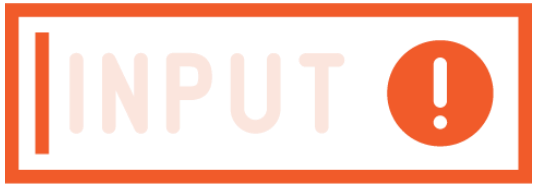
Final Words

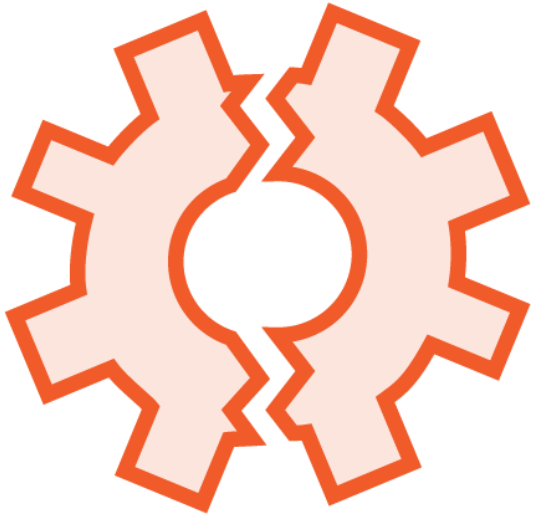
Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/

What Are We Defending Our Code From?



Incorrect entry



Invalid operations

System mishaps

Future developers

Strengthen Your Code's Defenses

Improve code comprehension

Improve code quality

Improve code predictability

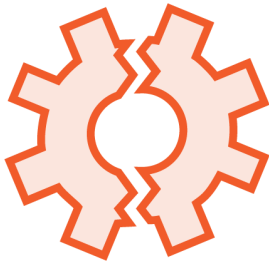
Improve Code Comprehension

Write code that is clean and easy to read

Single responsibility principle

Separation of concerns

Don’t repeat yourself (DRY)



Improve Code Quality

Build unit tests

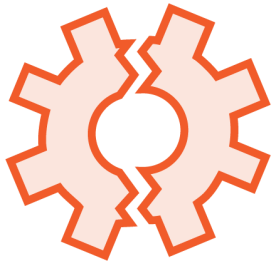
Re-execute tests after each modification

Improve Code Predictability

Principle of Least Surprise

Validate method arguments

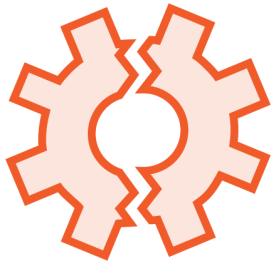
- Define a clear method signature
- Fail fast with guard clauses
- Refactor for separation of responsibilities



Improve Code Predictability

Handle nulls

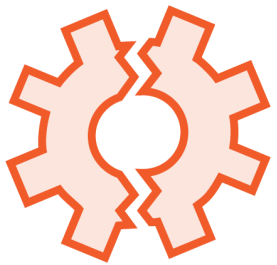
- Use nullable value types as needed
- Guard against null nullable value types
- Guard against null reference types
- Enable the reference type nullability features (C# 8+)
- Use nullable and non-nullable reference types



Improve Code Predictability

Return predictable results

- Return a value when expected
- Return a nullable type as needed
- Consider returning a tuple or object instead of throwing exceptions



Improve Code Predictability

Manage exceptions

- Define an exception management strategy
- Throw appropriate .NET exceptions
- Create and throw custom exceptions as needed
- Catch what you're thrown

Learning More

Pluralsight courses

- Working with Nulls in C#
- Error Handling in C# with Exceptions
- C# Unit Testing with xUnit (path)

Learning More

Pluralsight courses

- Object-Oriented Programming Fundamentals in C#
- C# Best Practices: Improving on the Basics
- C# Best Practices: Collections and Generics

Defensive Coding in C#

Image from the James Ford Bell Library, University of Minnesota