Lsp_Act33

🔍 LSP Activity 33: Identify & Refactor

🧠 Scenario:

You are part of a team working on a shape calculator. A teammate created a `Rectangle` class and decided to make `Square` inherit from it. The app was working fine with rectangles — but when they tried using squares, the behavior broke!

Your lead developer says:
🗣️ "This looks like a violation of the **Liskov Substitution Principle**. Can you find the problem?"

📝 Instructions:

Part 1: Spot the LSP Violation

1. Read the given code carefully
2. Test what happens when a `Rectangle` is used vs. a `Square`.

Code block

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def set_width(self, width):
        self.width = width

    def set_height(self, height):
        self.height = height

    def get_area(self):
        return self.width * self.height

class Square(Rectangle):
    def set_width(self, width):
        self.width = width
        self.height = width

    def set_height(self, height):
```

```
21          self.height = height
22          self.width = height
```

Part 2: **Explain the Violation**

Answer the following:

- Why is this a violation of the Liskov Substitution Principle?
- What is the expected vs actual behavior?
- Why does the `Square` cause the problem?

Part 3: **Refactor Using LSP**

Fix and Code the design so that: ✅ Each shape can be used **safely** where a `Shape` is expected
✅ No behavior breaks
✅ Each shape has its **own proper area calculation**