Exercise 1: The Basics (The Contract)

Goal: Create a simple interface and implement it in two different classes.

- Define an interface named IVehicle.
- It should have a method void Move().
- Create a class Car and a class Bicycle that both implement IVehicle.
- In Car.Move(), print "Driving on the road."
- In Bicycle.Move(), print "Pedaling on the path."

Exercise 2: Multiple Interface Implementation

Goal: Learn that a class can follow multiple "contracts" at once.

- Create two interfaces: IReadable (with a method void Read()) and IWriteable (with a method void Write(string text)).
- Create a class NotePad that implements **both**.
- Create a class ReadOnlyDocument that implements **only** IReadable.
- Test them by trying to call Write on both objects and observe why one fails.

Exercise 3: Explicit Interface Implementation

Goal: Handle naming conflicts when two interfaces have the same method name.

- Create an interface IPlayer with a method void Play().
- Create an interface IFile with a method void Play().
- Create a class MediaFile that implements both.
- Use **explicit implementation** so that IPlayer.Play() prints "Playing music" and IFile.Play() prints "Executing file".

Exercise 4: Interface Inheritance

Goal: Understand how interfaces can extend one another.

- Create an interface IWorkable with a method void Work().
- Create a second interface ISmartWorker that **inherits** from IWorkable.
- Add a method void Learn() to ISmartWorker.
- Create a class HumanWorker that implements ISmartWorker.
- **The Challenge:** Notice how many methods HumanWorker is now forced to implement? (It should be two).

Exercise 5: Interface Properties and Logic

Goal: Implement properties within an interface.

- Create an interface IShape with a property double Area { get; }.
- Create a class Circle (needs a Radius field).
  - Calculation: $$Area = \pi \times Radius^2$$
- Create a class Rectangle (needs Width and Height fields).
  - Calculation: $$Area = Width \times Height$$
- Create an array of IShape objects and write a function that calculates the total area of all shapes in the array.