Assignment: 07

Task 1: Class with Methods

Create a `Car` class with the following properties:

- `brand` (String)
- `model` (String)
- `year` (int)
- `mileage` (double)

Add a method `drive(double distance)` that increases mileage.

- Create two car objects, drive them for different distances, and print the updated mileage.

Task 2: Named Constructor

Modify the `Car` class to add a **named constructor** **`oldCar()`** that sets a default mileage of **100,000** for cars older than **10 years**.

- Create a car using this named constructor and print its details.

Task 3: Encapsulation (Getters & Setters)

Create a `Student` class with:

- Private `_marks` (int)
- A `getter` that returns marks
- A `setter` that ensures marks **cannot be negative or greater than 100**

Create a student object, try setting invalid marks, and print the final marks.

Task 4: List of Objects Processing

Create a `Product` class with:

- `name` (String)
- `category` (String)
- `price` (double)

Create a **list of products** and filter only **electronics** with a price above 5000.

Task 5: Word Frequency Counter

Write a function `wordFrequency(String sentence)` that:

- Counts how many times each word appears in a sentence ● Ignores **case sensitivity** (e.g., "Dart" and "dart" are the same)

Test it with a sample sentence.

Task 6: Using Continue & Break

Write a loop that:

- **Skips numbers divisible by 3** using `continue`
- **Stops at 17** using `break`

Print the remaining numbers.

Task 7: List of Maps (API-like Data Processing)

You have a list of **users (maps)**:

```
List<Map<String, dynamic>> users = [
  {"id": 1, "name": "Ali", "age": 25, "role": "admin"},
  {"id": 2, "name": "Sara", "age": 30, "role": "user"},
  {"id": 3, "name": "Ahmed", "age": 20, "role": "admin"},
  {"id": 4, "name": "Zara", "age": 28, "role": "user"}
];
```

- Write a function that **returns only admin users**.

Task 8: Function as a Parameter

Write a function `processNumber(int number, Function operation)` that:

- Applies the `operation` function to `number`
- Pass `doubleIt()` and `squareIt()` functions as parameters to `processNumber()`

Final Challenge: Object-Oriented System

Create a **Bank Account System** with:

1. **`BankAccount` class:**
   - Fields: `accountNumber`, `_balance` (private)
   - Methods: `deposit()`, `withdraw()`, `getBalance()`
   - Prevent withdrawal if balance **goes below 1000**
2. **Test Cases:**
   - Deposit money
   - Withdraw valid and invalid amounts