

Objectives

- Design in the Small: Code Smells
- Refactoring

Oct 24, 2008

Sprenkle - CS209

1

Reflection on Project 1

- What were the difficult parts of Project 1?
- Did they get any easier?
- Did you develop a system or any techniques to make the process easier?
- In the future, how could you make the process easier?

Don't forget what you've learned.
Integrate testing into your development.

Oct 24, 2008

Sprenkle - CS209

2

Review

- What is guaranteed in software development?
- What are some principles of design in Object-oriented Programming to address the challenge posed by that guarantee?
- What is the underlying theme of how to achieve those principles?

Oct 24, 2008

Sprenkle - CS209

3

Review: Best Practices

- (DRY): Don't repeat yourself
- Shy
 - Avoid Coupling
- Tell, Don't Ask
- Open-closed principle
- Avoid code smells

Oct 24, 2008

Sprenkle - CS209

4

Code Smells

A hint in the code that something could be designed better

- | | |
|-----------------------------|--|
| • Duplicated code | • Switch statements/long if statements |
| • Long method | • Shotgun surgery |
| • Large class | • Literals |
| • Long parameter list | • Global variables |
| • Very similar subclasses | • Side effects |
| • Too many public variables | • Using instanceof |
| • Empty catch clauses | |

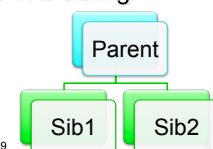
Oct 24, 2008

Sprenkle - CS209

5

Duplicated Code

- Example: same expression in 2 methods of the same class
 - Extract method
 - Call method from those two places
 - `MediaItem` class example in Eclipse
- Example: duplicated code in 2 sibling subclasses



Oct 24, 2008

Sprenkle - CS209

Duplicated Code

- Example: duplicated code in 2 sibling subclasses
 - Extract method, put into parent class
 - If similar but not duplicate, extract the duplicate code (or parameterize)
- Example: duplicated code in unrelated classes

Oct 24, 2008

Sprenkle - CS209

7

Duplicated Code

- Example: duplicated code in unrelated classes
 - Ask: where does method belong?
 - One solution:
 - Extract class
 - Use new class in classes
 - Another solution:
 - Keep in one class
 - Other class calls that method

Oct 24, 2008

Sprenkle - CS209

8

Refactoring: Solution to Code Smells

- Updating a program to improve its design and maintainability *without changing its current functionality significantly*
- Example
 - Creating a single function that replaces 2 or more sections of similar code
 - Reduces redundant code
 - Makes code easier to debug, test

After refactoring your code, what should you do next?

Oct 24, 2008

Sprenkle - CS209

9

Long Methods

- What's the problem with long methods?
- What made us write them?

Oct 24, 2008

Sprenkle - CS209

10

Long Methods

- Hard to understand (see) what method does
- Smaller methods have reader overhead
 - Look at code for called methods
 - But, should use descriptive names
- Solutions:
 - Find lines of code that go together (may be identified by a comment) and extract method

Oct 24, 2008

Sprenkle - CS209

11

Large Class

- Too many instance variables → trying to do too much
- Solutions:
 - Bundle groups of variables together into another class
 - Look for common prefixes or suffixes
 - If includes optional instance variables (only sometimes used), create subclasses
 - Look at how users use the class for ideas of how to break it up

Oct 24, 2008

Sprenkle - CS209

12

Long Parameter List

- More difficult to use (do I have everything?)
- If method signature changes, have a lot of places to change
- Solutions: Use objects
 - Instead of separate parameters for an object's data
 - Group parameters together

Oct 24, 2008

Sprenkle - CS209

13

Literals or Magic Numbers

- If a number has a special meaning, make it a constant
 - Distinguish between 0 and NO_VALUE_ASSIGNED
 - If value changes (-1 instead of 0), only one place to change

Oct 24, 2008

Sprenkle - CS209

14

Divergent Change & Shotgun Surgery

Problem: when make a change, can't identify single point to make change

Divergent Change

- Problem: one class commonly changed in different ways for different reasons
- Solution:
 - Identify changes for a particular cause
 - Put into a class

Shotgun Surgery

- Problem: a change causes changes in many classes
- Solution:
 - Identify class that changes should belong to

Goal: 1-to-1 mapping of common changes to classes

Oct 24, 2008

Sprenkle - CS209

15

Data Clumps

- You have some data that always "hangs out together"
 - Maybe they should be an object
 - Check: if you deleted one of those pieces of data. Would the others make sense?
 - If answer is no, should be an object

Oct 24, 2008

Sprenkle - CS209

16

Message Chaining

- Dynamic coupling:
getOrder().getCustomer().getAddress().getState()
- Problem: others have too much knowledge of how your class works
 - Order class has a Customer object...
 - Depends on too many other classes
- Fix: add delegate method
 - Example: add method getShippingState()
 - Can go too far if adding too many methods

Oct 24, 2008

Sprenkle - CS209

17

Lazy Class

- Problem
 - Classes cost time, money to maintain & understand
 - Class doesn't do much
- How could this happen?
 - Refactoring!
 - Planned to be implemented but never happened
- Solution
 - Get rid of class
 - May need to collapse subclass into parent class

Oct 24, 2008

Sprenkle - CS209

18

Speculative Generality

- Beware of too much abstraction, allowing for too much flexibility that isn't required
- Solution: Collapse classes

Oct 24, 2008

Sprenkle - CS209

19

Comments

- Should be reserved for *why*, not what
- Problem:
 - Lots of comments about what the code or method is doing
- Solutions:
 - If need a comment for a block of code (or a long statement) → create a method with a descriptive name
 - If need a comment to describe method, rename method with more descriptive name

Oct 24, 2008

Sprenkle - CS209

20

Other Code Smells

- Discuss more code smells and solutions (Design Patterns) later

Oct 24, 2008

Sprenkle - CS209

21

Rules of Thumb

- Code smells are not **always** bad
 - Do not always mean code is poorly designed
- Open code is not **always** bad
- Need to use your judgment
 - Good judgment comes from experience.
 - How do you get experience? *Bad judgment* works every time

Goal: Gain experience to improve your judgment

Oct 24, 2008

Sprenkle - CS209

22

Eclipse's Refactor Menu

- MenuItem class, getPadding

Oct 24, 2008

Sprenkle - CS209

23

Discussion of Abstraction

- What does abstraction allow?
- Are there any limitations to abstraction?

Oct 24, 2008

Sprenkle - CS209

24

Metrics Plugin

- Provides information about your classes
 - # of classes
 - # of lines of code per method
 - Coupling
 - ...
- <http://metrics.sourceforge.net>

Oct 24, 2008

Sprenkle - CS209

25

Bin-Fitting Problem

- Classic CS problem: fit as many of something (A) into as few (B) as possible
- Example
 - A: Files, which have a size
 - B: CDs or DVDs (Disks)



Oct 24, 2008

Sprenkle - CS209

26

Heuristics

- Worst fit
 - Store file in disk with most free space
- In-order worst fit
 - Put files on disk, in order seen
- In-decreasing-order worst fit
 - Sort files by size
 - Put on disks

Oct 24, 2008

Sprenkle - CS209

27

Finding the Disk With Most Free Space

- Keep the disks in sorted order by their free space
 - Java class: `PriorityQueue`
 - Uses `compareTo` method or `Comparator`

Oct 24, 2008

Sprenkle - CS209

28

Getting A Solution

- Import → General → Existing project into Workspace
 - Archive file: `/home/courses/cs209/handouts/bins.tar`
- Try running `Bin.java`
 - Run options
 - Argument: `data/example.txt`

Oct 24, 2008

Sprenkle - CS209

29

Refactoring Discussion

Looking at the `main` method on the handout...

- How clearly written is the code?
- What, if any, comments might be helpful within the code?
- Does it satisfy its role as a tutorial?
- What, if any, suggestions does this code make about how the remaining parts of the assignment will be written?
- How would you test this code for bugs?

Oct 24, 2008

Sprenkle - CS209

30

Assignment 10: Code Critique & Refactoring

- Given: a problem specification and a solution to the problem
 - You refactoring your own code is emotional
 - More objective with someone else's solution
- Goals
 - Read and understand someone else's code
 - Haven't done much of this in Java
 - Critique code (do you smell something?)
 - Identify, articulate problems
 - Refactor code to solve problems identified
 - Write tests to verify the code

Oct 24, 2008

Sprenkle - CS209

31