# CS409 Assignment 1

## Overview

In the JavaParser program I have chosen several different problems to analyse. Below is a list of the chosen problems.

* Large Class
* Long Method
* Long Parameter List
* Primitive Obsession
* Data Class

## Outline

In order to make the system as user-friendly as possible, a command line entry system is used so the user can specify which code smell they would like to test for in each directory. If the user would like to test a different directory they should change the file path in the JavaParser class.

Large Class

Large classes, like long methods, are difficult to read, understand and troubleshoot. If the class has too many responsibilities, you should question if I can be broken down into smaller, separate classes. It works by looking at the fields, method statements and constructors and adding up the statements to check if the threshold is no larger than 100 statements.

Long Method

A longer method is often harder to understand and troubleshoot. Where possible refactor into smaller method, and more modular code. It works by looking at the number of statements in the method and also the blockStmt within the method and checks that the number of statements is no larger than 10.

Long Parameter List

The more parameters a method has, the more complex it is. It works by looking at the number of parameters in each method and checks that is no longer are no more than 5 parameters.

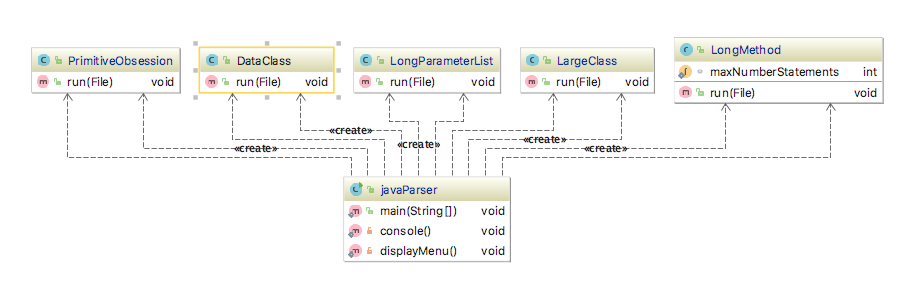
Primitive Obsession

The idea being you shouldn’t use primitive data types consistently over a class. If your data is sufficiently complex, create a c class to represent it. Primitive Obsession works by looking at the number of variables and works out by percentage how many are primitive. If more than 70% of all variables are primitive, this is seen to be primitive obsession.

Data Class

Avoid classes that passively store data. Classes should contain data *and* methods to operate on that data, too. It works by checking firstly if the numbers of statements are less than 3, then checks the prefixes of the methods and if they are all getters and setters this Is likely a data class.

## Class Diagram



## Results

Failures

Due to the code smells I choose to investigate, according to the test system, the test directory “Couplers” was irrelevant.

(X) = unexpected

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Abusers | Bloaters | Dispensibles | False Positives |
| Large Class | Grid | pass all – grid in different folder? | Pass all | Pass |
| Long Method | All methods fail except mousePressed in SolitairePanel  BarnsleyFernTwo – createFernwithTemp | Bresenham -paintComponent()  BarnsleyFern -  paintComponent()  Fail | Eertree – eertree()  Ciphollas – c()  Luhn – luhnTest() | Pass all |
| Long Parameter List | Fail - Grid checkLine() | ManOrBoy - a() | pass all | pass all |
| Primitive Obsession | pass all | Bresenham | Eertree | pass |
| Data Class | Grid  BarnsleyFernTwo(X) | ManOrBoy(X)  BarnsleyFern(X) | Eetreel luhn(X) | Fail(unexpected) |

## Statement Score

Long Method – Returns expected results included nested statements in BlockStmt – 5/5 if this qualifies for medium?

Long Parameter List – Works as expected - 2/2

Large Class – Works as expected – 5/5 if also qualifies for medium? Gets nested statements.

Data Class – Very basic, didn’t have enough time to implement fully 1 or 2/5

Primitive class – Also didn’t have enough time to fully implement and had problems accessing variables inside blockStatements 1/5

Overall estimated result – 12/20