Building a Safety Net

Week 3: Discussion Session

Advanced Software Engineering

Learning Objectives

- Understand the importance of characterization tests
- Learn techniques for safely adding tests to legacy code
- Master the art of identifying and creating seams

Understanding Characterization Tests

What are they?

- Tests that describe current behavior
- Document “what is” rather than “what should be”
- First line of defense when working with legacy code

Why do we need them?

- Capture existing behavior
- Provide safety for refactoring
- Document system behavior
- Enable incremental improvement

Discussion Questions

1. How do characterization tests differ from traditional unit tests? When would you prefer one over the other?
2. What strategies can be employed when the legacy code seems impossible to test? How do you balance pragmatism with best practices?
3. How do you determine the appropriate level of characterization testing before beginning refactoring? What factors influence this decision?

Code Examples

C++ Example: Adding Characterization Tests

```cpp
// Original Legacy Code
class InvoiceCalculator {
public:
    double calculateTotal(const std::vector<LineItem>& items) {
        double total = 0;
        for (const auto& item : items) {
            total += item.quantity * item.price;
            if (item.quantity > 10) {
                total *= 0.95;  // Bulk discount
            }
            if (total > 1000) {
                total *= 0.90;  // Large order discount
            }
        }
        return total;
    }
};

// Characterization Test
TEST_CASE("InvoiceCalculator preserves existing behavior") {
    InvoiceCalculator calc;
    std::vector<LineItem> items = {
```

Key Techniques

1. Capturing Current Behavior
   - Use automated tools where possible
   - Document exact current outputs
   - Include edge cases
2. Creating Seams
   - Extract Method
   - Extract Interface
   - Parameterize Constructor
   - Break Dependencies
3. Testing Strategies
   - Start with broad integration tests
   - Gradually add more specific tests
   - Use approval testing for complex output

Required Reading

- Chapters 8-10 of “Working Effectively with Legacy Code”
- Focus on:
  - Breaking Dependencies
  - Sensing and Separation
  - Test Organization

Next Week

Prepare for hands-on kata:

- Writing effective characterization tests
- Creating seams in legacy code
- Organizing test suites effectively