i2i Systems

REFACTORING: IMPROVING THE DESIGN OF EXISTING CODE

CHAPTER 3, 4, 5

Ahmet Sefa Çetin

24.07.2024

Overview

- Chapter 3 - Bad Smells in Code
- Chapter 4 - Building Tests
- Chapter 5 - Toward a Catalog of Refactorings

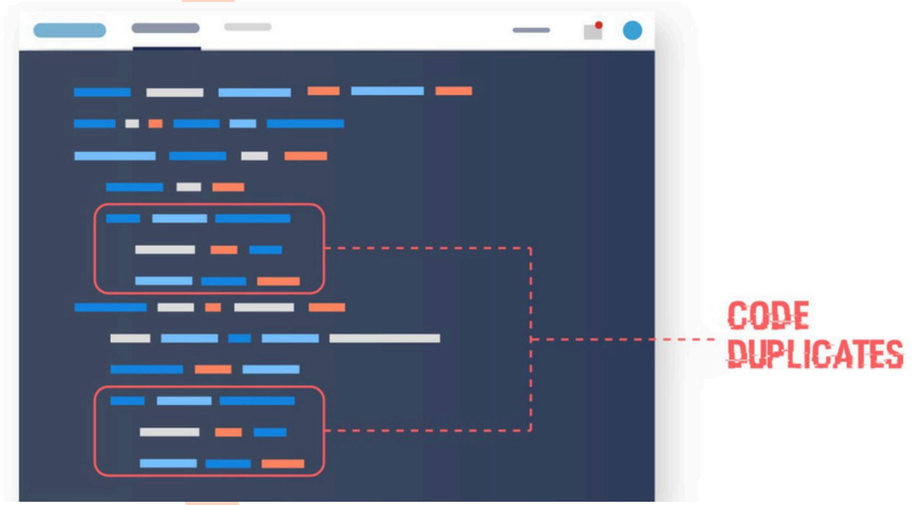
CHAPTER 3
BAD SMELLS IN CODE

Identifying and addressing common 'bad smells' in code to improve software design and maintainability.

“If it stinks, change it.”

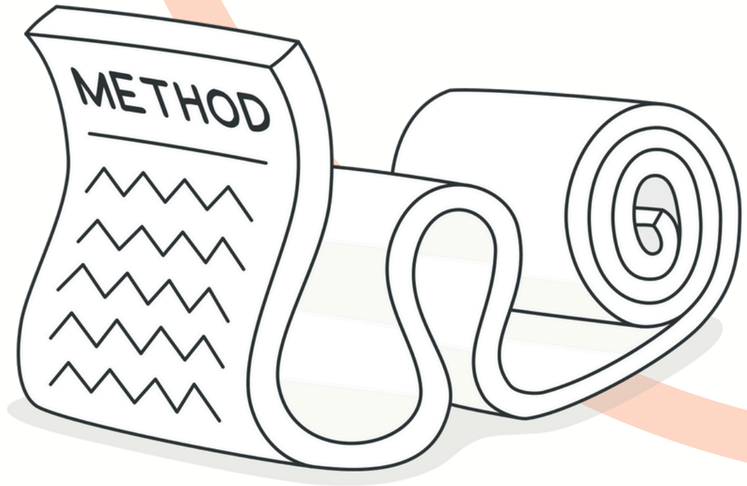
Duplicated Code

The presence of identical or very similar code in multiple places within a codebase.



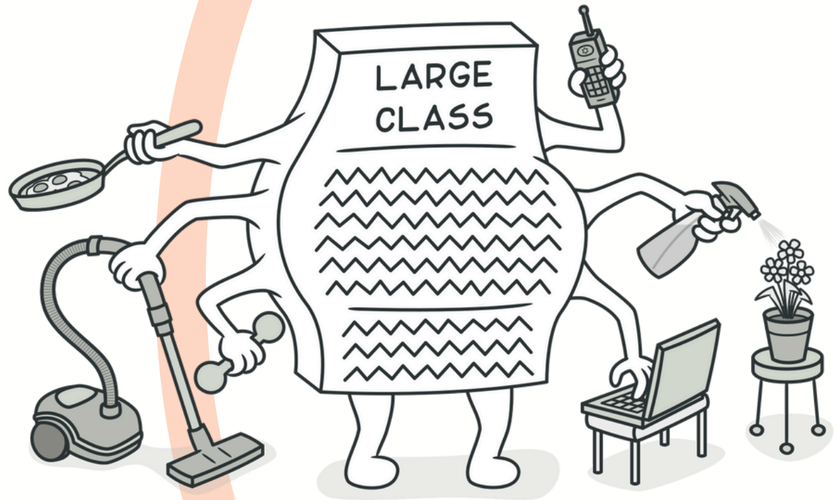
Long Method

A method that has grown too large and tries to accomplish too much, often containing multiple responsibilities.



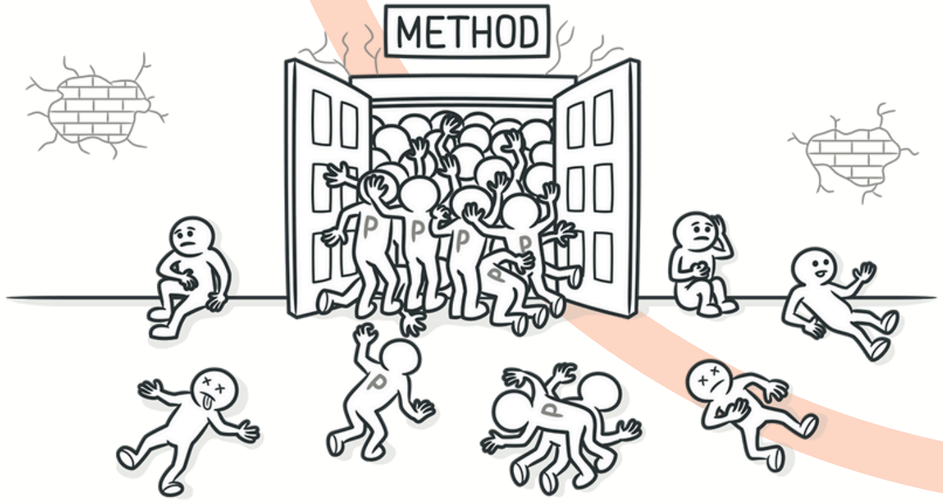
Large Class

A class that has grown too large and tries to do too many things, violating the Single Responsibility Principle.



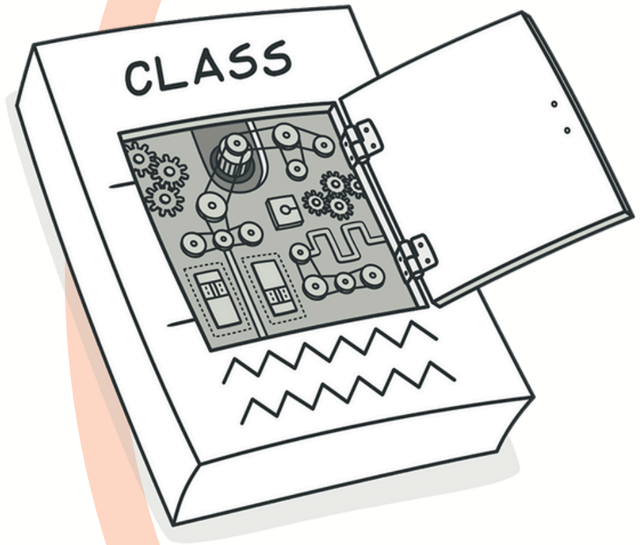
Long Parameter List

A method that takes too many parameters, making it difficult to understand and use.



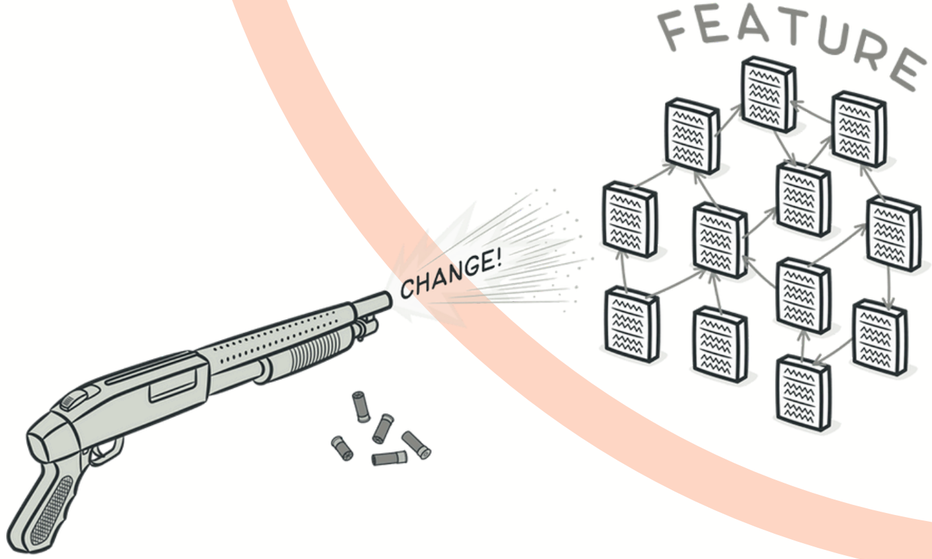
Divergent Change

A class that suffers from frequent changes for different reasons, indicating that it has multiple responsibilities.



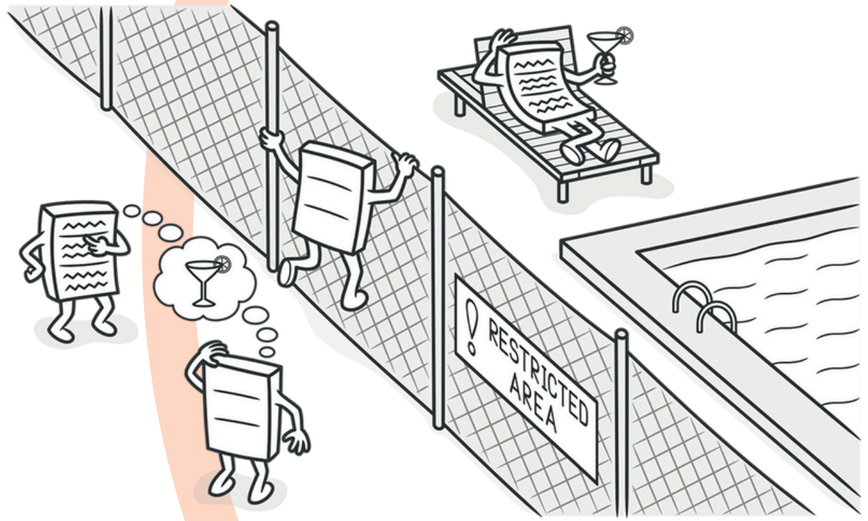
Shotgun Surgery

A change that requires altering many small pieces of code in different places, indicating poor modularity.



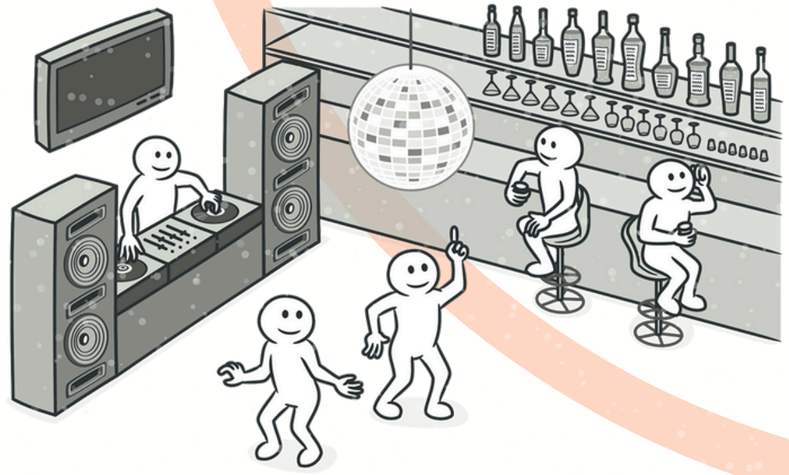
Feature Envy

A method that is more interested in the data of another class than that of the class it is in.



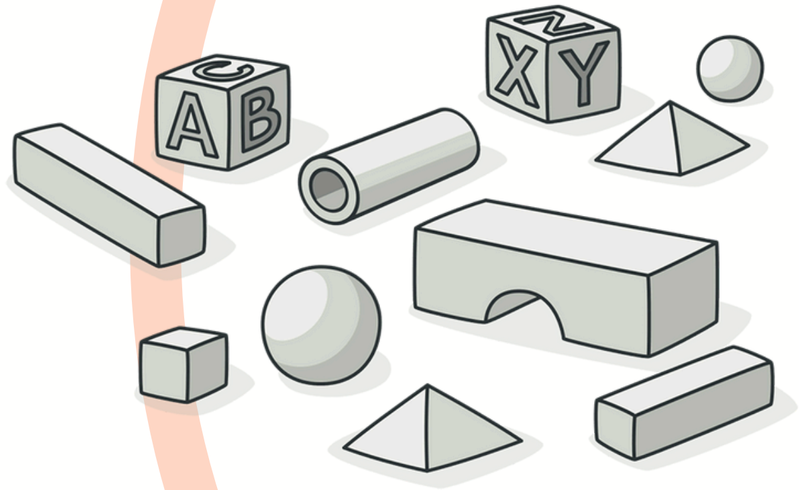
Data Clumps

Groups of data items that tend to be passed around together and should be combined into their own object.



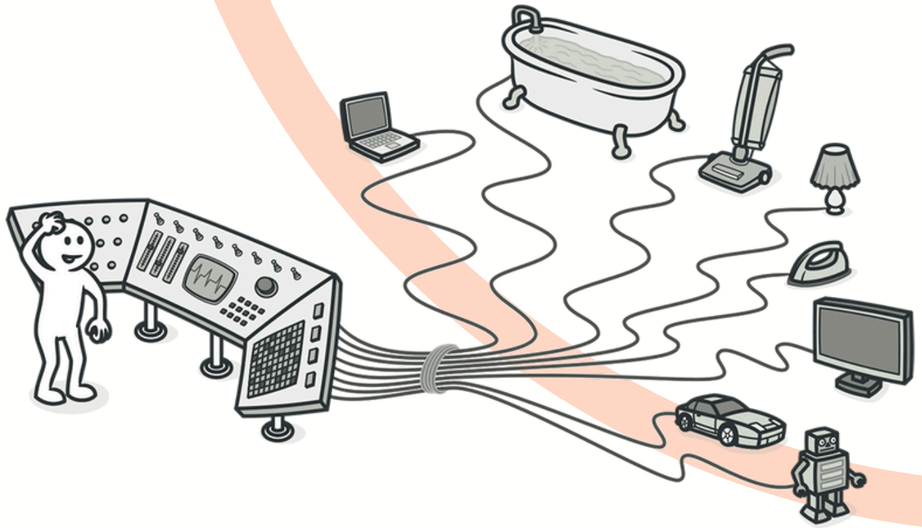
Primitive Obsession

The excessive use of primitive data types instead of small objects for simple tasks.



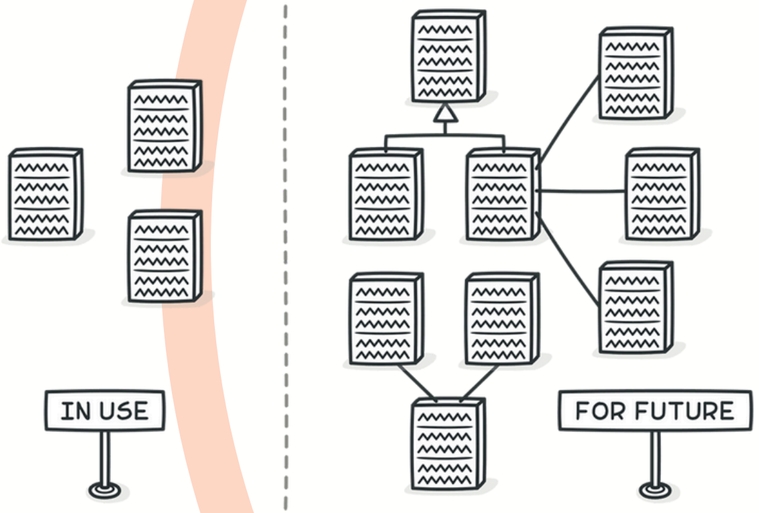
Switch Statements

Frequent switch statements that can be replaced with polymorphism to improve code clarity and maintainability.



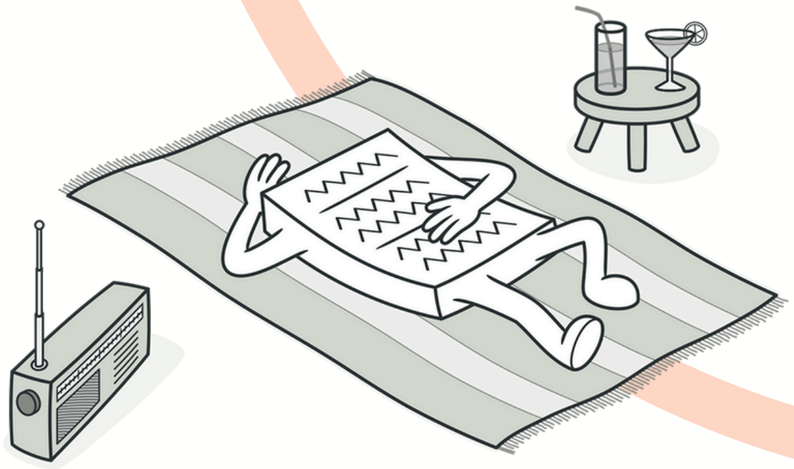
Speculative Generality

Code that is more general or flexible than it needs to be, often because of anticipating future requirements.



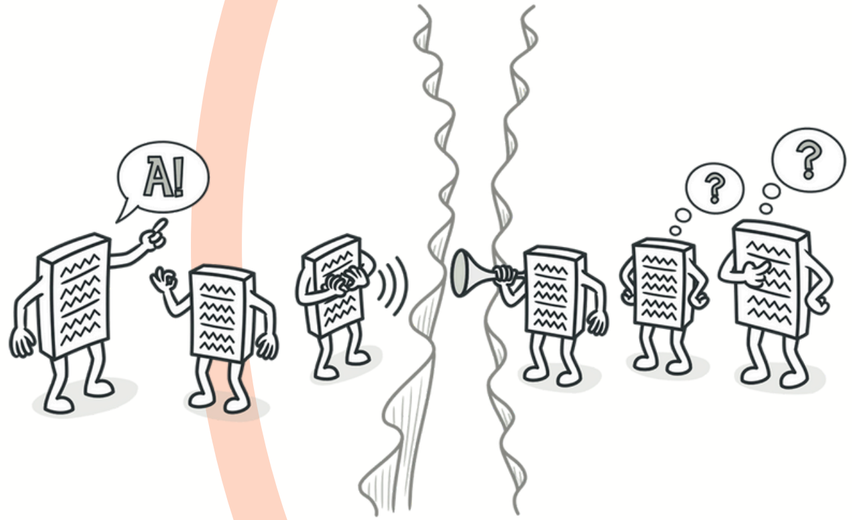
Lazy Class

A class that does too little and does not justify its existence in the codebase.



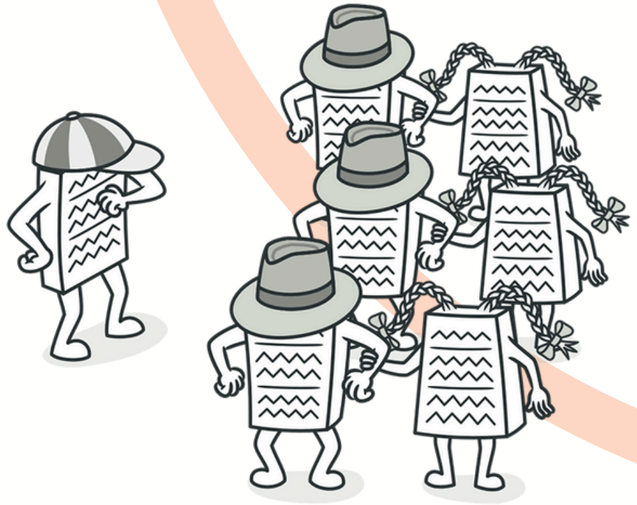
Message Chains

A sequence of calls that navigate through multiple objects to get a desired value, leading to tight coupling.



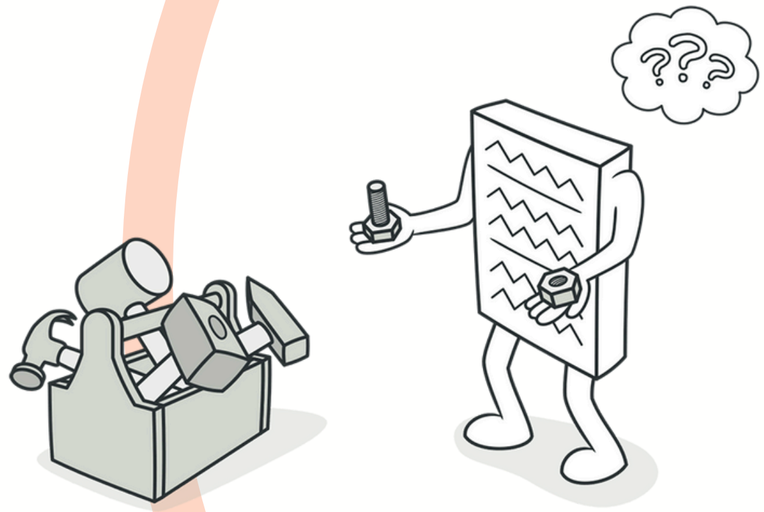
Middle Man

A class that delegates almost all of its work to another class, adding unnecessary complexity.



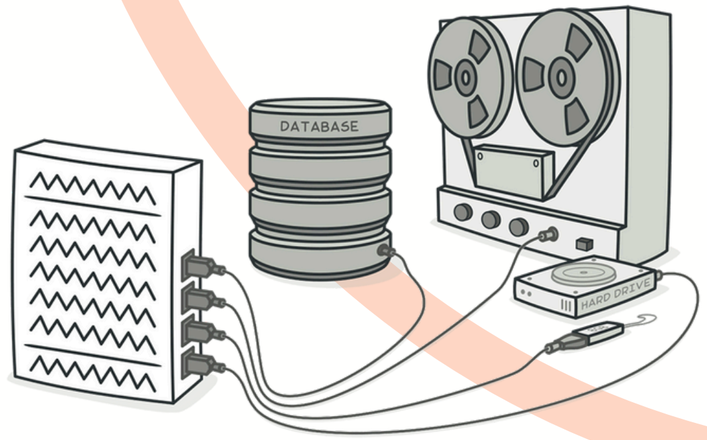
Incomplete Library Class

A library class that lacks the needed functionality, requiring extensions or workarounds.



Data Class

A class that primarily holds data with little or no behavior, leading to poor encapsulation.



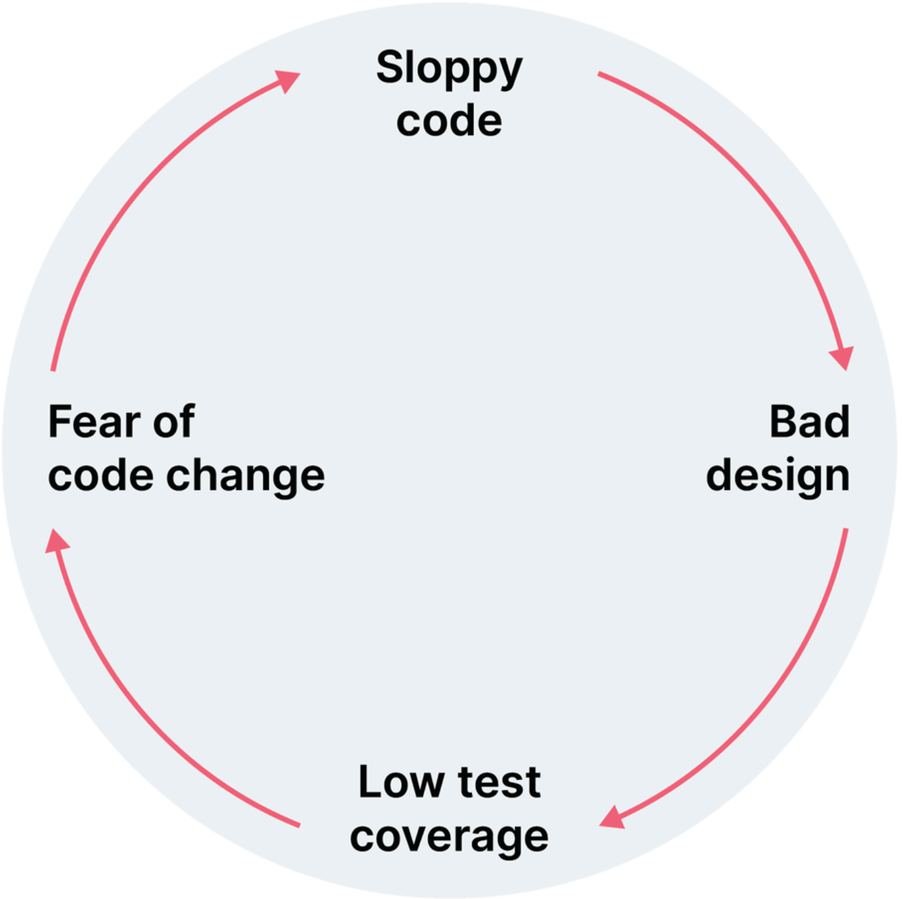
CHAPTER 4
BUILDING TESTS

Building a solid suite of tests to ensure the stability and reliability of code during refactoring.

Importance of Pre-existing Tests

Refactoring requires pre-existing tests to ensure code stability during changes.

Automated tests catch bugs early, reducing time spent on debugging.



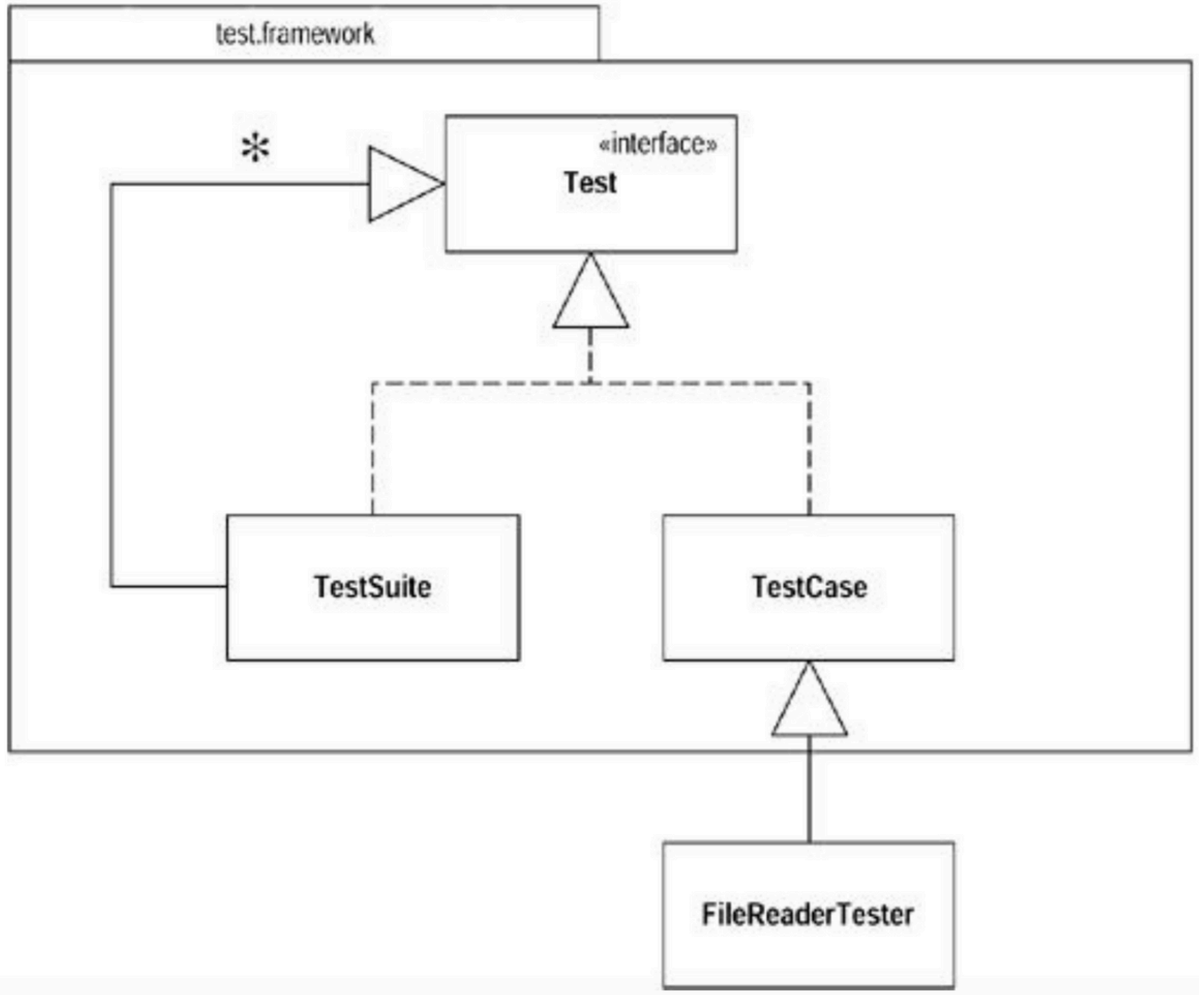
JUnit Test Framework

JUnit is a popular testing framework for Java applications.

It simplifies writing and running tests, ensuring code reliability.

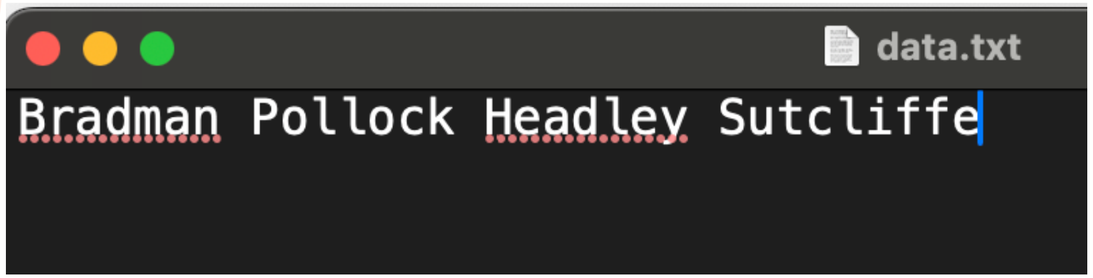
JUnit tests are written in classes that extend TestCase.

```
class FileReaderTester extends TestCase {
    private FileReader _input;

    public FileReaderTester(String name) {
        super(name);
    }

    protected void setUp() throws IOException {
        _input = new FileReader("data.txt");
    }

    protected void tearDown() throws IOException {
        _input.close();
    }
}
```



Test Methods

Test methods contain the actual test logic and assertions.




```
public void testRead() throws IOException {
    char ch = '&';
    for (int i = 0; i < 4; i++)
        ch = (char) _input.read();
    assert('d' == ch);
}
```

```
public void testReadAtEnd() throws IOException {
    int ch = -1234;
    for (int i = 0; i < 33; i++)
        ch = _input.read();
    assertEquals(-1, ch);
}
```

Test Suites

A test suite groups multiple test classes together.

It allows running all related tests at once for comprehensive coverage.

```
public class MasterTester {
    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }

    public static Test suite() {
        TestSuite suite = new TestSuite();
        suite.addTest(new FileReaderTester("testRead"));
        suite.addTest(new FileReaderTester("testReadAtEnd"));
        return suite;
    }
}
```

```
.
Time: 0.110
OK (1 tests)
```

```
.F
Time: 0.220
!!!FAILURES!!!

Test Results:
Run: 1 Failures: 1 Errors: 0
There was 1 failure:
1) FileReaderTester.testRead test.framework.AssertionFailedError
```

CHAPTER 5
TOWARD A CATALOG OF REFACTORINGS

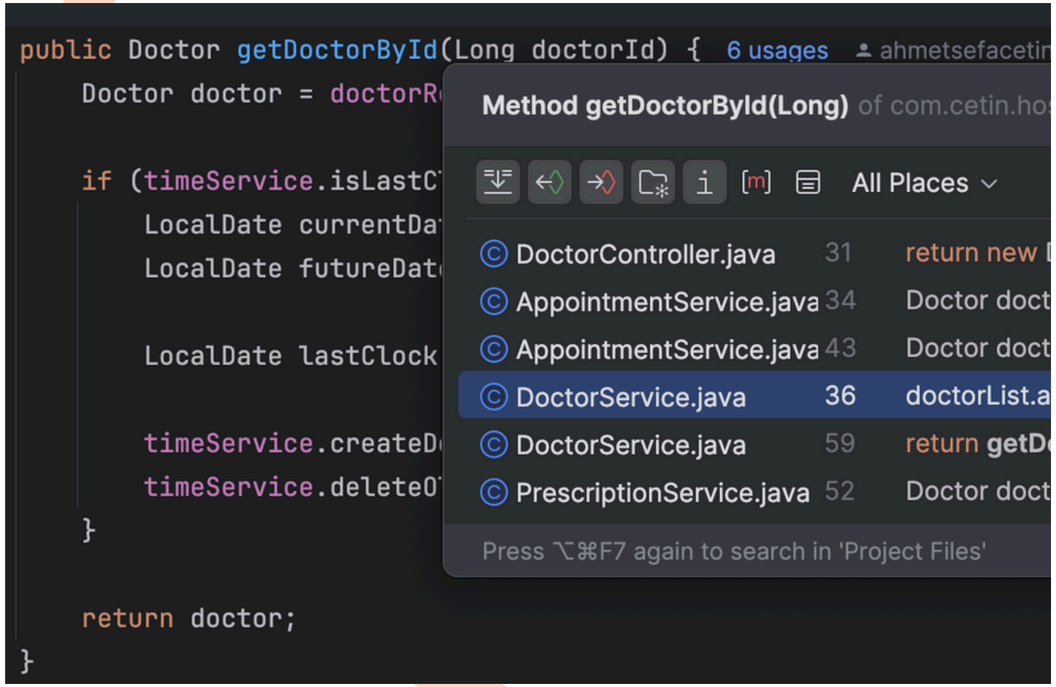
This chapter provides a solid starting point for your own refactoring work.

Format of the Refactorings

- The **name** provides a clear and concise identifier for the refactoring.
- The **summary** briefly describes the refactoring's purpose and effect.
- The **motivation** explains why the refactoring is necessary and beneficial.
- The **mechanics** offer step-by-step instructions to perform the refactoring.
- The **examples** illustrate the code transformation before and after the refactoring.

Finding References

- It is crucial to locate all references to a method, class, or variable before making changes.
- Utilize automated tools and IDE features to efficiently find references.
- Manual searching is error-prone and slow, making reliable tools essential.
- Ensures that the refactoring process is thorough and consistent, maintaining code integrity and functionality.



How Mature Are These Refactorings

- These refactorings have evolved and been refined over the years within the software community.
- Many software experts and academics have tested and improved these refactorings.
- Manual searching is error-prone and slow, making reliable tools essential.
- Ensures that the refactoring process is thorough and consistent, maintaining code integrity and functionality.

References

1. Refactoring: Improving the Design of Existing Code - Second Edition
2. https://refactoring.guru
3. https://chatgpt.com/

Thanks for watching!

Do you have any questions ?

 linkedin.com/in/ahmetsefacetin

 ahmetsefacetin@outlook.com