Dart Programming Exercises

1. Variables, Functions, and Control Flow

Write a program that:

- Prompts the user to input a number.

- Defines a function `isPrime(int number)` to check if the number is prime.

- Calls the function and prints "Prime" if the number is prime, "Not Prime" otherwise.

- Additionally, calculates the factorial of the number using a loop and prints the result.

Goal: Practice control flow, function definition, and loops.

2. Collections, Loops, and Conditional Statements

Write a function `processNumbers` that:

- Accepts a list of integers.

- Filters out numbers less than 5.

- Squares each remaining number.

- Returns the sum of the squared numbers.

Use the function with the list `[2, 4, 6, 8, 10]` and print the result.

Goal: Learn to work with collections, loops, and filtering conditions.

3. Null Safety, Strings, and Higher-Order Functions

Write a function `formatNames(List<String?> names)` that:

- Removes null values from the list.

- Capitalizes the first letter of each name using String methods.

- Returns the formatted list as a single string, with names separated by commas.

Test it with the list `["john", null, "alice", "bob"]`.

Goal: Understand null safety, string manipulation, and higher-order functions.

4. OOP, Constructors, and Methods

Create a class `BankAccount` with:

- Properties: `String accountHolder`, `double balance`.
- A constructor to initialize the properties.
- Methods:
  - `deposit(double amount)` to add to the balance.
  - `withdraw(double amount)` to subtract from the balance (ensure sufficient funds).
  - `getDetails()` to print the account holder's name and balance.

Create an instance, perform some transactions, and print the account details.

Goal: Learn OOP concepts like constructors, methods, and object manipulation.

5. OOP, Mixins, Inheritance, and Abstraction

Create a small program that includes the following:

- An abstract class `Animal` with:
  - A method `makeSound()` (abstract).
  - A concrete method `describe()` that prints "I am an animal.".
- A class `Dog` that:
  - Inherits from `Animal`.
  - Implements the `makeSound()` method to print "Bark!".
- A mixin `RunFast` that:

  - Adds a method `run()` that prints "I run fast!".

- A class `Greyhound` that:

  - Extends `Dog`.

  - Mixes in `RunFast`.

In the `main()` function:

- Create an instance of `Greyhound`.

- Call `makeSound()`, `describe()`, and `run()` on the instance.

Goal: Combine OOP principles like inheritance, mixins, and abstraction.