页面 / Home

Code Smells

由 Triphol "Pao" Nilkuha (admin)創建于八月 20, 2014

Code Smells

What is a code smell?

According to Wikipedia, a code smell is "any symptom in the source code of a program that possibly indicates a deeper problem". Code smells tend to be patterns that commonly show up in source code that when fixed often lead to better more maintainable, reliable, and cleaner code. The following is an incomplete list of common code smells with examples and suggested solutions for fixing them. Over the course of the semester, we encourage you to look for these smells in your code and other students code and do your best to fix them in your assignments.

Duplicate Code

What is it: When segments of source code are repeated throughout the program.

How to fix it:

Type	Solution
Duplicate Methods in subclasses	Move code to superclass, create a superclass if needed
Duplicate expressions in superclass	Extract duplicates into their own methods
Duplicate expressions in different classes	Extract duplicates to a common component

Long Methods/Functions

What is it: When methods or functions are excessively long

How to fix it:

Type	Solution
Code that will not fit on a page	Extract functions from long fragments
Can't think of the function all at once	Extract into several smaller functions, add comments

Large Classes

What is it: Any class with more than 6-8 functions and 12-14 variables

How to fix it: split into component classes, create superclasses

Long Parameter List

What is it: When a function or method has too many parameters (generally more than 3-4)

How to fix it: Introduce a parameter object in place of many parameters to a function, but this is only worth doing if there are several functions with the same parameters, could also use a dynamic parameter object that is multipurpose (think Java Properties object)

Message Chain

What is it: When you call several functions successively such as:

```
person.getAddress().getZip();
```

How to fix it: Replace commonly called chains with helper functions such as:

```
person.getZip();
```

Feature Envy

What is it: When code wants to be in a different class, such as:

```
csDept.getFaculty().add(newProfessor);
csDept.setFacultyCount(csDept.getFacultyCount()+1)
```

How to fix it: Create a composite function that handles all necessary actions, such as:

```
csDept.addFaculty(newProfessor);
```

that handles the above two statements.

Switch statements, nested ifs

What is it: The use of switch statements where unnecessary, when if statements are deeply nested (more than 2 deep)

How to fix it:

1. Replace with a method call
2. Make subclasses for each case
3. Try to keep nesting to at most two levels

Temporary Fields

What is it: When instance variables are only used for part of the lifetime of an object

How to fix it: Change those instance variables into local variables to where they are used or move them to another object that better suits them

Refused Bequest

What is it: A is subclass of B, A overrides methods of B, does not use some inherited methods and fields of B

How to fix it: Give A and B a common superclass and move what A and B both use into it

Too Many Bugs

What is it: When functionality of your assignment suffers due to too many bugs in the code

How to fix it: Unit test to find bugs, fuzz your application with various inputs to test all possible cases

Too hard to understand

What is it: When your source code is not easily understood when read by someone reading it for the first time

How to fix it:

1. Use descriptive variable names (example: rowIndex instead of i in for loops)
2. Use many meaningful comments to guide reader through the code

Too hard to change

What is it: When your code becomes too hard to change when one of its specifications changes. Examples include:

1. a change in input format
2. a change in output format
3. a change in internal data structures
4. a change in communications format/protocol

How to fix it: Modularize your code - make more classes that each expose an interface but hide their internal algorithms and data structures. Some example modules that you might include in your assignments may be:

1. a module that only handles input
2. a module that only handles output
3. modules that each perform a piece of the program's logic

Using a decomposition similar to this, if you changed any of the above specifications, you would only have to change one module of your code, rather than the entire program source.

无标签