Programming Exercises: Comprehensive Challenges

1. Variables, Collections, and Functions

Write a program that:

- Accepts a list of integers as input.
- Defines a function `calculateStats(List<int> numbers)` that:
  - Calculates the sum of all numbers.
  - Finds the maximum and minimum values in the list.
  - Returns a formatted string with these statistics.
- Call the function with `[5, 12, 7, 3, 18]` and print the result.

2. Loops, Conditional Statements, and Strings

Write a program that:

- Accepts a string from the user.
- Counts the occurrences of each vowel (a, e, i, o, u) in the string using a `Map<String, int>`.
- Ignores case (e.g., count A and a together).
- Prints the vowels and their counts.

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

Write a function `findLargest(List<int>? numbers)` that:

- Checks if the input list is null or empty. If so, returns null.
- Otherwise, finds and returns the largest number in the list using the `reduce` method.
- Call the function with `[10, 3, 45, 7]` and `null`, and print the results.

4. OOP, Polymorphism, and Interfaces

Create a program with the following:

- An interface `Shape` with:
  - A method `double area()`.
- Two classes `Circle` and `Rectangle` that:
  - Implement the `Shape` interface.
  - Provide implementations for the `area()` method.
  - Use appropriate constructors to initialize properties (radius for Circle, length and width for Rectangle).
- In `main()`, create instances of `Circle` and `Rectangle` and print their areas.

5. Mixins, Inheritance, and Method Overriding

Create a program that:

- Defines a mixin `Fly` with:
  - A method `fly()` that prints "I can fly!".
- Creates a base class `Bird` with:
  - A method `chirp()` that prints "Chirp chirp!".
- Creates a class `Parrot` that:
  - Extends `Bird`.
  - Mixes in `Fly`.
  - Overrides the `chirp()` method to print "Parrot is chirping!".
- In `main()`, create a `Parrot` instance and call `fly()` and `chirp()`.