Lab 4 – Android Studio: App Development  
Andrei Koshelev  
100894383  
  
3101-02  
INFT  
  
Durham College  
Mohammad Shamas  
  
  
Friday, November 1st, 2024

**Table of Contents:**Cover Page …………………………………………….............................................. p. 1  
Table of Contents ........................................................................................................ p. 2  
Q1: Topic Overview ............................................................................................. pp. 3 - 6

Definition ............................................................................................................... p. 3

How and why the Objects are used in Programming……….................................. p. 3

Comparison Between Objects and Future....................................................... pp. 3 - 4

Advantages and Limitations of Using Objects in Dart…............................... pp. 5 - 6

Q2: Example Implementation............................................................................... pp. 5 - 6

#1 Code Example ........................................................................................... pp. 6 - 7

#2 Code Example............................................................................................ pp. 7 - 8

Q3: Real-World Scenario...................................................................................... pp. 8 - 9

References ................................................................................................................. p. 10

**Q1: Topic Overview**  
**Objects – Dart programming Language**

**Definition**  
 Dart is a programming language in which an object is a class unit that represents a particular thing and bundles data, the attributes in this case, and the functions associated with it, the methods (Dart-Tutorial.com, 2022). Objects are defined as the primary objective in implementing object-oriented mechanisms utilized in Dart programming, where data and functions related to particular entities are contained in these basic components. All these are reflected in the understanding that every object is an example of a class with strongly defined properties and methods.  
 **How and why the Objects are used in Programming**  
 Utilizing object-oriented features, Dart supports Flutter by aiding in the design of fast and interactive UI systems that are also easy to edit and manage (Mistry, 2024). Objects are the building blocks to enable a systematic way of coding for developers, to build highly complex systems using modular and reusable code which is close to real structures.  
 **Comparison Between Objects and Future**   
 In Dart, a Future is a crucial part of the software. It is responsible for the correct operation of asynchronous provisions, which are the values and results that might not be available right now (Dart, 2024). It is particularly useful especially during the use of applications to attend to programs that might cause a delay. Unlike objects that represent the entities structured of different elements, objects of a Future are specially designed for the processing of asynchronous data and can allow the program to even carry out other functions when it is waiting for some output.

**• Purpose and Use:** While in Dart programming language objects characterize entities with the behavior and state there is a Future regarding values and missed values. Despite that it is especially useful in Flutter applications whereby the data has to be gotten from a server or the database and still let the main thread free. In this instance Objects enable user access to properties and functions right away With Future meanwhile one can do delayed answers and respond on user’s input much later in the code (Dart, 2024).

**• Structure and Functionality:** An object combines data and procedures which are together called behaviors into one envelope. In the meantime, Future uses techniques like .then() .catchError() and .whenComplete() to manage the response conditions of a successful operation, error condition and the state of the operation. Also, in the scope of Dart, there are keywords such as async and await for more legible handling of the Future object—legible in that it can be used to provide a very clear picture of the execution of a series of asynchronous events that would appear synchronous in nature (Dart, 2024).  
**• Use Case Example:** In an app created using Flutter technology, calling an API to obtain content for the users of the app can be easily done using a Future so that data is obtained from the server in the background and retained for future use. When loading user information from the server a certain entity, in this case a model such as a user could be set up with all properties of the response data e.g. the name and the email. In this case, the association ensures the object manager’s aftereffects once the service findings are released.

Despite their fundamental differences, static objects and Futures support different features in a system. Whereas Objects are classes that encapsulate data and information of a particular entity, Future allows the application to execute the necessary tasks in the background avoiding blocking threads which enables the system to be responsible during execution.  
 **Advantages and Limitations of Using Objects in Dart  
Advantages:**

**• Async Programming Support:** Object-oriented design supported in Dart programming language is very friendly to the concept of asynchronous programming that is a core requirement for the applications in real-time mode (Rasmussen, 2020).

**• Modularity and Reusability:** An encapsulation feature that aids the developer in making efficiency and order within code, which is an important component for touch based applications built in Flutter (Dart-Tutorial.com, 2022).  
**• Null Safety:** Within the class hierarchy, the Dart language introduces a type system that can include null safety, aiming to reduce the errors due to null references in objects which improves the safety and reliability of applications (Rasmussen, 2020).  
**• Enhanced Organization:** Splitting user interface components and procedural logic into classes allows for better code order which is excellent in the context of programming especially when dealing with rich internet applications (RIA) (Mistry, 2023).

**Limitations:**

**• Verbose Syntax:** For simple operations, the hierarchical chain of dart language part does not work very well, because there is a requirement to define classes even when the number of its elements is not high (Mistry, 2023).

**• Single Inheritance Constraint:** This is unlike the practice of some programming languages, which allow for multiple inheritance in situations where Dart only supports inheritance from one parent class (Rasmussen, 2020).  
**• Object Overhead:** Having too many objects can be very taxing to the system, particularly in constrained environments such as mobile web where the system resources are minimized by default. (Dart-Tutorial.com, 2022).

**Q2: Example Implementation** **#1 Code Example:**  
A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated **#2 Code Example:  
A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**  
 **Q3: Real-World Scenario** The Record object in a Flutter application for student management at the academic facility can stand in for each student's academic profile, which includes their name, student ID, program name, academic year, semester, and GPA. Students can view their profiles within the app, and administrators can verify if students are eligible for various scholarships or programs. The application uses Record objects to keep data organized and functionality encapsulated, which makes the code more scalable, modular, and maintainable.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 **References:**  
Dart Dev. (2024). *The impact of mobile computing on individual behavior*. https://dart.dev/libraries/async/async-await  
Dart Tutorial. (2024). *Object-oriented programming: Object in Dart*.   
 https://dart-tutorial.com/object-oriented-programming/object-in-dart/  
Mistry, K. (2024, June 17). *Object-oriented programming in Dart*. Medium.   
 https://medium.com/@kavyamistry0612/object-oriented-programming-in-dart- 7755483131a0  
Rasmussen, M. (2020). *Creating objects and classes in Dart and Flutter*. Dart Academy.   
 https://dart.academy/creating-objects-and-classes-in-dart-and-flutter/