Fundamentals:
The Single Responsibility Principle

Steve Smith
http://pluralsight.com/

Outline

- SRP Defined
- The Problem
- An Example
- Refactoring to Apply SRP
- Related Fundamentals

SRP: The Single Responsibility Principle

The Single Responsibility Principle states that every object should have a single responsibility, and that responsibility should be entirely encapsulated by the class.

Wikipedia

There should never be more than one reason for a class to change.

Robert C. "Uncle Bob" Martin

SINGLE RESPONSIBILITY PRINCIPLE

Just Because You Can, Doesn't Mean You Should

Cohesion and Coupling

- Cohesion: how strongly-related and focused are the various responsibilities of a module
- Coupling: the degree to which each program module relies on each one of the other modules

Strive for low coupling and high cohesion!

Responsibilities are Axes of Change

- Requirements changes typically map to responsibilities
- More responsibilities == More likelihood of change
- Having multiple responsibilities within a class couples together these responsibilities
- *The more classes a change affects, the more likely the change will introduce errors.*

Demo

The Problem With Too Many Responsibilities

The Problem

- **Cash Transactions Don't Need Credit Card Processing**
- **Point of Sale Transactions Don't Need Inventory Reservations**
  - Store inventory is updated separately in our system
- **Point of Sale Transactions Don't Need Email Notifications**
  - The customer doesn't provide an email
  - The customer knows immediately that the order was a success
- **Any change to notifications, credit card processing, or inventory management will affect Order as well as the Web and Point of Sale implementations of Order!**

Demo

Refactoring to a Better Design

What is a Responsibility?

- "a reason to change"
- A difference in usage scenarios from the client's perspective
- Multiple small interfaces (follow ISP) can help to achieve SRP

Summary

- **Following SRP leads to lower coupling and higher cohesion**
- **Many small classes with distinct responsibilities result in a more flexible design**
- **Related Fundamentals:**
  - Open/Closed Principle
  - Interface Segregation Principle
  - Separation of Concerns
- **Recommended Reading:**
  - Clean Code by Robert C. Martin [http://amzn.to/Clean-Code]

Credits

- **Images Used Under License**
  - http://www.lostechies.com/blogs/derickbailey/archive/2009/02/11/solid-development-principles-in-motivational-pictures.aspx

- **SRP Article**
  - http://www.objectmentor.com/resources/articles/srp.pdf

For more in-depth online developer training visit

pluralsight

see what you can learn

on-demand content from authors you trust