Module - 2

PAGE NO.:
DATE: / /

Dart Programming Essentials.

1. Explain the fundamentals data types in Dart (int, double, string, List, Map, etc.) and their uses.

Fundamental Data Types in Dart:

(1.) int: Whole number (without decimal point).

Example: 10, -5, 100

Use: When we want to store numbers like age, quantity, marks, etc.

```
int age = 20;
```

(2.) double: Decimal numbers (numbers with a point).

Example: 3.14, -0.5, 124.0

Use: For values like price, height, weight, etc.

```
double pi = 3.14;
```

(3.) String: A sequence of character (text).

Example: "Hello", "abc123"

Use: To store names, messages, or any

kind of text.

(4.) bool (Boolean): True or false.

Example: true, false

Use: For conditions, like checking if something in on/off, in valid/invalid, etc.

bool isLoggedIn = true;

(5.) var: A keyword that automatically decides the data type based on the value we assign.

Use: When we want ~~to~~ ~~ast~~ Dart to figure out the type for us.

var city = "Rajkot"; //Dart understands
//it's a String

(6.) List: A collection of values (like an array).

Example: [1, 2, 3], ["apple", "banana"]

Use: When you we want to store multiple items together.

List<String> fruits = ["apple", "banana", "Mango"

(7.) Set: A collection like a list, but no duplicates are allowed, and order doesn't matter.

Example: {"Red", "Green", "Blue"}, "Red"};

Use: When we want only unique items.

Set<int> a = {1, 2, 3, 4, 2};

(8.) Map: Stores data in key-value pairs.

Example: {"name": "Vani", "course": "Flutter"}

Use: When we want to look things up quickly, like a dictionary.

```
Map<String, int> marks = {
    "Math": 90,
    "English": 85
};
```

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

(1.) if statement: Used to check a condition. If it's true, it runs the code inside.

Example:

```
int age = 20;
if(age >= 18) {
    print("You can vote");
}
```

(2.) if...else statement: If the condition is true, run the first block. Otherwise runs the second block.

Example:

```
~~int marks~~
int age = 20;
if(age >= 18) {
    print("You can vote");
} else {
    print("You can't vote");
}
```

(3.) if...else...if...else: Used when we have multiple conditions to check.

Example:

```
int marks = 85;
if (marks >= 90) {
    print("A Grade");
} else if(marks >= 80) {
    print("B Grade");
} else {
    print("C Grade");
}
```

(4.) for loop: Used when we want to run a block of code a certain number of times.

Example:

```
for (int i = 1; i <= 5; i++) {
    print("Number: $i");
}
```

(5.) while loop: Keeps running as long as the condition is true.

Example:

```
int i = 1;
while (i <= 5) {
    print("Hello");
    i++;
}
```

(6.) do... while loop: Like while, but the code runs at least once before checking the condition.

Example:

```
int i = 1;
do {
    print("Count: $i");
    i++;
} while (i <= 3);
```

(7.) switch Statement: Used when we want to many fixed values for a

single variable.

Example:

```
String grade = "B";
switch (grade) {
  case "A":
     print("Excellent!");
     break;
  case "B":
     print("Good Job!");
     break;
  case "C":
     print("Needs improvement!");
     break;
  default:
     print("Invalid grade");
}
```

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

(1.) Class: A class is a like a blueprint or template.

It defines how an object should look and behave - like its properties (data) and functions (actions).

Think of it as a design for a house. It tells what room it should have, but the house doesn't exist yet.

(2.) Class:
(2.) Object: An object is a real thing created using a class.
It follows the blueprint and has its own actual values.
Like a house that was built using the house design (class). We can build many houses (objects) from one design.

(3.) Inheritance: Inheritance means one class can reuse things from another class.
It helps avoid writing the same code again and again.
Types:
(1.) Single Inheritance
(2.) Multilevel Inheritance
(3.) Hybrid Inheritance
(4.) Hierarchical Inheritance
Multiple Inheritance is not supported in Dart, but we can use interface for same behavior.

(4.) Polymorphism: Polymorphism means one action can behave differently based on the object.
It makes programs more flexible and powerful.

(5.) Abstraction: Abstraction means hiding the complex parts and showing only what's necessary. In coding, abstraction helps focus on what something does, not how it does it.

(6.) Encapsulation: Encapsulation means wrapping up the data and functions together and keeping it safe from outside interference.
It makes code more secure and easier to manage.

(7.) Interface: An interface is like a contract. It tells what functions a class must have, but not how they work.
It's used to make sure different classes follow the same set of rules, even if they behave differently.

4. Describe asynchronous programming in Dart, including Future, async, await, and streams.
Sometimes in apps, some tasks take time like downloading data from the internet, reading a file, etc. Instead of stopping everything and waiting, Dart lets the app keep running and finish those tasks in the background. This is called

asynchronous programming.

Future: A Future means something that will be done later.
Future is just a promise: "I'll give you the result after some time."

async: The async keyword means "this function is asynchronous."
We can only use await in async function.

await: await tells Dart: "Pause here until this Future is done, then move on."
It lets us wait for a result without blocking the whole app.

stream: A stream gives us many values over time, not just one like a Future.
Use stream when we expect multiple events.