Dart Programming Essentials

**1 : Explain the fundamental data types in Dart (int, double, String, List, Map, etc.) and their uses.**

**Ans :**

*Dart is a statically typed language, meaning variables have specific types.*

*1 : int(Integer)*

*-> Integer is use to store numerical values like 1,2,3,4,5….*

*->* ***integer store******64-bit signed values***

***-> EX: int a = 25;***

*2 : double*

*-> double is use to store decimal numbers like 4.1564*

*->* ***double store******64-bit floating point***

***-> EX: double d = 3.142;***

*3 : String*

*-> String is use to store sequence of characters like “name”*

*->* ***Can be declared using single (' ') or double (" ") quotes.***

***-> EX: String str = “name”;***

*4 : List(Array)*

*-> A* ***collection of ordered elements like [1,3,4]***

*->* ***Can store heterogeneous data***

***-> EX: List fruits = ["Apple", "Banana", "Cherry"];***

*5 : Map*

*->* ***Dictionary-like*** *data structure*

*->* ***Stores values mapped to unique keys. Keys can be any data type (commonly String or int)***

***-> EX: Map stu = {“name” : 1,***

***“name2”: 2 };***

**2. Describe control structures in Dart with examples of if, else, for, while, and switch.**

**Ans :**  *Control structures help manage the flow of execution in Dart programs. The main types are conditional statements (if, else, switch) and loops (for, while, do-while).*

*-> 1 : If else :*

*Ex:*

*void main() {*

*int number = 10;*

*if (number > 0) {*

*print("The number is positive.");*

*} else if (number < 0) {*

*print("The number is negative.");*

*} else {*

*print("The number is zero.");*

*}*

*}*

*-> 2 : Switch*

*Ex:*

*void main()*

*String grade = "B";*

*switch (grade) {*

*case "A":*

*print("Excellent!");*

*break;*

*case "B":*

*print("Good job!");*

*break;*

*case "C":*

*print("You can improve.");*

*break;*

*default:*

*print("Invalid grade.");*

*}*

*}*

*-> 3 : for loop*

*Ex :*

*void main() {*

*for (int i = 1; i <= 5; i++) {*

*print(" $i");*

*}*

*}*

*-> 4 : While loop*

*Ex:*

*void main() {*

*int count = 0;*

*while (count < 3) {*

*print("Count: $count");*

*count++;*

*}*

*}*

**3 : Explain object-oriented programming concepts in Dart, such as classes, inheritance, polymorphism, and interfaces.**

**Ans :**

**1. Class and Object**

**-> A class is a blueprint that defines the properties (variables) and behaviors (methods) of an object.**

**-> An object is an instance of a class that holds specific values and allows interaction with defined methods.**

**-> Classes help in structuring code by grouping related functionality, making development more efficient and organized.**

**2. Encapsulation :**

**-> Encapsulation is the principle of restricting direct access to certain data within an object.**

**-> This is achieved by using private variables (define with \_) and exposing them through getters and setters.**

**-> It improves data security and maintainability by ensuring controlled access to an object's internal state.**

**3. Inheritance :**

**-> Inheritance allows a child class to use properties and methods from a parent class, reducing code duplication.**

**-> It establishes a hierarchical relationship, where the subclass extends the superclass to enhance or override functionality.**

**-> The super keyword enables calling the parent class’s constructor or methods within the child class.**

**-> dart does not support multiple and hybrid Inheritance.**

**4. Polymorphism :**

**-> Polymorphism enables different classes to be treated as instances of the same superclass while having their unique method implementations.**

**-> It allows method overriding, where a subclass provides its own implementation of a method inherited from the parent class.**

**-> This enhances flexibility and extensibility, making it easier to modify or extend functionalities without altering the existing codebase.**

**5. Abstract Classes & Interfaces :**

**Abstract Classes**

**-> An abstract class serves as a template for other classes and cannot be instantiated directly.**

**-> It may contain both abstract methods (without implementation) and concrete methods that subclasses can inherit.**

**-> Abstract classes help enforce consistent structure across related classes.**

**Interfaces :**

**-> Dart does not support multiple inheritance, but interfaces allow classes to implement multiple behaviors.**

**-> An interface is a contract that specifies method signatures but does not provide implementations.**

**-> A class that implements an interface must define all its methods, allowing modular and flexible design.**