1. **Create a Class and Object**:

- Create a class named `Person` with attributes such as name, age, and address.

- Create an object of the `Person` class and set its attributes.

- Print the details of the person using the object.

2. **Class Inheritance**:

- Create a base class `Animal` with attributes like `name` and `sound`.

- Create derived classes (e.g., `Dog`, `Cat`, `Cow`) that inherit from the `Animal` class.

- Each derived class should have a method that makes the animal produce its sound.

- Create objects of these classes and call their respective methods to produce sounds.

3. **Encapsulation**:

- Create a class `Student` with private attributes (name, age, and roll number).

- Implement getter and setter methods to access and modify these attributes.

- Create an object of the `Student` class and demonstrate the use of getter and setter methods.

4. **Polymorphism**:

- Create a base class `Shape` with a method `calculateArea()`.

- Create derived classes (e.g., `Circle`, `Rectangle`, `Triangle`) that override the `calculateArea()` method to calculate the area of their respective shapes.

- Create objects of these classes and call the `calculateArea()` method to compute the areas.

5. **Abstraction**:

- Create an abstract class `Shape` with an abstract method `area()`.

- Create concrete classes (e.g., `Circle`, `Rectangle`, `Triangle`) that inherit from the `Shape` class and implement the `area()` method.

- Demonstrate the use of abstraction by creating objects of the concrete classes and calculating their areas.

6. **Composition**:

- Create a class `Author` with attributes like name and books (a list of book titles).

- Create a class `Book` with attributes like title, publication year, and an author (an object of the `Author` class).

- Create an object of the `Author` class and add multiple books to it.

- Print the details of the author and their books.

7. **Association**:

- Create a class `Library` and a class `Book`.

- Use association to model the relationship between a library and its collection of books.

- Demonstrate how a library can have multiple books and print the list of books in the library.

8. **Inheritance and Interfaces**:

- Create a base class `Vehicle` with attributes like make, model, and year.

- Create an interface `Driveable` with a method `startEngine()`.

- Create derived classes (e.g., `Car`, `Motorcycle`) that inherit from `Vehicle` and implement the `Driveable` interface.

- Create objects of these classes and call the `startEngine()` method.

9. **Class Constructors**:

- Create a class `Product` with attributes like name, price, and category.

- Implement multiple constructors, including named constructors, for the `Product` class.

- Create objects using different constructors and print their details.

10. **Static Members**:

- Create a class `MathUtils` with static methods for common mathematical operations (e.g., `add`, `subtract`, `multiply`, `divide`).

- Demonstrate how to call these static methods without creating an instance of the class.

These tasks cover a range of OOP concepts, including classes, inheritance, encapsulation, polymorphism, abstraction, composition, association, interfaces, constructors, and static members. Practice these tasks to strengthen your understanding of OOP in Dart.