Async and await in dart

Exercise #4

Flutter Developer Bootcamp

# **Purpose**

This Exercise demonstrates asynchronous programming in Dart using Futures and the async/await syntax and to handle asynchronous operations in Dart, ensuring that code execution continues without blocking while waiting for asynchronous tasks to complete.

**Problem**

In the provided Exercise You need to create as follows:

Simulates fetching data from a network asynchronously, Use async/await to handle the asynchronous operation cleanly, printing "Program started", then "Fetching data...", followed by either the received data or an error message, and finally "Program ended".

**How to Solve**

1. Checkout the Exercise from Git Repo:

git clone -b <user-branch> <repo-URL>

2. Open the root folder inside VS Code

3. Open the root folder in terminal

4. Run the command dart run filename.dart

5. Imports:

* + The code imports the dart:async library, which provides classes and functions for asynchronous programming.

6. Main Function (main()):

* + Print "Program started" to indicate the start of the program.
  + Calls the fetchData() function.
  + Prints "Program ended" after fetchData() call.

7. fetchData() Function:

* + Declared as async, indicating it performs asynchronous operations.
  + Prints "Fetching data..." to indicate data retrieval is in progress.
  + Uses a try-catch block:
    - Awaits the result of simulateNetworkRequest() using await.
    - If successful, prints the received data.
    - If an error occurs during the asynchronous operation, catches the error and prints an error message.

8. simulateNetworkRequest() Function:

* + Return a Future<String>, indicating it will eventually produce a String result asynchronously.
  + Simulates a network request delay using Future.delayed, which resolves after 2 seconds.
  + When the Future resolves, it returns a string containing simulated data.

9. Go To File: <specific-file-with-async and await-method> à <method-name>, implement your logic.

**You will Achieve**

When you complete this Exercise you will learn the following:

* **Asynchronous Data Fetching:** Simulate fetching data from a network asynchronously, mimicking a real-world scenario where data retrieval takes time.
* **Error Handling:** Implement error handling to manage potential failures during the asynchronous operation, ensuring graceful handling of errors.

**Here are the methods and functions included:**

**1.main() Function:**

* Entry point of the program.
* Prints "Program started".
* Calls fetchData().
* Prints "Program ended".

**2.fetchData() Function:**

* Asynchronous function (async keyword).
* Prints "Fetching data..." to indicate the start of data retrieval.
* Uses await to wait for the result of simulateNetworkRequest().
* Handles any errors that occur during the asynchronous operation using a try-catch block.
* If successful, prints the received data.
* If an error occurs, prints an error message.

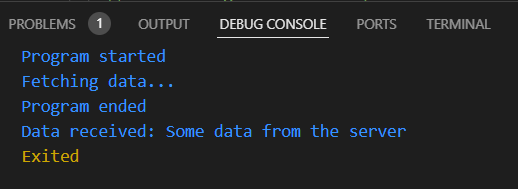
**3.simulateNetworkRequest() Function:**

* Returns a Future<String>, indicating that it will produce a string result asynchronously.
* Uses Future.delayed to simulate a network request delay of 2 seconds.
* When the delay is complete, it returns a string containing simulated data.

Overall, this exercise demonstrates the use of asynchronous programming in Dart, including the use of Futures, async/await syntax, and error handling mechanisms.

# **Screenshots**

## **Expected output (using async and await)**



## 

# **How to submit your Exercise**

Push your project back to the same git branch using command:

<command name>

# **Happy Coding!**