

## Objectives

- Refactoring
- Code Critique
  - Identifying smells
  - Refactoring for readability

Start up Eclipse for later exercise

Oct 28, 2011

Sprenkle - CSCI209

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?
- What do you think of JUnit in Eclipse?

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

Oct 28, 2011

Sprenkle - CSCI209

2

## Review

- What goal are we designing to?
- What are some principles of design in Object-oriented Programming?
- What is the typical fix for code smells?
  - What is a limitation of those fixes?
- How do we address change in general?

Oct 28, 2011

Sprenkle - CSCI209

3

## 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

Eclipse: Refactor → Extract Constant

Oct 28, 2011

Sprenkle - CSCI209

4

## Divergent Change & Shotgun Surgery

**Problem:** when make a change, can't identify single point in code 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 (extract)

### Shotgun Surgery

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

**Goal:** I-to-I mapping of common changes to classes

Oct 28, 2011

Sprenkle - CSCI209

5

## Data Clumps

- **Problem:** You have some data that always "hangs out together"
- **Possible Solution:** 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

Eclipse: Refactor → Extract Class

Oct 28, 2011

Sprenkle - CSCI209

6

## Message Chaining

- Dynamic coupling:  
`getOrder().getCustomer().getAddress().getState()`
- Problem: client coupled to navigation structure
  - Depends on too many other classes
  - Change to intermediate class → Change in this class
- Fix: add delegate method
  - Example: add method `getShippingState()`
  - Can go too far if adding too many methods

Eclipse: Check references  
Refactor → Extract Method

Oct 28, 2011

7

## Middle Man

- Issue: Many methods of one class are delegating to another class
- How could this happen?
  - Refactoring!
- Possible Solutions
  - Inline method into caller
  - If there is additional behavior, replace delegation with inheritance to turn the middle man into a subclass of the real object

Oct 28, 2011

Sprenkle - CSCI209

8

## Lazy Class

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

Oct 28, 2011

Sprenkle - CSCI209

9

## Speculative Generality

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

Oct 28, 2011

Sprenkle - CSCI209

10

## Comments

**Problem:** Comments used as Febreze to cover up smells

- Describe what the code or method is doing
  - Should be reserved for *why*, not what
  - 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
- These comments are different from API comments

Oct 28, 2011

Sprenkle - CSCI209

11

## Olympics Scores Refactored

```
public static void main(String argv[]) {
    Scanner scan = new Scanner(System.in);

    double difficulty_score =
        getValidDifficultyScoreFromUser(scan);

    double[] execution_scores = readScoresFromUserFile(
        scan);

    double exec_avg = calculateAverageExecutionScore(
        execution_scores);

    displayResults(difficulty_score, execution_scores,
        exec_avg);
}
```

Oct 28, 2011

Sprenkle - CSCI209

12

## More Code Smells

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

Oct 28, 2011

Sprenkle - CSCI209

13

## 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 28, 2011

Sprenkle - CSCI209

14

## Refactoring Practice

Bookstore application

```

if (type.equals("topic")){
    String array[] = new String[2];
    for (int i = 0; i < bookNum; i++){
        Book currentBook = books.get(i);
        if (currentBook.getTopic().equals(argument)){
            flag += 1;
            array[0] = "true";
            if (flag == 1) //if we append each time, there will
                // be a 'null' at the beginning
                array[1] = "Title: " + currentBook.getTitle() + "\n"
                    + "Item Number: " + currentBook.getItemNumber();
            else array[1] += "Title: " + currentBook.getTitle() +
                "\n" + "Item Number: " + currentBook.getItemNumber();
        }
    }
    if (flag == 0){
        array[0] = "false";
        array[1] = "No items match your query\n";
    }
}

```

Oct 28, 2011

Sprenkle - CSCI209

17

## Refactoring Practice

Refactoring is a general concept: applies to Python too!

```

if prompt.split(" ")[0] == "buy" and nextCond == False:
    result = server.FrontEnd.buy(int(prompt.split(" ")[1]))
    array = server.FrontEnd.lookup(int(prompt.split(" ")[1]))

    if result == True:
        nextCond = True
        print "bought book " + str(array[0])

    else:
        nextCond = True
        print "Book out of stock or doesn't exist"

```

Oct 28, 2011

Sprenkle - CSCI209

16

Refactoring for Readability

## CODE CRITIQUE

Oct 28, 2011

Sprenkle - CSCI209

17

## Refactoring for Readability

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."  
 -- Martin Fowler  
 "Refactoring: Improving the Design of Existing Code"

Oct 28, 2011

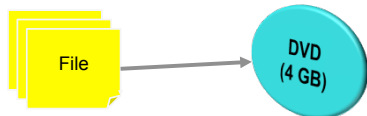
Sprenkle - CSCI209

18

## 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)

Another example:  
jobs on CPUs



How could we solve this problem?

Oct 28, 2011

Sprenkle - CSCI209

## One Heuristic: Worst Fit

- General idea: Store file in disk with most free space
- **In-order** worst fit
  - Put files on disk, in order seen
- **In-decreasing-order** worst fit
  1. Sort files by size
  2. Put on disks

Oct 28, 2011

Sprenkle - CSCI209

20

## Worst Fit: Finding Disk With Most Free Space

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

Oct 28, 2011

Sprenkle - CSCI209

21

## 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 28, 2011

Sprenkle - CSCI209

22

## Refactoring Discussion

Looking at `Bins` `main` method on 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 will be written?
- How would you test this code for bugs?

Oct 28, 2011

Sprenkle - CSCI209

23

## Pair Discussion of `Bins` Solution

- What does the code do?
  - What is the purpose/responsibility of each class?
- What are the good parts of the code?
- What are some of the code smells?

Oct 28, 2011

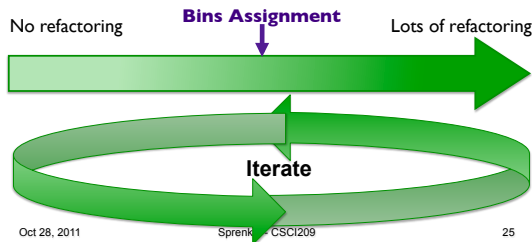
Sprenkle - CSCI209

24

## Notes on Assignment 9

- No “right” answer
  - Many design decisions
  - **Defend your design decision** in code critique

**Focus: Readability**



## Assignment 9: Code Critique & Refactoring

- Given: a problem specification and a solution to the problem
  - 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

**Due Wed**

Oct 28, 2011

Sprenkle - CSCI209

26

## TODO

- Assignment 9
- Extra Credit Opportunity
  - Leyburn Library, 1:30 – 2:30 p.m. today
  - Ask CS students about their summer research
  - 5 pts per poster
  - Send me email about
    - What is the problem they are trying to solve?
      - Motivation for solving problem
    - What was the biggest challenge they had to try to solve?

Oct 28, 2011

Sprenkle - CSCI209

27