Exercise 1: The One-Liner

Goal: Create a simple data container using the most concise syntax possible.

- **Task:** Define a public record named Product that takes a string Name, a decimal Price, and a string Category.
- **Validation:** In your Main method, instantiate a product and print it to the Console. Notice how the output is automatically formatted without you writing a ToString() method.

Exercise 2: Value-Based Equality

Goal: Understand why records are different from classes when comparing objects.

- **Task:** 1. Create a class CoffeeClass with Name and Size properties. 2. Create a record CoffeeRecord with the same properties. 3. In Main, create two identical instances of the class and two identical instances of the record.
- **Validation:** Use Console.WriteLine to compare the two class instances and then the two record instances using ==. Explain why one returns False and the other returns True.

Exercise 3: Non-destructive Mutation

Goal: Practice using the with keyword to "change" immutable data.

- **Task:** 1. Define a record User(string Username, string Email, bool IsAdmin). 2. Create an instance where IsAdmin is false. 3. Create a second instance named adminUser based on the first one, but use the with expression to change IsAdmin to true.
- **Validation:** Print both users to verify the original remained unchanged (immutable) and the new one has the updated value.

Exercise 4: Record Inheritance & Primary Constructors

Goal: Pass data through a hierarchy using primary constructors.

- **Task:** 1. Create a base record Vehicle(string Make, string Model). 2. Create a derived record ElectricCar that inherits from Vehicle and adds an int BatteryCapacity. 3. Ensure the ElectricCar constructor correctly passes the Make and Model to the base Vehicle.
- **Validation:** Instantiate an ElectricCar and use **Deconstruction** to extract the Model and BatteryCapacity into two local variables.

Exercise 5: The "Record Struct" Performance Challenge

Goal: Differentiate between heap-allocated records and stack-allocated record structs.

- **Task:** 1. Define a public readonly record struct GPSLocation(double Latitude, double Longitude). 2. Why would we use readonly record struct for a GPS coordinate instead of a standard record class? (Hint: Think about memory and high-frequency updates).
- **Validation:** Try to change the Latitude of an existing GPSLocation instance directly. Observe the compiler error and fix it using the with keyword instead.