User Study of Tools to Assist Java Programmers in Finding Bugs

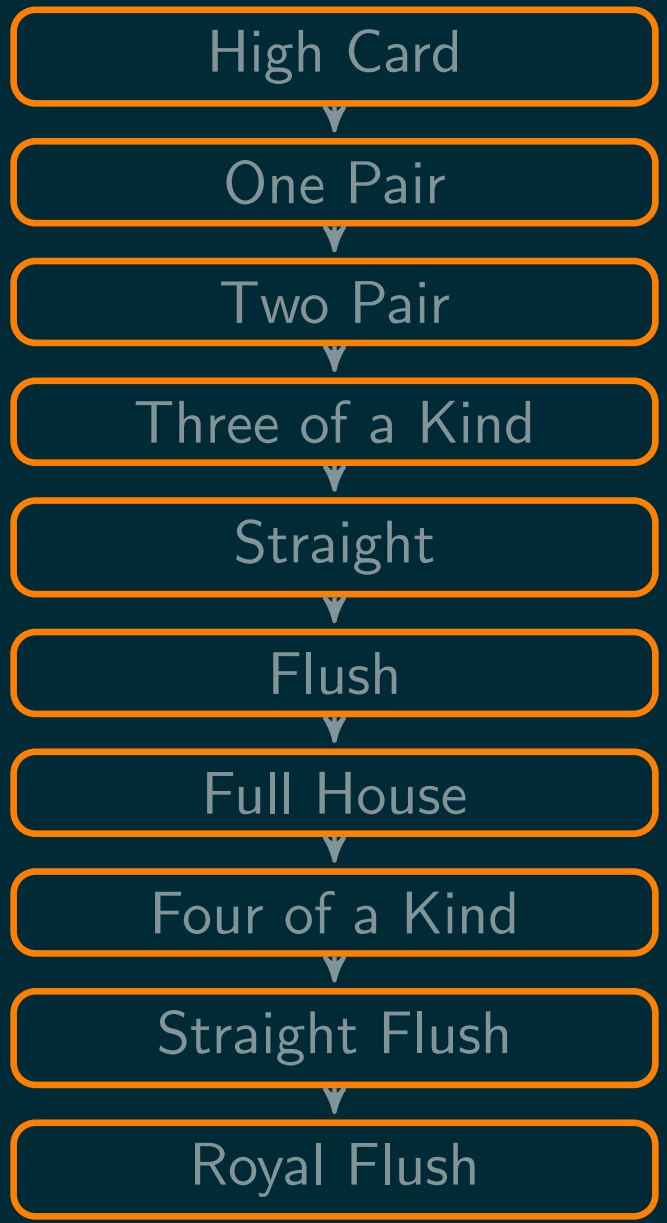
Andreas Bach Landgrebe

Department of Computer Science
Allegheny College

November 24, 2015

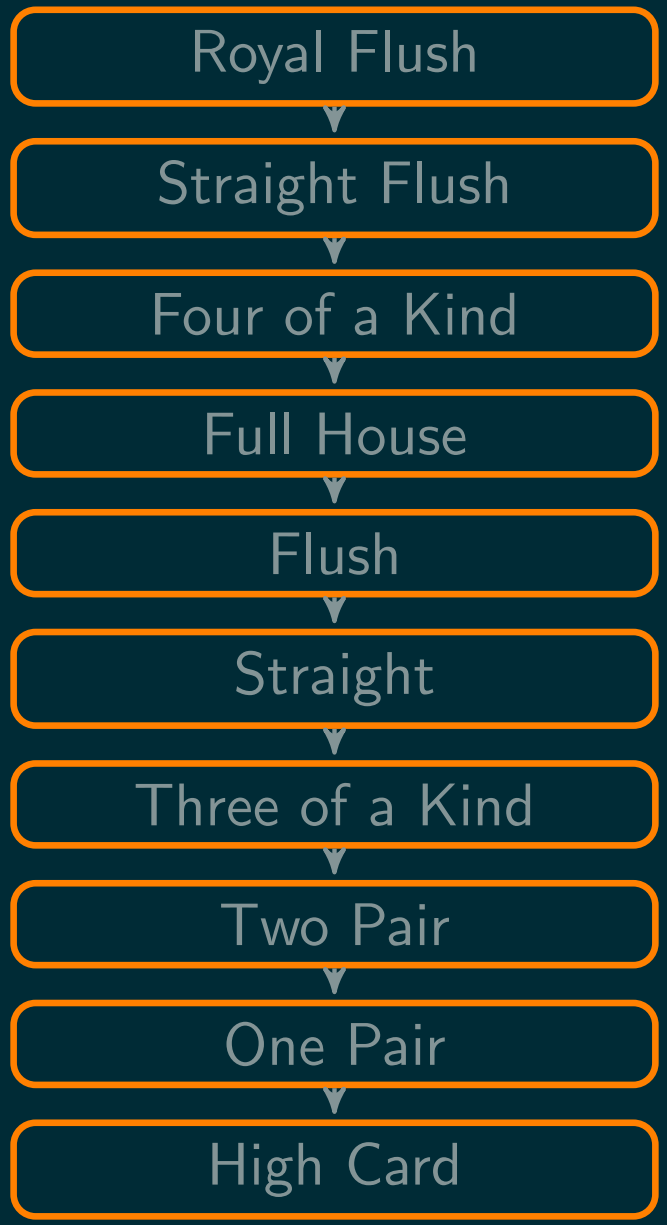
Motivation

Wrong Poker Representation



Motivation

Correct Poker Representation



Motivation

Definitions

- Logic Bug: is a bug in a program that causes it to operate incorrectly but not to terminate or crash.

Motivation

Definitions

- Logic Bug: is a bug in a program that causes it to operate incorrectly but not to terminate or crash.
- Static Program Analysis: analysis of computer software that is performed without actually executing programs.
- Dynamic Program Analysis: analysis of computer software that is performed by executing programs on a real or virtual processor.

Motivation

Why I decided this project

- Logic errors are the most difficult to find and fix.

Motivation

Why I decided this project

- Logic errors are the most difficult to find and fix.
- There is not one tool that will do everything for the programmer.

Motivation

Why I decided this project

- Logic errors are the most difficult to find and fix.
- There is not one tool that will do everything for the programmer.
- These three tools are the most popular pieces of software when looking for logic-based bugs in Java.

Overview of Project

- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.

Overview of Project

- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.
- Eclipse Integrated Development Environment

Overview of Project

- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.
- Eclipse Integrated Development Environment
  - Findbugs

Overview of Project

- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.
- Eclipse Integrated Development Environment
  - Findbugs
  - PMD

Overview of Project

- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.
- Eclipse Integrated Development Environment
  - Findbugs
  - PMD
  - checkstyle

Overview of Project

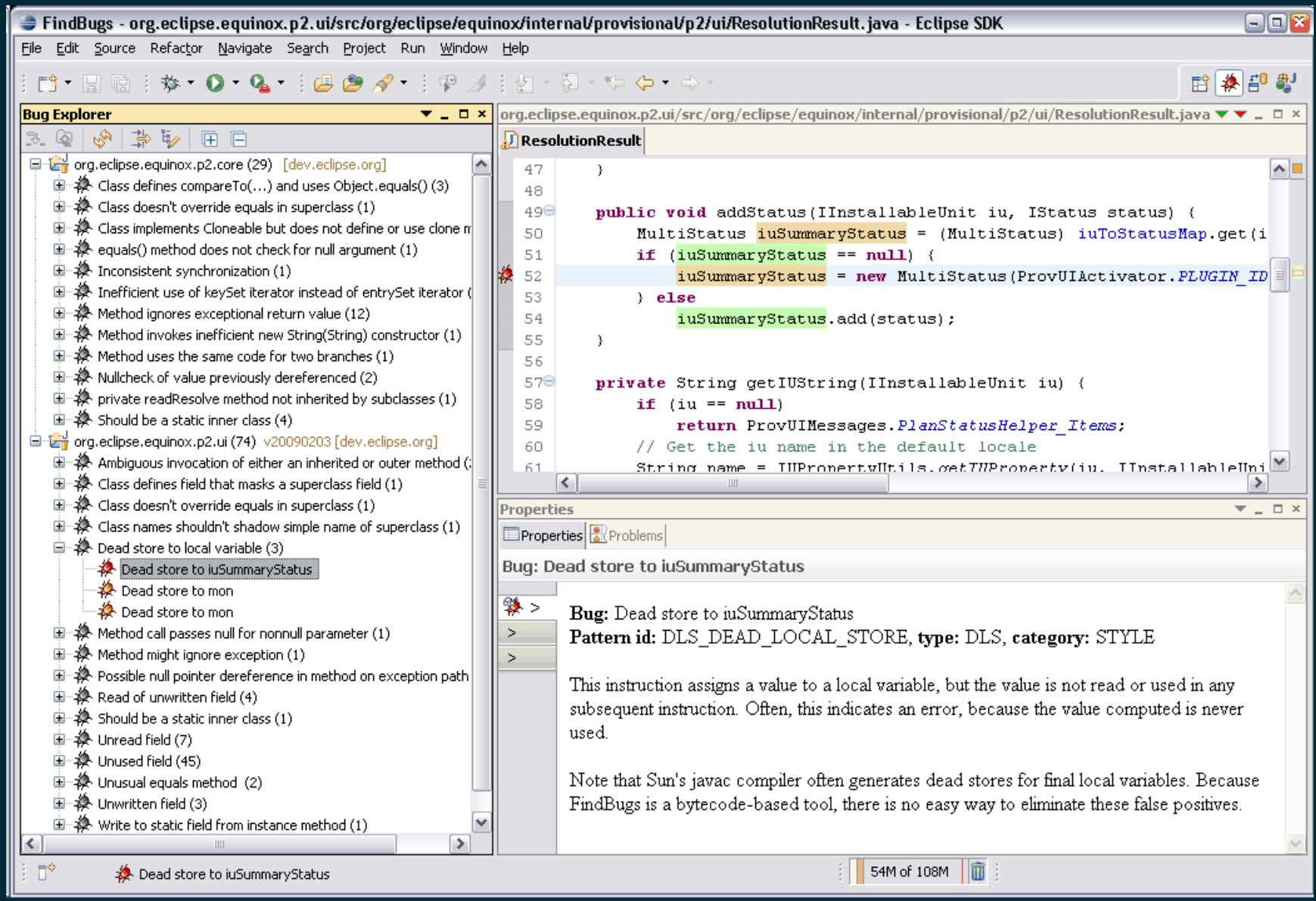
- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.
- Eclipse Integrated Development Environment
  - Findbugs
  - PMD
  - checkstyle
- Implementation of five Java files with buggy code.

Overview of Project

- Three Static Code Analysis Tools
- Evaluate each of these three tools and provide an evaluation on how each tool operates.
- Eclipse Integrated Development Environment
  - Findbugs
  - PMD
  - checkstyle
- Implementation of five Java files with buggy code.
- Human Study starting with students of Computer Science 112.

Tools

Findbugs



Tools

Features of FindBugs

- Operates on Java bytecode

Tools

Features of FindBugs

- Operates on Java bytecode
- Additional rule sets can be plugged in.

Tools

Features of FindBugs

- Operates on Java bytecode
- Additional rule sets can be plugged in.
- Bugs Rank 1-20

Tools

Features of FindBugs

- Operates on Java bytecode
- Additional rule sets can be plugged in.
- Bugs Rank 1-20
  - Rank 1-4: Scariest

Tools

Features of FindBugs

- Operates on Java bytecode
- Additional rule sets can be plugged in.
- Bugs Rank 1-20
  - Rank 1-4: Scariest
  - Rank 5-9: Scary

Tools

Features of FindBugs

- Operates on Java bytecode
- Additional rule sets can be plugged in.
- Bugs Rank 1-20
  - Rank 1-4: Scariest
  - Rank 5-9: Scary
  - Rank 10-14: Troubling

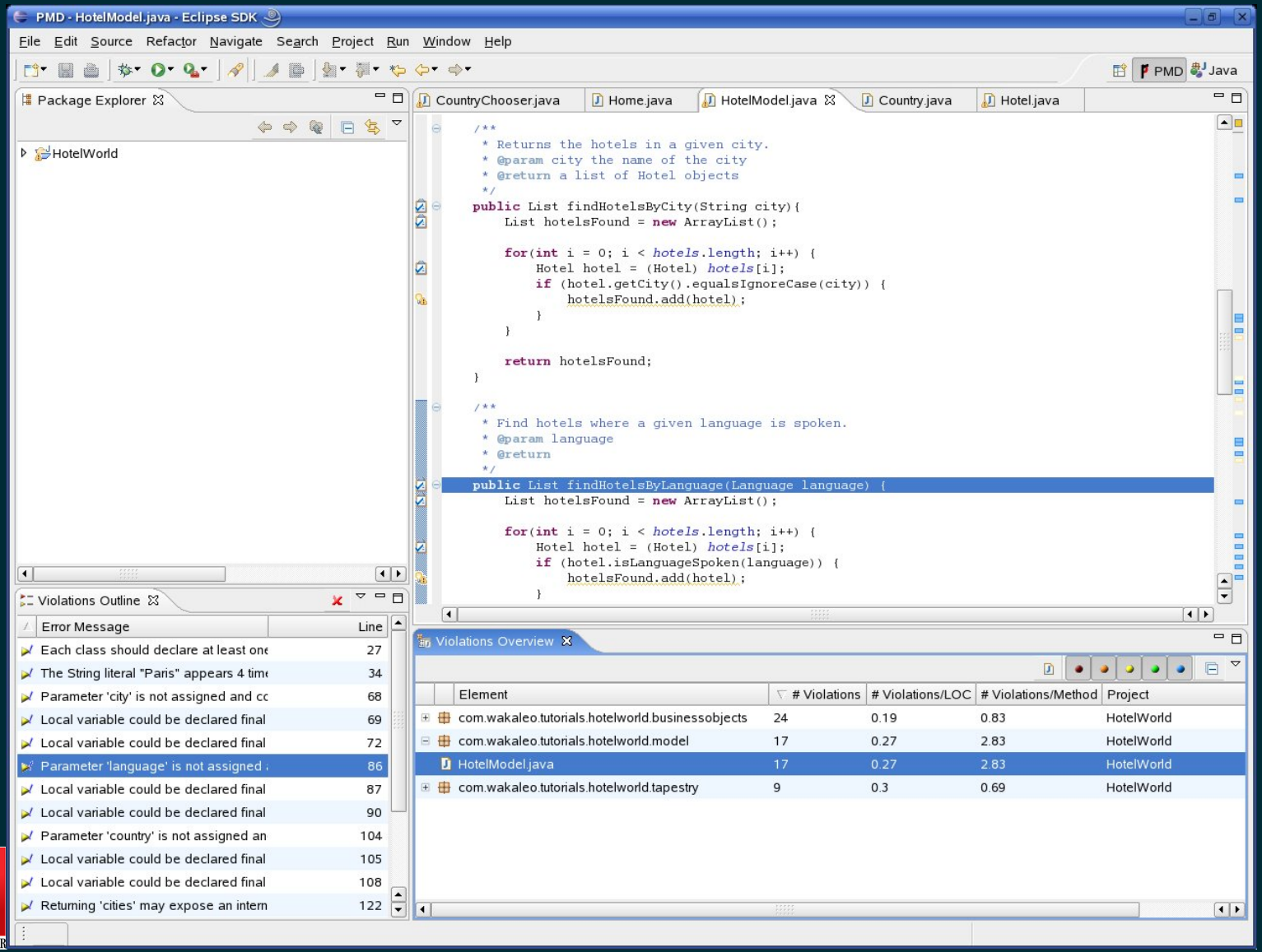
Tools

Features of FindBugs

- Operates on Java bytecode
- Additional rule sets can be plugged in.
- Bugs Rank 1-20
  - Rank 1-4: Scariest
  - Rank 5-9: Scary
  - Rank 10-14: Troubling
  - Rank 15-20: Concern

Tools

PMD


PMD - HotelModel.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
PMD Java
Package Explorer
HotelWorld
CountryChooser.java Home.java HotelModel.java Country.java Hotel.java
/**
* Returns the hotels in a given city.
* @param city the name of the city
* @return a list of Hotel objects
*/
public List findHotelsByCity(String city){
List hotelsFound = new ArrayList();
for(int i = 0; i < hotels.length; i++) {
Hotel hotel = (Hotel) hotels[i];
if (hotel.getCity().equalsIgnoreCase(city)) {
hotelsFound.add(hotel);
}
}
return hotelsFound;
}
/**
* Find hotels where a given language is spoken.
* @param language
* @return
*/
public List findHotelsByLanguage(Language language) {
List hotelsFound = new ArrayList();
for(int i = 0; i < hotels.length; i++) {
Hotel hotel = (Hotel) hotels[i];
if (hotel.isLanguageSpoken(language)) {
hotelsFound.add(hotel);
}
Violations Outline
Error Message Line
Each class should declare at least one 27
The String literal "Paris" appears 4 time 34
Parameter 'city' is not assigned and cc 68
Local variable could be declared final 69
Local variable could be declared final 72
Parameter 'language' is not assigned 86
Local variable could be declared final 87
Local variable could be declared final 90
Parameter 'country' is not assigned an 104
Local variable could be declared final 105
Local variable could be declared final 108
Returning 'cities' may expose an intern 122
Violations Overview
Element # Violations # Violations/LOC # Violations/Method Project
com.wakaleo.tutorials.hotelworld.businessobjects 24 0.19 0.83 HotelWorld
com.wakaleo.tutorials.hotelworld.model 17 0.27 2.83 HotelWorld
HotelModel.java 17 0.27 2.83 HotelWorld
com.wakaleo.tutorials.hotelworld.tapestry 9 0.3 0.69 HotelWorld


pmd
DON'T SHOOT THE MESSENGER

Tools

PMD

- Java source code analyzer

Tools
PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules

Tools

PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules
- Not true errors, but rather inefficient code
  - PMD possible flaws to detect
    - Possible bugs - Empty try/catch/switch blocks.
    - Dead code - Unused variables and private methods.

Tools

PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules
- Not true errors, but rather inefficient code
  - PMD possible flaws to detect
    - Possible bugs - Empty try/catch/switch blocks.
    - Dead code - Unused variables and private methods.
    - Empty if/while statements

Tools

PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules
- Not true errors, but rather inefficient code
  - PMD possible flaws to detect
    - Possible bugs - Empty try/catch/switch blocks.
    - Dead code - Unused variables and private methods.
    - Empty if/while statements
    - Overcomplicated expressions - for loops that could be while loops

Tools

PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules
- Not true errors, but rather inefficient code
  - PMD possible flaws to detect
    - Possible bugs - Empty try/catch/switch blocks.
    - Dead code - Unused variables and private methods.
    - Empty if/while statements
    - Overcomplicated expressions - for loops that could be while loops
    - Suboptimal code - Wasteful String usage

Tools

PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules
- Not true errors, but rather inefficient code
  - PMD possible flaws to detect
    - Possible bugs - Empty try/catch/switch blocks.
    - Dead code - Unused variables and private methods.
    - Empty if/while statements
    - Overcomplicated expressions - for loops that could be while loops
    - Suboptimal code - Wasteful String usage
    - Classes with high Cyclomatic Complexity measurements (complexity of a program).

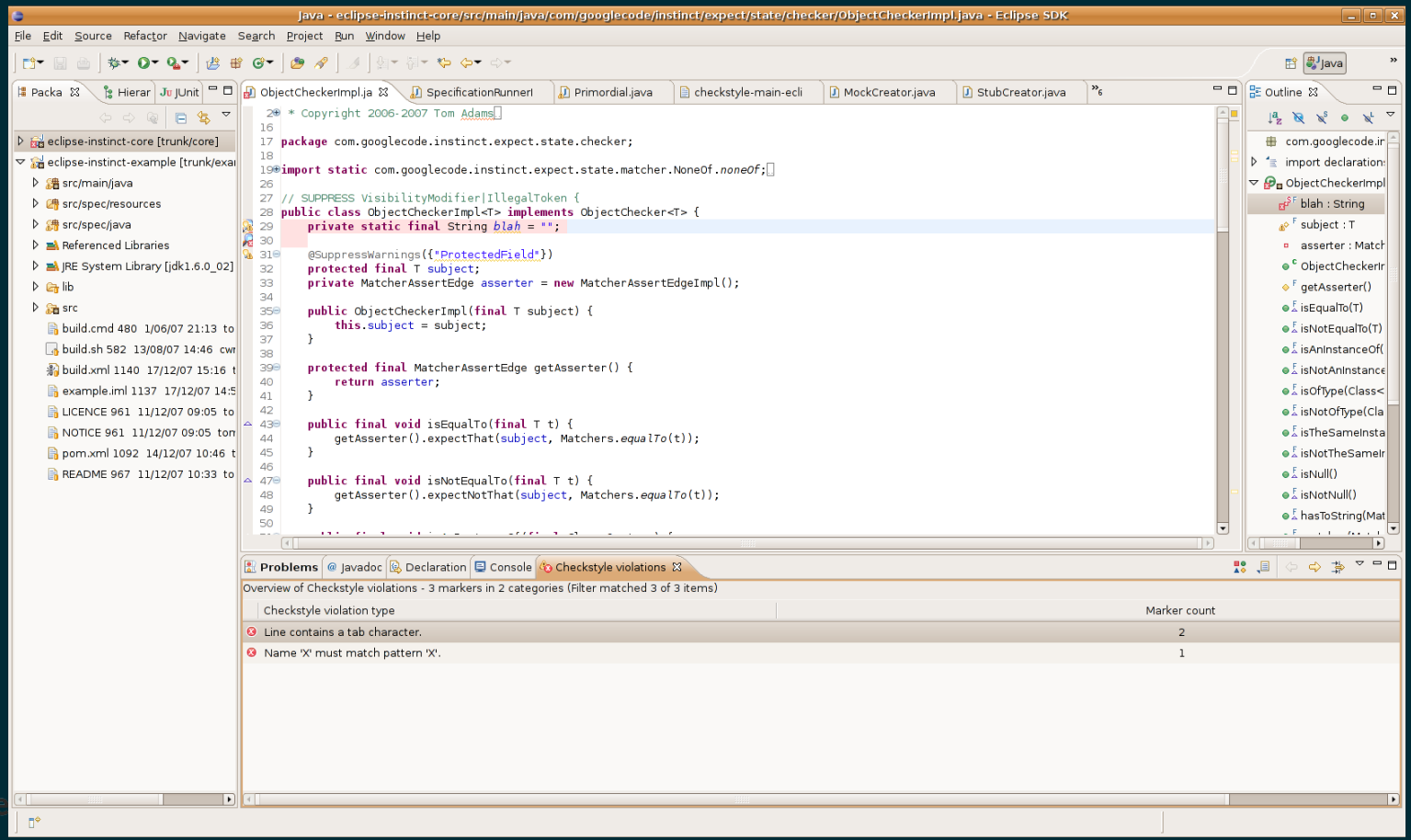
Tools

PMD

- Java source code analyzer
- Built-in rules and supports the ability to write custom rules
- Not true errors, but rather inefficient code
  - PMD possible flaws to detect
    - Possible bugs - Empty try/catch/switch blocks.
    - Dead code - Unused variables and private methods.
    - Empty if/while statements
    - Overcomplicated expressions - for loops that could be while loops
    - Suboptimal code - Wasteful String usage
    - Classes with high Cyclomatic Complexity measurements (complexity of a program).
    - Duplicated code - copied-pasted code could mean copied/pasted bugs.

Tools

Checkstyle



Tools

Checkstyle

- Java source code analyzer

Tools

Checkstyle

- Java source code analyzer
- Checks if Java source code adheres to a coding standard

Tools

Checkstyle

- Java source code analyzer
- Checks if Java source code adheres to a coding standard
- Highly configurable to support any coding standard

Tools

Checkstyle

- Java source code analyzer
- Checks if Java source code adheres to a coding standard
- Highly configurable to support any coding standard
- Designed to check for programming practices.

Evaluation Strategy

- Five Java programs with logic-based bugs

Evaluation Strategy

- Five Java programs with logic-based bugs
  - Misplaced semi-colon

Evaluation Strategy

- Five Java programs with logic-based bugs
  - Misplaced semi-colon
  - Velocity logic-bug

Evaluation Strategy

- Five Java programs with logic-based bugs
  - Misplaced semi-colon
  - Velocity logic-bug
  - String == vs. String equals() comparison bug

Evaluation Strategy

- Five Java programs with logic-based bugs
  - Misplaced semi-colon
  - Velocity logic-bug
  - String == vs. String equals() comparison bug
  - Memory leak bug

Evaluation Strategy

- Five Java programs with logic-based bugs
  - Misplaced semi-colon
  - Velocity logic-bug
  - String == vs. String equals() comparison bug
  - Memory leak bug
  - Switch case bug

Evaluation Strategy

Human Study Survey

- Did you find the bugs?

Evaluation Strategy

Human Study Survey

- Did you find the bugs?
- If yes, did Findbugs assist you in finding these bugs?

Evaluation Strategy

Human Study Survey

- Did you find the bugs?
- If yes, did Findbugs assist you in finding these bugs?
- If yes, did PMD assist you in finding these bugs?

Evaluation Strategy

Human Study Survey

- Did you find the bugs?
- If yes, did Findbugs assist you in finding these bugs?
- If yes, did PMD assist you in finding these bugs?
- If yes, did Checkstyle assist you in finding these bugs

Evaluation Strategy

Human Study Survey

- Did you find the bugs?
- If yes, did Findbugs assist you in finding these bugs?
- If yes, did PMD assist you in finding these bugs?
- If yes, did Checkstyle assist you in finding these bugs
- Would you use any of these tools in the future for finding logic-based bugs?

Evaluation Strategy

Human Study Survey

- Did you find the bugs?
- If yes, did Findbugs assist you in finding these bugs?
- If yes, did PMD assist you in finding these bugs?
- If yes, did Checkstyle assist you in finding these bugs
- Would you use any of these tools in the future for finding logic-based bugs?
- Suggestion to further improve these tools.

Evaluation Strategy

How To Evaluate Human Study

- Computer Science 112 Students

Evaluation Strategy

How To Evaluate Human Study

- Computer Science 112 Students
- Knowledge past Computer Science 111

Evaluation Strategy

How To Evaluate Human Study

- Computer Science 112 Students
- Knowledge past Computer Science 111
- 24 hours to evaluate and find the logic bugs

Evaluation Strategy

How To Evaluate Human Study

- Computer Science 112 Students
- Knowledge past Computer Science 111
- 24 hours to evaluate and find the logic bugs
- IRB approval and CITI Training

Deliverables

- Evaluation of how each of these three tools operates.

Deliverables

- Evaluation of how each of these three tools operates.
- User study of whether these three tools assisted in finding these bugs.

Deliverables

- Evaluation of how each of these three tools operates.
- User study of whether these three tools assisted in finding these bugs.
- Human Study of whether these three tools were able to help Java programmers finding logic bugs.

Deliverables

- Evaluation of how each of these three tools operates.
- User study of whether these three tools assisted in finding these bugs.
- Human Study of whether these three tools were able to help Java programmers finding logic bugs.
- Java source code with logic-based bugs

Conclusion

Thank You!

Comments, Questions or Concerns?