Dart Lab Exercise: Variables, Conditional Structures, Loops, Functions, OO Concepts and Streams

Chaouki BAYOUDHI

October 9, 2023

Introduction

In this lab exercise, you will explore Dart programming by working on conditional structures, loops, Object-Oriented (OO) concepts, and explicit exercises. Dart is a versatile programming language commonly used for building web and mobile applications. This exercise will provide hands-on experience with these essential aspects of Dart programming.

Prerequisites

Before you begin, ensure that you have the Dart SDK installed on your machine. You can download it from the official Dart website: `https://dart.dev/get-dart`

Lab Tasks

Task 1: Conditional Structures and Loops

Create a Dart program that demonstrates the use of conditional structures (if-else statements) and loops (for and while loops) with the following requirements:

1. Implement a function that checks whether a given number is even or odd using an if-else statement.
2. Use a for loop to print the first 10 Fibonacci numbers.
3. Use a while loop to find and print all prime numbers between 1 and 50.

Task 2: Dart Functions

In this task, you will work with Dart functions.

Part 1: Function Declaration

Create a Dart function called `calculateArea` that calculates and returns the area of a triangle. The function should take two parameters: `base length` and `height`.

Part 2: Function with Default Parameters

Create a Dart function called `printMessage` that takes a message as a parameter and an optional parameter `repeatCount` (default value: 1). The function should print the message the specified number of times.

Part 3: Function as a First-Class Citizen

Implement a function called `performOperation` that takes two numbers and a function as parameters. The function should perform the specified operation on the two numbers and return the result. You can pass built-in operations like addition, subtraction, multiplication, or custom operations as functions.

Part 4: Recursive Function

Write a recursive function called `calculateFactorial` that calculates the factorial of a given non-negative integer.

Task 2: Object-Oriented Concepts

In this task, you will work with Object-Oriented (OO) concepts in Dart.

Part 1: Class Definition

Create a class called `Person` with the following attributes:

- `first_name` (int)
- `last_name` (String)
- `email` (String)
- `birthdate` (date)

Add a parameter constructor to the class and implement methods within the class for setting and getting these attributes.

Part 2: Inheritance

Create a subclass called `Student` that extends the `Person` class. Add additional attributes such as `studentId` and `major`. Implement methods for setting and getting these attributes within the `Student` class.

Part 3: Abstract Class

Create an abstract class called `Employee` with attributes like `name`, `age`, and `email`. Implement abstract methods for calculating salary. Create two concrete subclasses (`Manager` and `Developer`) that extend the `Employee` class and implement their own salary calculation methods.

Part 4: Polymorphism

Instantiate several `Person`, `Student`, `Manager`, and `Developer` objects. Demonstrate polymorphism by calling methods on these objects and printing their attributes.

Part 5: Questions

Answer the following questions related to OO concepts:

1. What is inheritance, and how is it used in Dart?
2. Explain the concept of an abstract class. When and why would you use it?
3. How does polymorphism contribute to code flexibility and reuse in object-oriented programming?

Task 3: Streams

Write a Dart program that uses streams to process a list of numbers. Create a stream controller, add numbers to the stream, and implement a stream listener that calculates the sum of the numbers as they are added to the stream. Print the cumulative sum to the console.

Conclusion

This lab exercise has provided you with hands-on experience in Dart programming, covering conditional structures, loops, Object-Oriented (OO) concepts including inheritance, abstract classes, and polymorphism, as well as explicit exercises to challenge your skills. These fundamental skills are essential for developing robust Dart applications.