA

Lewis NGENDAHIMANA

Dart Programming Lab 1

REPORT

Contents

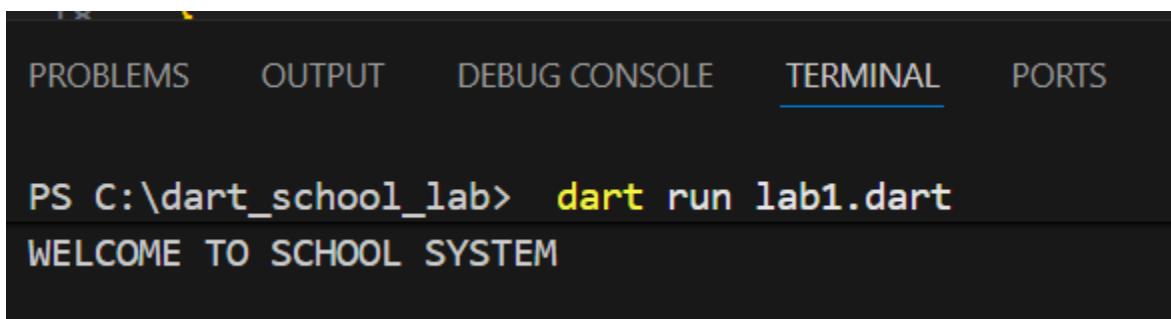
Part 1: Functions	3
Q1: welcomeMessage Function.....	3
Q2: createStudent with Named Parameters	3
Q3: createTeacher with Optional Parameters.....	4
Part 2: Constructors and Classes.....	4
Q4: Student Class with Constructor.....	4
Q5: Object Creation and Usage	4
Part 3: Inheritance	5
Q6: Person Class	5
Q7: Student Inherits from Person	5
Part 4: Interfaces	5
Q8: Abstract Class Registrable	5
Q9: Student Implements Registrable	6
Part 5: Mixins.....	6
Q10: AttendanceMixin.....	6
Q11: Applying AttendanceMixin to Student	6
Part 6: Collections.....	7
Q12: List of Student Objects.....	7
Q13: Map of Students	7
Part 7: Anonymous and Arrow Functions	8
Q14: Anonymous Function.....	8
Q15: Arrow Function	8
Part 8: Asynchronous Programming	9
Q16: Async Function loadStudents().....	9
Q17: Using await in main().....	9
Part 9: Integration Challenge	10
Q18: Mixins vs Inheritance.....	10
Q19: NotificationMixin	11
Q20: How Learning Dart Helps Understand Flutter.....	11
SOURCE CODE.....	11

Lab Report: Dart Programming for Flutter

Part 1: Functions

Q1: welcomeMessage Function

Output:

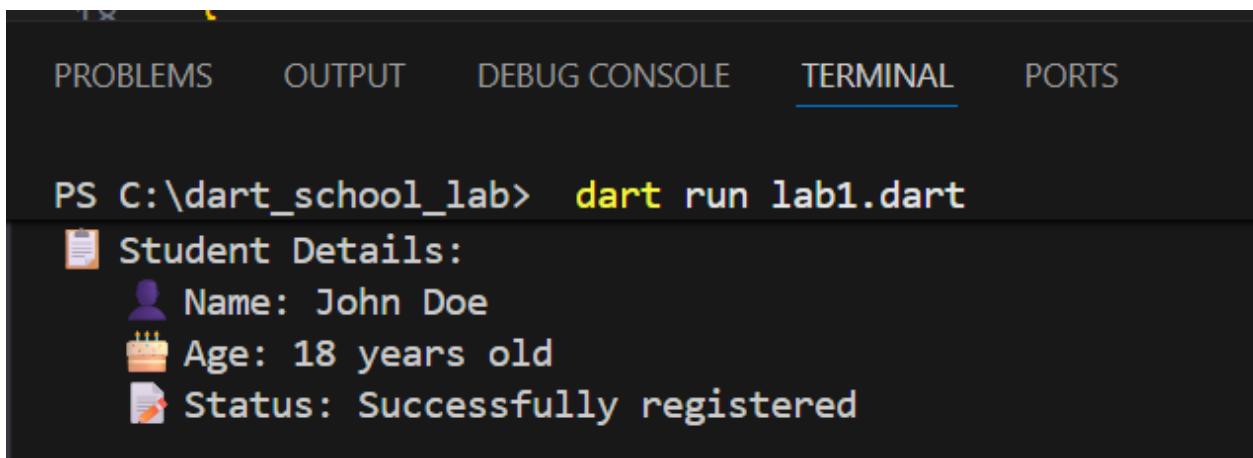


The screenshot shows a terminal window with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS. The terminal content is as follows:

```
PS C:\dart_school_lab> dart run lab1.dart
WELCOME TO SCHOOL SYSTEM
```

Q2: createStudent with Named Parameters

Output:



The screenshot shows a terminal window with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS. The terminal content is as follows:

```
PS C:\dart_school_lab> dart run lab1.dart
📋 Student Details:
👤 Name: John Doe
🎂 Age: 18 years old
📝 Status: Successfully registered
```

Mobile

Q3: createTeacher with Optional Parameters

Output:

```
PS C:\dart_school_lab> dart run lab1.dart
👩🏫 Teacher Registration:
Name: Mr. Anderson
Subject: Mathematics

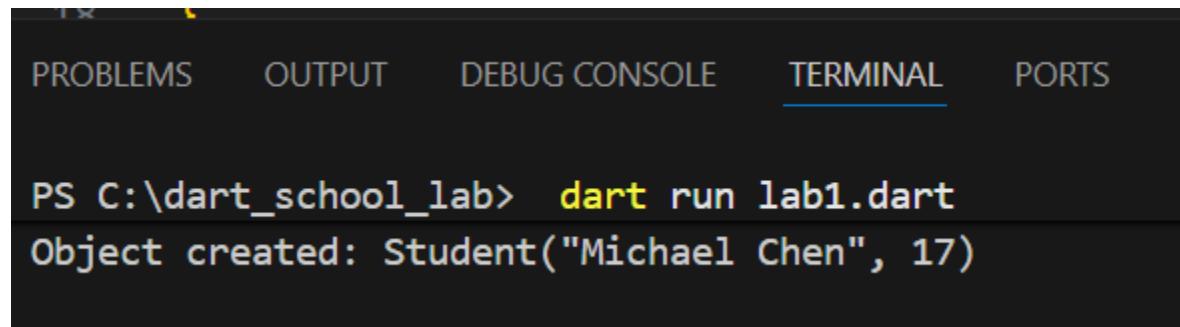
--- Testing without subject ---

👩🏫 Teacher Registration:
Name: Ms. Garcia
⚠️ Subject not assigned
```

Part 2: Constructors and Classes

Q4: Student Class with Constructor

Output:

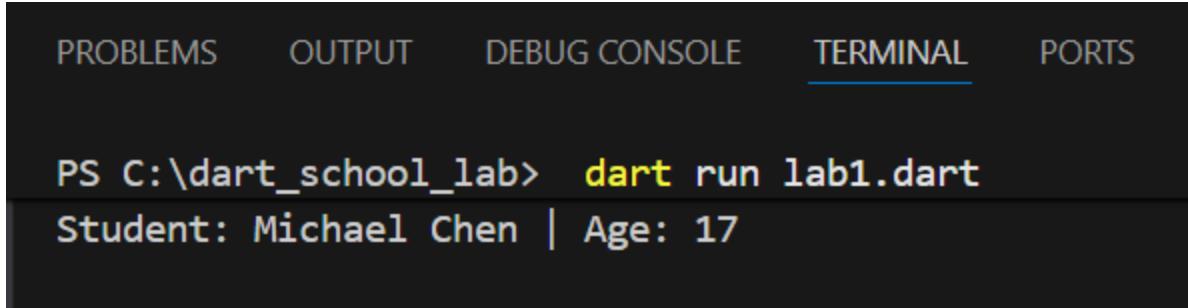


A screenshot of a terminal window in a code editor. The tab bar at the top shows tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined in blue), and PORTS. The terminal window displays the following text:

```
PS C:\dart_school_lab> dart run lab1.dart
Object created: Student("Michael Chen", 17)
```

Q5: Object Creation and Usage

Output:



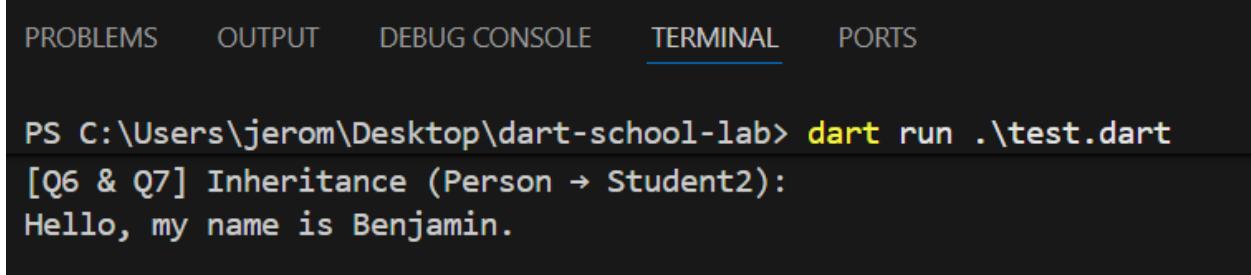
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\dart_school_lab> dart run lab1.dart
Student: Michael Chen | Age: 17
```

Part 3: Inheritance

Q6: Person Class

Output:

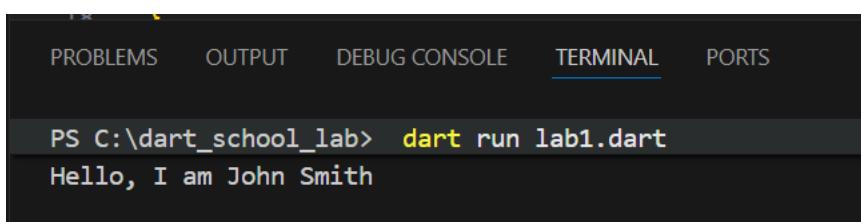


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\jerom\Desktop\dart-school-lab> dart run .\test.dart
[Q6 & Q7] Inheritance (Person → Student2):
Hello, my name is Benjamin.
```

Q7: Student Inherits from Person

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\dart_school_lab> dart run lab1.dart
Hello, I am John Smith
```

Part 4: Interfaces

Q8: Abstract Class Registrable

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dart_school_lab> dart run lab1.dart
Abstract class Registrable defined with registerCourse method
```

Q9: Student Implements Registrable

Q8 & Q9

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dart_school_lab> dart run lab1.dart
✓ Robert Johnson registered for: Physics
Total courses: 1
```

Part 5: Mixins

Q10: AttendanceMixin

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dart_school_lab> dart run lab1.dart
AttendanceMixin created with markAttendance() method
```

Q11: Applying AttendanceMixin to Student

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\dart_school_lab> dart run lab1.dart
Student with AttendanceMixin created: Emma Wilson

Marking attendance 3 times:
  📅 Attendance marked! Total: 1
  📅 Attendance marked! Total: 2
  📅 Attendance marked! Total: 3
Total attendance: 3
```

Part 6: Collections

Q12: List of Student Objects

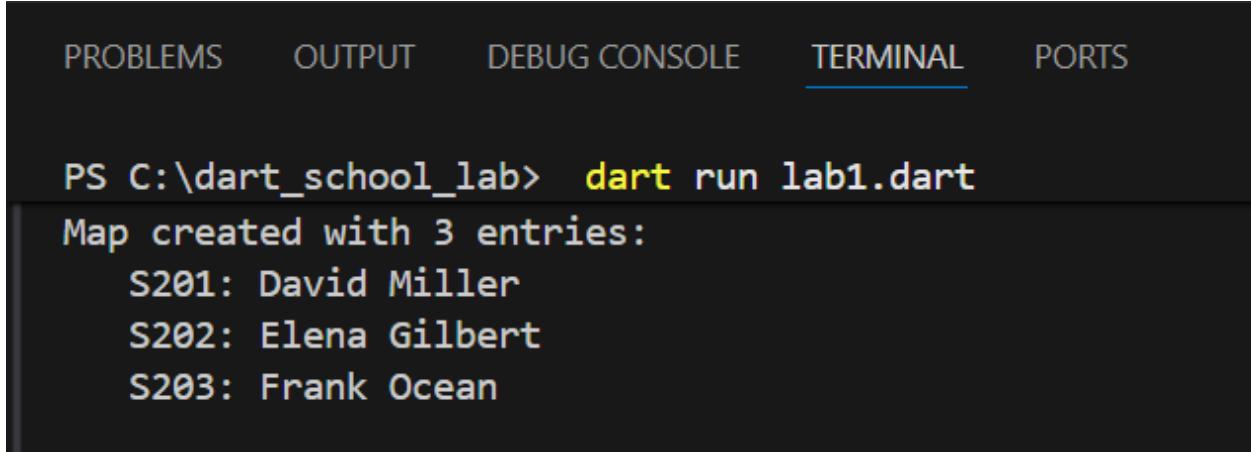
Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\dart_school_lab> dart run lab1.dart
List created with 3 students:
  - Alex Turner (Age: 18)
  - Bella Swan (Age: 17)
  - Chris Evans (Age: 19)
```

Q13: Map of Students

Output:



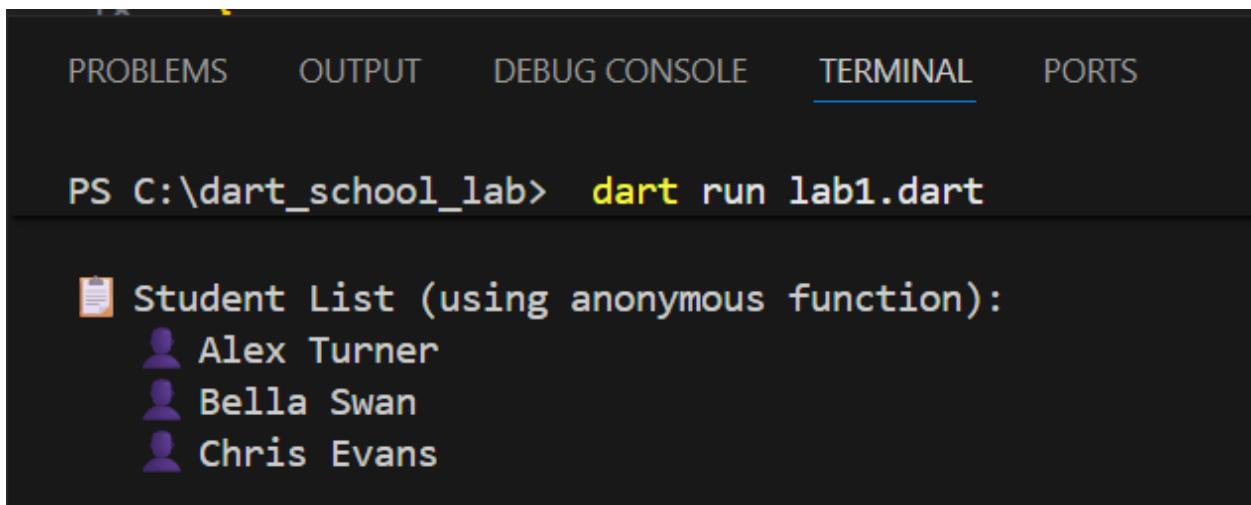
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dart_school_lab> dart run lab1.dart
Map created with 3 entries:
S201: David Miller
S202: Elena Gilbert
S203: Frank Ocean
```

Part 7: Anonymous and Arrow Functions

Q14: Anonymous Function

Output:

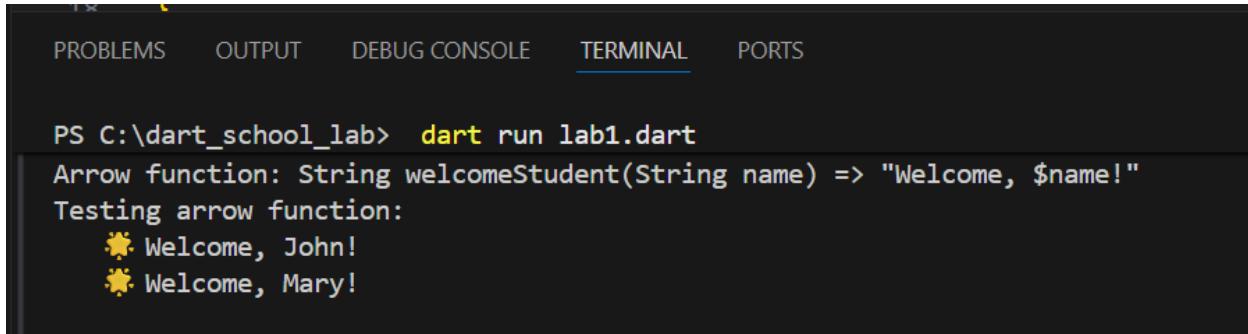


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\dart_school_lab> dart run lab1.dart
Student List (using anonymous function):
  ↗ Alex Turner
  ↗ Bella Swan
  ↗ Chris Evans
```

Q15: Arrow Function

Output:



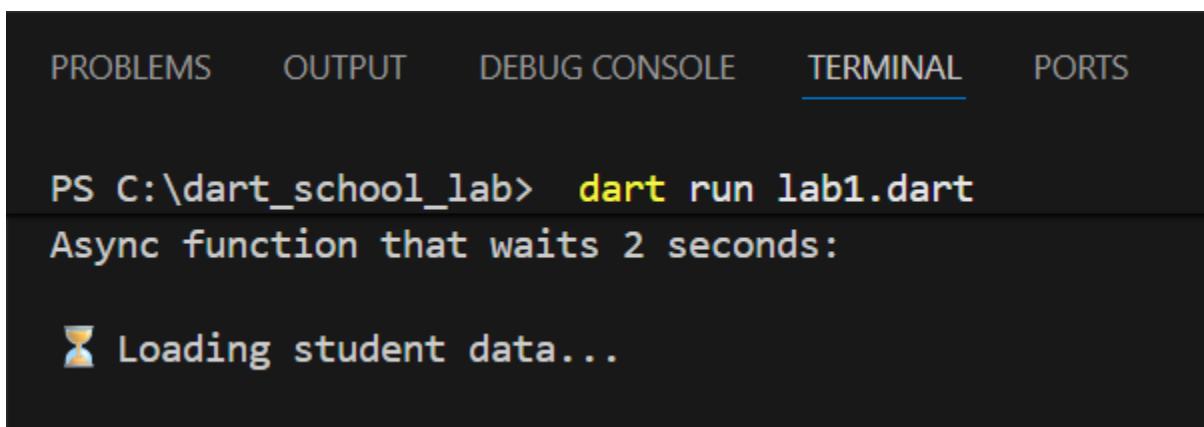
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\dart_school_lab> dart run lab1.dart
Arrow function: String welcomeStudent(String name) => "Welcome, $name!"
Testing arrow function:
  ★ Welcome, John!
  ★ Welcome, Mary!
```

Part 8: Asynchronous Programming

Q16: Async Function loadStudents()

Output:



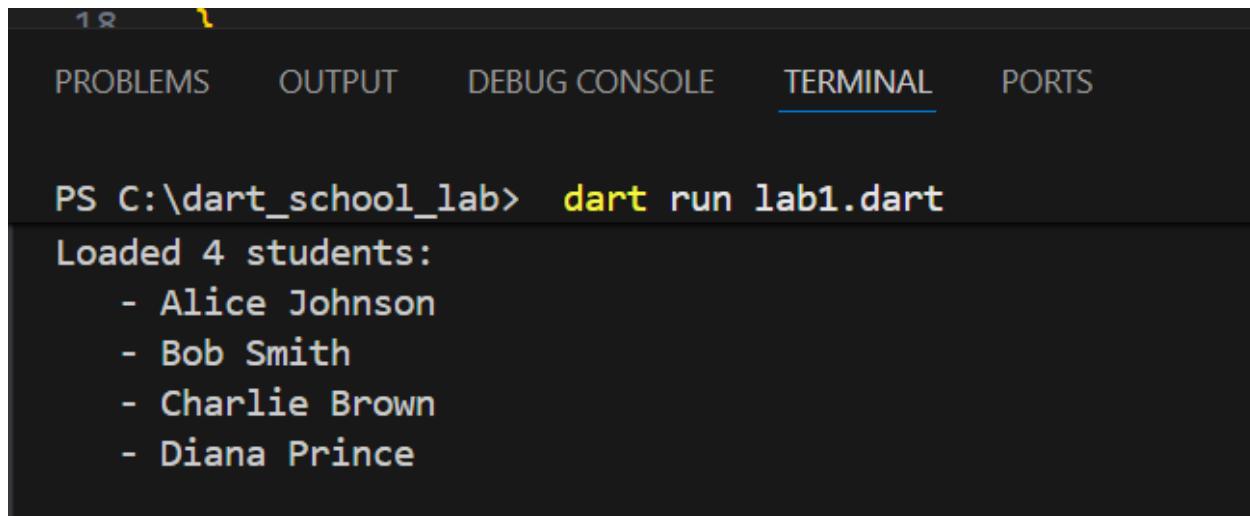
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\dart_school_lab> dart run lab1.dart
Async function that waits 2 seconds:

  ⏳ Loading student data...
```

Q17: Using await in main()

Output:



A screenshot of a terminal window from a code editor. The tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal output shows the command `PS C:\dart_school_lab> dart run lab1.dart` and its response: "Loaded 4 students:" followed by a list of student names: Alice Johnson, Bob Smith, Charlie Brown, and Diana Prince.

```
PS C:\dart_school_lab> dart run lab1.dart
Loaded 4 students:
- Alice Johnson
- Bob Smith
- Charlie Brown
- Diana Prince
```

Part 9: Integration Challenge

Q18: Mixins vs Inheritance

Mixins are a powerful Dart feature that allows reusable code to be shared across multiple class hierarchies without establishing an “is-a” relationship.

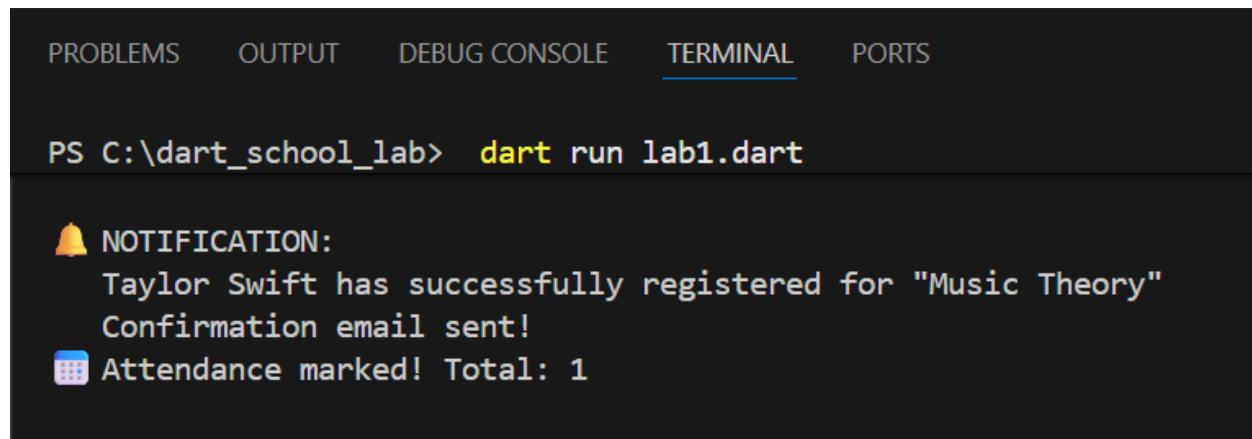
They are useful for adding common behavior such as logging, notification, or attendance tracking to classes that are otherwise unrelated.

Unlike inheritance, which creates a strict parent-child chain and is limited to a single superclass, mixins can be applied to any class using the `with` keyword, and a class can use several mixins at once.

Inheritance defines what an object *is* (e.g., a Student is a Person), while mixins define what an object *can do* (e.g., a Student can send notifications). Mixins promote composition over inheritance, keeping class hierarchies simple and avoiding the problems of deep or multiple inheritance.

They encapsulate cross-cutting concerns, making code more modular, maintainable, and reusable.

Q19: NotificationMixin



The screenshot shows a terminal window with the following interface:

- Top bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS.
- Output area:
 - Command: PS C:\dart_school_lab> **dart run lab1.dart**
 - Notification message:
 - 🔔 NOTIFICATION:
 - Taylor Swift has successfully registered for "Music Theory"
 - Confirmation email sent!
 - 📅 Attendance marked! Total: 1

Q20: How Learning Dart Helps Understand Flutter

Learning Dart is essential for understanding Flutter because Flutter is built entirely on Dart—every Flutter application is fundamentally a Dart application.

The concepts I practiced in this lab translate directly to Flutter development: classes define widgets, constructors configure widget properties using named parameters, inheritance allows creating custom widgets that extend Flutter's built-in widgets, and mixins add lifecycle methods or common behaviors to widgets.

When I fetch data from an API in Flutter, I use the same `async/await` syntax I learned here. When I display lists of data, I work with Dart collections like `List` and `Map`. Anonymous functions become event handlers for button presses, and arrow functions provide concise callbacks.

Without understanding Dart, Flutter widgets would seem like magic; with Dart knowledge, I recognize that Flutter is simply a sophisticated framework leveraging Dart's object-oriented and asynchronous features.

This lab has given me the confidence to move from console programs to building real mobile apps, knowing that the same language rules and programming patterns apply seamlessly.

SOURCE CODE 

GITHUB LINK: <https://github.com/sticker2024/dart-school-lab.git>