PyReflect

```
$ pyreflect _
```

CodeSniffers

Chris Buonocore, Meghana Ginjpalli, Han Wang

6/6/2016

Motivation

“A code smell is a surface indication that usually corresponds to a deeper problem in the system”

~ Martin Fowler

- The presence of code smells indicates a need for refactoring

Goals

- Create PyReflect
  - Use Python to analyze code and detect bad smells in Java
- Detect well-known bad coding practices and provide opportunities for refactoring
- Propose ideas for enhancement: User defined rules, simple refactoring suggestions, and additional visualization models
- Compare and evaluate PyReflect against the well known Java static analysis tools (JDeodorant and PMD)

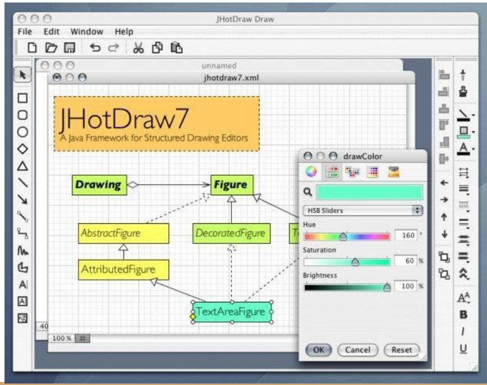
Approach

1. Use Plyj: A Java syntax parser written in Python
2. Traverse and parse source code AST for code smells
    - Long Parameter List
    - Long Method
    - God Class
    - Duplicated Code
    - Lazy Class

3. Represent source files as JSON trees for visualization

Datasets

- Small
  - 15 single file examples (50 LOC) of extracted code smells
  - Examples taken from Jason Gorman's best code practices workshop, Codemanship.
  - Used for Ground Truth baseline analysis of smell detection.
- Medium
  - Taken from Code Refactoring Course: https://github.com/danpersa/code-smell
  - Multi-file (8 files, 15-90 LOC) Java Project containing multiple, simultaneous code smells.
- Large
  - JHotDraw 7.0.6 - Real World Paint tool project written in Java.
    - LOC = 32,126
    - Number of methods = 3,377
    - Number of files = 309
    - Size = 23.9 MB



PyReflect vs. well-known tools

- JDeodorant
  - Eclipse plugin
  - Detects God Class, Feature Envy, Long Method, Type Checking
  - Provides refactoring recommendations and applies fixes to the corresponding code smells
- PMD
  - Eclipse plugin
  - Detects Large Class, Long Method, Long Parameter List, Duplicated Code
  - Detects over 300 potential problems (dead code, empty try, unused variables)
  - Allows user to select rules from a given list and set threshold values

Code Smell Results

Number of Detected Code Smells	Long Parameter List	Long Method	God Class	Duplicate Code	Lazy Class
JDeodorant	N/A	0	57	N/A	N/A
PMD	4	30	42	3	N/A
PyReflect	4	3	26	2	11

Discussion

- Long Parameter List
  - JDeodorant does not have this feature, but PMD and PyReflect agree
- Long Method
  - JDeodorant only considers long methods to be a bad smell if the method has the potential to be split into another function (Refactoring Opportunity)
  - PMD counts number of lines while PyReflect only counts actual statements and ignores whitespace
- God Class
  - JDeodorant only counts god class if it has the potential to be split into another class
  - PMD and PyReflect use the same metrics but PyReflect simplifies it
    - WMC > VeryHigh (47): Methods complexity
    - ATFD > Few (5): Access to foreign data
    - TCC < ⅓: Class cohesion

Discussion

- Duplicate Code
  - JDeodorant does not support this code smell
  - PMD is more powerful because it tokenizes the code and determines duplications within the whole directory
  - PyReflect looks for duplication only in the same file
- Lazy Class
  - JDeodorant and PMD do not support this code smell
  - PyReflect reports several classes that are doing minimal work and could be merged together

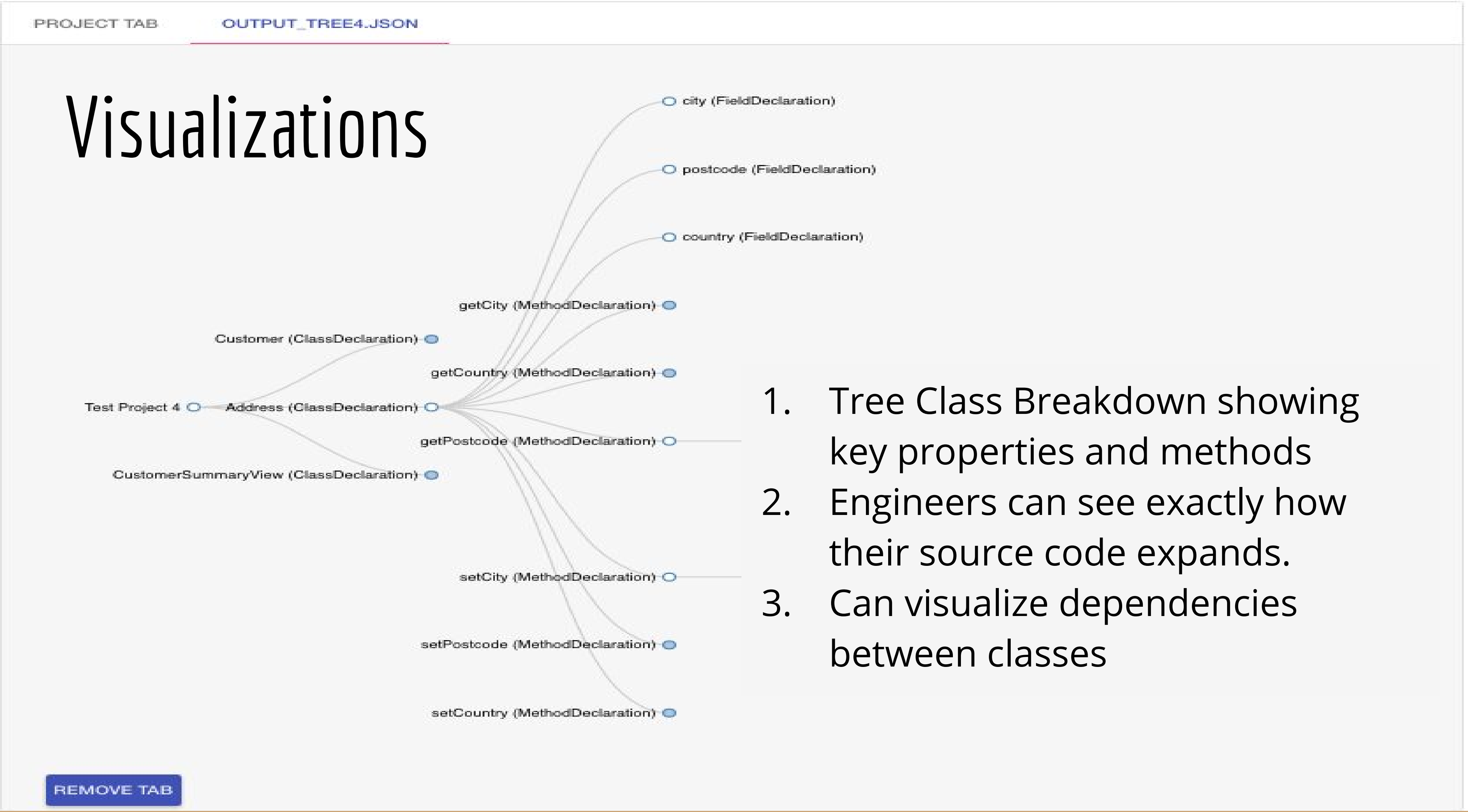
Discussion

- JDeodorant
  - Automatic refactoring
  - Limited smells
  - Takes longer to run
- PMD
  - Better performance: Duplicate Code
  - Hard to use
    - Numerous (>300) but trivial smells
    - Duplicate Code is in a separate module
    - Rebuild after changing ruleset and parameters
- PyReflect
  - Easy to use: Command-line tool
  - Reasonable smells
  - Improvement needed: Duplicate Code

Time (seconds)	JDeodorant	PMD	PyReflect
Parsing	25.46	17.34	14.85
Detecting	85.67	1.56	1.60

Time Consumed for God Class

Visualizations


PROJECT TAB
OUTPUT_TREE4.JSON
city (FieldDeclaration)
postcode (FieldDeclaration)
country (FieldDeclaration)
getCity (MethodDeclaration)
Customer (ClassDeclaration)
getCountry (MethodDeclaration)
Test Project 4
Address (ClassDeclaration)
getPostcode (MethodDeclaration)
CustomerSummaryView (ClassDeclaration)
setCity (MethodDeclaration)
setPostcode (MethodDeclaration)
setCountry (MethodDeclaration)
REMOVE TAB

1. Tree Class Breakdown showing key properties and methods
2. Engineers can see exactly how their source code expands.
3. Can visualize dependencies between classes

Possible Threats to Validity

- **External** Can PyReflect's AST model easily extend to other languages?
- **Internal** How does PyReflect handle more complex code smells? Is there possible confusion that could cause misclassification and/or smells being missed?
- **Construct** Is our evaluation dataset legitimate enough to be used as benchmarks for the indicated code smells? Is there a universal definition of all the code smells or does smell detection require human judgment or review?
- **Conclusion** Are JDeodorant and PMD the code smell detection standard for Java? Is this comparison reasonable?

Future Work

- Expand PyReflect to support more complex code smell detection
- Enable the user to define their own smell detection templates
- Perform a more-detailed performance comparison such as time scale and project structure complexity
- Add exportable and editable visualizations - allowing edits to the underlying code and API’s through the visual diagrams

Demo