Mixins

Workshop #1

Flutter Developer Bootcamp

# **Purpose**

This workshop demonstrates how to use a mixin called LoggingMixin to provide logging functionality to multiple classes in Dart programming language.

# **Problem**

In the given workshop, there is class named A which uses Logging mixin and it also have instances of class using Logging mixin. You need to create class B and instances of a class B and also create the shared Logging functionality.

# **How to Solve**

1. Checkout the workshop 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. Create class B and instances of class B using mixins.

6. Additionally create Logging functionality.

7. Go To File: <specific-file-with-mixins-method> à <method-name>, implement your mixin logic.

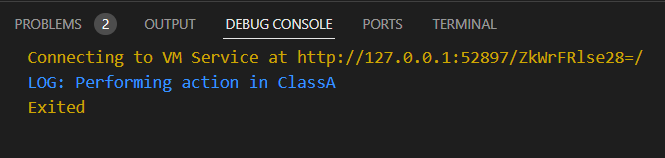
**You will Achieve**

When you complete this workshop you will learn the following:

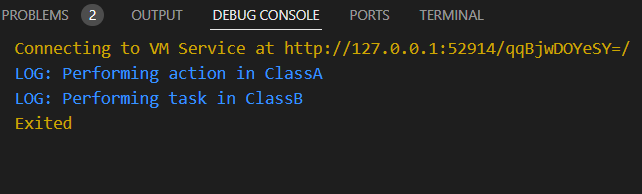
* **Mixin (LoggingMixin):**
* **Function:** logMessage(String message): This function logs a message to the console. It takes a string message as a parameter and prints it prefixed with "LOG:".
* **Class :**
* **ClassA:**
* **Method:** performAction(): This method belongs to ClassA and represents some action performed by objects of ClassA. Inside this method, it calls the logMessage method inherited from the LoggingMixin to log a message specific to ClassA actions.
* **ClassB:**
* **Method: performTask():** This method belongs to ClassB and represents some task performed by objects of ClassB. Inside this method, it calls the logMessage method inherited from the LoggingMixin to log a message specific to ClassB tasks.
* **Main Function:**
* Creates instances of ClassA and ClassB.
* Calls the performAction() method on the instanceA object of ClassA, which in turn logs a message related to the action performed by ClassA.
* Calls the performTask() method on the instanceB object of ClassB, which in turn logs a message related to the task performed by ClassB.
* So, the code demonstrates the usage of a mixin to provide shared functionality (logging) to multiple classes (ClassA and ClassB) without code duplication. It logs messages specific to actions/tasks performed by each class instance.

# **Screenshots**

## **Before implementation (without creating Class B and logging functionality )**



## **After implementation (With creating Class B and logging functionality)**



# **How to submit your workshop**

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

<command name>

# **Happy Coding!**