Module 1: The AI-Native Smells Detector, Lesson 1: Rigidity | Assignment: The Pricing Engine

Overview

A small pricing function has been growing for six months. Every new customer tier required edits in three places. Read the code, diagnose the rigidity, and sketch a refactor.

Estimated Time

10 minutes

Before You Start

- Read the code block below carefully.
- Have a blank text file or notepad ready for your answers.
- Do NOT run or refactor the code yet - the goal is the diagnosis and the sketch.

REMEMBER

Rigidity is measured by the cost of a typical change. The question to ask is not 'is this code wrong?' but 'what happens when I add the next tier?'

Your Task

Read the following pricing engine. Answer the four questions in the Submit section.

```
# pricing.py - the system as it stands today

TIERS = ["free", "basic", "pro", "enterprise"]

def monthly_price(tier: str, seats: int) -> float:
    if tier == "free":
        base = 0
    elif tier == "basic":
        base = 19
    elif tier == "pro":
        base = 49
    elif tier == "enterprise":
        base = 199
    else:
        raise ValueError(f"unknown tier: {tier}")
```

```
    if tier == "free":
        per_seat = 0
    elif tier == "basic":
        per_seat = 5
    elif tier == "pro":
        per_seat = 12
    elif tier == "enterprise":
        per_seat = 25
    else:
        per_seat = 0

    if tier == "free":
        cap = 3
    elif tier == "basic":
        cap = 10
    elif tier == "pro":
        cap = 50
    elif tier == "enterprise":
        cap = 1000
    else:
        cap = 0

    seats = min(seats, cap)
    return base + per_seat * seats
```

What to Submit

1. Name the smell in one sentence. (You should be able to do this in less than 20 seconds.)
2. Count: if you add a new tier called 'team' with its own base price, per-seat price, and seat cap, how many separate places in this function do you have to edit?
3. Sketch (in three to five lines, in pseudocode or a code skeleton) a refactor that turns 'add a new tier' into 'add one row to a data structure.'
4. Bonus: which Bob Martin smell from the course would your refactor ALSO improve, as a side effect?

Solution Sketch (Instructor)

SOLUTION

Smell: rigidity - one logical change (adding a tier) requires three physical changes (three separate if/elif chains). Edit count: three. Refactor: replace the three chains with a single TIERS data structure (a dict keyed by tier name, with 'base', 'per_seat', and 'cap' fields). The function becomes one lookup plus one arithmetic expression. Side effect: removing the inline if-chains also reduces viscosity (the wrong way - adding another branch - is no longer easier than the right way - adding a row to the dict).

Source File Reference

This assignment is part of Lesson 1 of Module 1: The AI-Native Smells Detector. The code snippet above is a teaching example. The full hands-on for this lesson uses related source files in the source_code/ directory.