Core Dependency Breaking Techniques

Week 7: Discussion Session

Advanced Software Engineering

Learning Objectives

- Understand dependency breaking techniques
- Learn to use adapters, seams, and interfaces
- Enable safe refactoring of tightly coupled code

Dependency Breaking Techniques

Adapters

- Introduce new interfaces to decouple code
- Allow substitution of dependencies for testing

Seams

- Places where you can alter behavior without editing code
- Types: Preprocessor, Link, Object, and Parameter seams

Discussion Questions

1. What are the most common sources of hard-to-break dependencies in C++ and Python codebases?
2. How do adapters and seams differ in their application? When would you use each?
3. What are the risks of introducing new interfaces or seams? How can you mitigate them?
4. Share an example where breaking a dependency enabled a major refactoring or improvement.

Code Examples

C++ Example: Adapter Pattern

```cpp
class LegacyLogger {
public:
    void writeLog(const std::string& msg) {
        // ... legacy logging ...
    }
};

class ILogger {
public:
    virtual void log(const std::string& msg) = 0;
};

class LoggerAdapter : public ILogger {
    LegacyLogger legacy;
public:
    void log(const std::string& msg) override {
        legacy.writeLog(msg);
    }
};
```

C++ Example: Object Seam

Key Takeaways

- Adapters and seams enable testability and flexibility
- Breaking dependencies is essential for safe refactoring
- Use interfaces and parameterization to inject dependencies

Required Reading

- Chapters 11, 14-15 of “Working Effectively with Legacy Code”
- Focus on:
  - Dependency breaking patterns
  - Seams and their types
  - Refactoring strategies

Next Week

Prepare for hands-on kata:

- Implementing adapters and seams
- Refactoring for testability
- Managing risk during dependency breaking