

FREECOL TACTITAL TEST PLAN

William Cosulich

Lauren Medley

Erica Muschelli

Jose Quinchuela

COSC 603

Table of Contents

Introduction

Features that will be tested

Approach to Testing/Improving Software

- a) Activities
- b) Criteria
- c) Testing tools

Criteria for pass/fail

Introduction

Features that will be tested

Approach to Testing/Improving Software

- d) Activities
- e) Criteria
- f) Testing tools

Criteria for pass/fail

Test Deliverables/status communication documents

- a) Test plan
- b) Test design specifications
- c) Test case specifications
- d) Test process
- e) Test logs.
- f) Test trouble reports
- g) Test summary reports
- h) Test input data and test output data(or where they are located)

Environmental Needs

- a) Hardware
- b) Software
- c) Any other software/supplies needed to run the test.

- d) Mode of usage (stand-alone, web based)
- e) Test tools needed.
- f) Any other testing needs. (publications)

Responsibilities

Schedule

Introduction

FreeCol is an Open Source version of Colonization. Colonization is heavily based on Civilization which some consider to be the best turn-based strategy game for the PC in the history of mankind. FreeCol differs from the original game in two regards: it supports multiplayer games and uses an isometric map. At some point in the future, the game might also add support for rectangular tiles similar to those used in the original game. Our team will be focusing on several suggested packages: client, control, gui and others. Our team will select and prioritize 20 classes to improve the quality of the code as well as to include test cases within the packages above mentioned. The test plan will help the team to report on refactoring activities, test deliverables, test tools and also identify overall team responsibilities.

Features to be tested.

The team decided on the 20 classes listed below which were prioritized according to severe bugs found, high code violations found, high code complexity, methods too long, styling and documentation and code coverage.

Feature Number	Package Name	Class Name
1	net.sf.freecol.metaserver	MetaServer
2	net.sf.freecol.common.model	Limit
3	net.sf.freecol.common.option	RangeOption
4	net.sf.freecol.common.option	SelectOption
5	net.sf.freecol.common.server.model	Unit
6	net.sf.freecol.server.ai	REFAIPlayer
7	net.sf.freecol.server.model	ServerUnit
8	net.sf.freecol.server.model	ServerColony
9	net.sf.freecol.client.gui.panel	DragListener
10	net.sf.freecol.client.gui.panel	BuildingDetailPanel
11	net.sf.freecol.client.gui.panel	BuildingToolTip
12	net.sf.freecol	FreeCol
13	net.sf.freecol.client	ClientOptions
14	net.sf.freecol.client.control	InGameController
15	net.sf.freecol.client.gui	FontLibrary
16	net.sf.freecol.client.gui	Choicelitem
17	net.sf.freecol.client.gui	FrameMentionListener
18	net.sf.freecol.client.gui	TerrainCursor
19	net.sf.freecol.common	FreeColSeed
20	net.sf.freecol.client.common	ServerInfo

Approach to Testing

The team's approach to testing was prioritized testing according to severe bugs found, high code violations found, high code complexity, methods too long, styling issues and documentation and code coverage. We used the following tools to help prioritize what classes would be tested. : FindBugs, PMD, Google Code Pro, JDeodorant, CheckStyle, JAutoDoc, EclEmma, and JUnit Testing.

Also the team added code coverage to classes other than the 20 that we prioritized that did had of 0% coverage

Feature1: MetaServer

Activities: Fix bug that was found, fix high code violation, lowered the Cyclomatic Complexity, Refactored Methods that were too long, Add JDoc documentation for the class.

Test Tools: FindBugs, PMD, JDeorderant, Google Code Pro, Eclipse refactoring tools, JAutoDoc

Feature2: Limit

Activities: Fix bug that was found, fix high code violation, lower the Cyclomatic Complexity, Refactored Methods that were too long, Add JDoc documentation for the class.

Test Tools: FindBugs, PMD, JDeorderant, Google Code Pro, Eclipse refactoring tools, JAutoDoc

Feature3: RangeOption

Activities: Fix bug that was found, fix high code violation, Add JDoc documentation for the class.

Test Tools: FindBugs, PMD, JAutoDoc

Feature4: SelectOption

Activities: Fix bug that was found, fix high code violation, Add JDoc documentation for the class.

Test Tools: FindBugs, PMD, JAutoDoc

Feature5: Unit

Activities: Fix high code violation, Add JDoc documentation for the class.

Test Tools: PMD, JAutoDoc

Feature6: REFAIPlayer

Activities: Fix high code violation, lower the Cyclomatic Complexity, Add JDoc documentation for the class.

Test Tools: PMD, CodePro, Eclipse refactoring tools, JAutoDoc

Feature7: ServerUnit

Activities: Fix high code violation, lower the Cyclomatic Complexity, Add JDoc documentation for the class.

Test Tools: PMD, CodePro, Eclipse refactoring tools, JAutoDoc

Feature8: Draglistener

Activities: Lower the Cyclomatic Complexity, Fix high code violation, Refactor Method Too Large, Add JDoc documentation for the class.

Test Tools: CodePro, Eclipse refactoring tools, JDeodorant, JAutoDoc

Feature9: ServerColony

Activities: Fix high code violation, Add JDoc documentation for the class.

Test Tools: PMD, JAutoDoc

Feature10: BuildingDetailPanel

Activities: Lower the Cyclomatic Complexity, Add JDoc documentation for the class.

Test Tools: CodePro, Eclipse refactoring tools, JAutoDoc

Feature10: BuildingToolTip

Activities: Lower the Cyclomatic Complexity, Add JDoc documentation for the class.

Test Tools: CodePro, Eclipse refactoring tools, JAutoDoc

Feature12: FreeCol

Activities: Add JUnit Test Cases, Clean up style issues, Add JDoc documentation for the class.

Test Tools: CodePro, JUnit, Eclipse refactoring tools, CheckStyle, Eclipse Formatter tool, JAutoDoc

Feature13: ClientOption

Activities: Add JUnit Test Cases, Clean up style issues, Add JDoc documentation for the class.

Test Tools: CodePro, JUnit, Eclipse refactoring tools, CheckStyle, Eclipse Formatter tool, JAutoDoc

Feature14: InGameController

Activities: Add JUnit Test Cases, Clean up style issues, Add JDoc documentation for the class.

Test Tools: CodePro, JUnit, Eclipse refactoring tools, CheckStyle, Eclipse Formatter tool, JAutoDoc

Feature15: FontLibrary

Activities: Add JUnit Test Cases and bring up the code coverage, Add JDoc documentation for the class

Test Tools: CodePro, JUnit, EclEmma, JAutoDocs

Feature16: ChoiceItem

Activities: Add JUnit Test Cases and bring up the code coverage, Add JDoc documentation for the class

Test Tools: CodePro, JUnit, EclEmma, JAutoDocs

Feature17: FrameMowntionListener

Activities: Add JUnit Test Cases and bring up the code coverage, Add JDoc documentation for the class

Test Tools: CodePro, JUnit, EclEmma, JAutoDocs

Feature18: TerrainCursor

Activities: Add JUnit Test Cases and bring up the code coverage, Add JDoc documentation for the class

Test Tools: CodePro, JUnit, EclEmma, JAutoDocs

Feature19: FreeColSeed

Activities: Add JUnit Test Cases and bring up the code coverage, Add JDoc documentation for the class

Test Tools: CodePro, JUnit, EclEmma, JAutoDocs

Feature20: ServerInfo

Activities: Add JUnit Test Cases and bring up the code coverage, Add JDoc documentation for the class

Test Tools: CodePro, JUnit, EclEmma, JAutoDocs

Extra Features: All activities for extra features:

NationSummary.java

Operand.java

Limit.java

IndianNationType.java

Disaster.java

NationOptions.java

Nation.java

NationType.java

BuidableType.java

AbstractUnit.java

ResourceType.java

AbstractGoods.java

Stance.java

Tension.java

Activities: Use eclEmma to improve code coverage and JUnit to generate test cases.

Test Tools: Eclipse refactoring tools, JUnit, EclEmma

Criteria for Pass/Fail

The criteria for passing was based on a number of items that also depended on what type of issues that were being addressed on the classes. Not all criteria for pass/fail was used for each class as not tests/ improvements were performed on all classes used. For the classes with bugs, criteria for passing was to removal of the severe bugs found while criteria for failure was not fixing the bugs. For the classes with high code violations, the criteria for passing removal of the code violation while the criteria for failure was to leave in the violations, For classes that we were lowering the Cyclomatic Complexity for, the criteria for passing was to have Cyclomatic Complexity be reduced at all while we did not have any failure criteria as we found that this could not apply to all class as their Cyclomatic Complexity was low to begin. For classes that had methods that were too long, the criteria for passing was the making their methods were refactored into smaller ones. Their criteria for failure was not completely refactoring their methods until their no-longer refactorable. Adding documentation pass or fail criteria was more of the we added the JDoc for the class or we didn't. For classes that we added JUnit test cases for, the passing criteria was that we made sure that all the test cases passed while the criteria for failure was that some of the classes failed. For classes that we had added code coverage for the pass criteria was that increased the coverage to over 50% while the failure criteria was that the coverage was less than 50%.

Feature1: MetaServer

Pass Criteria: Fixed bug found, high code violation fixed, lowered the Cyclomatic Complexity by any percent, refactored all Methods that were too long, added JDoc documentation.

Failure Criteria: Bug found not fixed, high code violation not fixed, left methods too long, didn't add any JDoc documentation.

Feature2: Limit

Pass Criteria: Fixed bug found, high code violation fixed, lowered the Cyclomatic Complexity by any percent, refactored all Methods that were too long, added JDoc documentation.

Failure Criteria: Bug found not fixed, high code violation not fixed, left methods too long, didn't add any JDoc documentation.

Feature3: RangeOption

Pass Criteria: Fixed bug found, high code violation fixed, added JDoc documentation.

Failure Criteria: Bug found not fixed, high code violation not fixed, didn't add any JDoc documentation.

Feature4: SelectOption

Pass Criteria: Fixed bug found, high code violation fixed, added JDoc documentation.

Failure Criteria: Bug found not fixed, high code violation not fixed , didn't add any JDoc documentation.

Feature5: Unit

Pass Criteria: High code violation fixed, added JDoc documentation.

Failure Criteria: high code violation not fixed, left methods too long, didn't add any JDoc documentation.

Feature6: REFAIPlayer

Pass Criteria: High code violation fixed, lowered the Cyclomatic Complexity by any percent, added JDoc documentation.

Failure Criteria: High code violation not fixed, didn't add any JDoc documentation.

Feature7: ServerUnit

Pass Criteria: high code violation fixed, lowered the Cyclomatic Complexity by any percent, added JDoc documentation.

Failure Criteria: high code violation not fixed, didn't add any JDoc documentation.

Feature8: Draglistener

Pass Criteria: high code violation fixed, lowered the Cyclomatic Complexity by any percent, refactored all Methods that were too long, added JDoc documentation.

Failure Criteria: high code violation not fixed, left methods too long, didn't add any JDoc documentation.

Feature9: ServerColony

Pass Criteria: high code violation fixedadded JDoc documentation.

Failure Criteria: high code violation not fixed, didn't add any JDoc documentation.

Activities: Fix high code violation, Add JDoc documentation for the class.

Test Tools: PMD, JAutoDoc

Feature10: BuildingDetailPanel

Pass Criteria: lowered the Cyclomatic Complexity by any percent, added JDoc documentation.

Failure Criteria: didn't add any JDoc documentation.

Feature10: BuildingToolTip

Pass Criteria: lowered the Cyclomatic Complexity by any percent, added JDoc documentation.

Failure Criteria: didn't add any JDoc documentation.

Feature12: FreeCol

Pass Criteria: All test cases passed, Cleaned style issues, added JDoc documentation.

Failure Criteria: Some test cases failed, Leave style issues found, didn't add any JDoc documentation.

Activities: Add JUnit Test Cases, Clean up style issues, Add JDoc documentation for the class.

Feature13: ClientOption

Pass Criteria: All test cases passed, Cleaned style issues, added JDoc documentation.

Failure Criteria: Some test cases failed, Leave style issues found, didn't add any JDoc documentation.

Feature14: InGameController

Pass Criteria: All test cases passed, Cleaned style issues, added JDoc documentation.

Failure Criteria: Some test cases failed, Leave style issues found, didn't add any JDoc documentation.

Feature15: FontLibrary

Pass Criteria: All test cases passed, increased coverage to over 50%, added JDoc documentation.

Failure Criteria: Some test cases failed, coverage less than 50%, didn't add any JDoc documentation.

Feature16: Choiceltem

Pass Criteria: All test cases passed, increased coverage to over 50%, added JDoc documentation.

Failure Criteria: Some test cases failed, coverage less than 50%, didn't add any JDoc documentation.

Feature17: FrameMowntionListener

Pass Criteria: All test cases passed, increased coverage to over 50%, added JDoc documentation.

Failure Criteria: Some test cases failed, coverage less than 50%, didn't add any JDoc documentation.

Feature18: TerrainCursor

Pass Criteria: All test cases passed, increased coverage to over 50%, added JDoc documentation.

Failure Criteria: Some test cases failed, coverage less than 50%, didn't add any JDoc documentation.

Feature19: FreeColSeed

Pass Criteria: All test cases passed, increased coverage to over 50%, added JDoc documentation.

Failure Criteria: Some test cases failed, coverage less than 50%, didn't add any JDoc documentation.

Feature20: ServerInfo

Pass Criteria: All test cases passed, increased coverage to over 50%, added JDoc documentation.

Failure Criteria: Some test cases failed, coverage less than 50%, didn't add any JDoc documentation.

Test Deliverables

The following items will be turned in upon completion of software improvements:

a) Test plan

Improved and complete FreeCol test plan.

b) Test process

Statement coverage report

c) Test logs.

Screenshots (coverage reports)

d) Test trouble reports

Test trouble reports are handled within GitHub in the issue generation feature of GitHub.

e) Test summary reports

Summary reports. We can have summary of how we tested the code. See the Schedule part for this.

f) Test input data and test output data(or where they are located)

See criteria for pass/fail that refer to the test input and output data.

Environmental Needs

a) Hardware

Dell laptop inspiron 1525 series
Dell laptop
HP laptop
Mac laptop

b) Software

Eclipse IDE Mars, Luna,
Java rei 1.8.0

c) Any other software/supplies needed to run the test.

Eclipse
Java Version 7 and above
Windows Vista and above
Mac O.S
GitHub

d) Mode of usage (stand-alone, web based)

Stand alone.

e) Test tools needed.

Google Code Pro Metrics
Code Pro Analytix
FindBugs
CheckStyle
PMD
JUnit
EclEmma
JAutoDocs,
Eclipse Formatter tool,
Eclipse Refactor tool,

f) Any other testing needs. (publications)

FreeCol docs, FreeCol FQA, FreeCol test cases.

Responsibilities

Identify the groups responsible for all aspects of testing and correcting problems

Erica: Setting up the repository and uploading the project on the repository for all to use.

Lauren/Erica:

Erica with assistance from Lauren work on the majority of the JUnit test cases that were created for the project. They also worked to increase the coverage for a number of classes. They selected files where coverage was 0.0% to show code coverage improvement when developing our test cases. Some GUI files were selected but they felt that those classes contained within the “common” files were a bit more significant since they seemed to be used the most for our test case development.

Test cases created to improve branch code coverage for the following:

1. Net.sf.freecol.client.gui.ChoiceItem.java: Achieved 88.9% branch coverage.
2. Net.sf.freecol.client.gui/FontLibrary.java: Achieved 69.6.% branch coverage.
3. net.sf.freecol.client.gui/FrameMotionListener.java: Achieved 96.1% branch coverage.
4. net.sf.freecol.client.gui/TerrainCursor.java: Achieved 94.7% branch coverage.

The screenshot shows the Eclipse IDE interface with the following details:

- Editor View:** Displays Java code for a class named `Ability.java`. The code includes a switch statement based on `rightHandSide.getScopeLevel()`, handling cases for `SETTLEMENT`, `TAZER`, and `GAME`.
- Coverage View:** Located in the bottom right corner, this view provides a summary of code coverage across various packages and files. It includes columns for Element, Coverage, Covered Instructions, Missed Instructions, and Total Instructions.
- Table: Coverage Summary**

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
FreeCol	0.8 %	2,747	331,688	334,435
FreeColTest	8.0 %	1,420	16,321	17,741
net.sf.freecol.common.model	8.0 %	1,420	16,321	17,741
net.sf.freecol.common	0.0 %	0	15,665	15,665
net.sf.freecol.common	63.2 %	1,420	397	397
net.sf.freecol.common	69.6 %	348	152	500
net.sf.freecol.common	88.9 %	567	71	638
net.sf.freecol.common	96.1 %	292	12	304
net.sf.freecol.common	94.7 %	213	12	225
net.sf.freecol	0.0 %	0	12	12

5. net.sf.freecol.common/FreeColSeed.java: Achieved 63.0% branch coverage.
6. Net.sf.freecol.common/ServerInfo.java: Achieved 96.4% branch coverage.

Java - FreeCol/src/net/sf/freecol/common/model/Limit.java - Eclipse

File Edit Source Refactor Navigate Search Project CodePro Bad Smells Run Window Help

Quick Access | Java EE | Git | Java | Functional Test Specification | Functional Test Reporting

Package Explorer | Project Explorer | JUnit | UnitTest | Limit.java | LimitTest.java | Building.java | Map.java

```

380     Integer rhs = null;
381     switch (rightHandSide.getScopeLevel()) {
382     case SETTLEMENT:
383         rhs = rightHandSide.getValue(settlement);
384         break;
385     case PLAYER:
386         rhs = rightHandSide.getValue(settlement.getOwner());
387         break;
388     case GAME:
389         rhs = rightHandSide.getValue(settlement.getGame());
390         break;
391     default:
392         rhs = rightHandSide.getValue();
393         break;
394     }
395 }
396 
```

Problems | Javadoc | Declaration | Search | Console | Progress | Audit | Metrics | Generation Results | Coverage | Duplicated Code

net.sf.freecol.common (May 15, 2016 7:12:46 AM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
FreeCol	21 %	6,902	327,533	334,435
FreeColTest	71.9 %	12,739	4,990	17,729
git	71.9 %	12,739	4,990	17,729
net.sf.freecol.common.model	79.1 %	12,386	3,279	15,665
net.sf.freecol.common.guis	0.0 %	0	1,667	1,667
net.sf.freecol.common.mon	88.9 %	353	44	397
FreeColSeedTest.java	63.0 %	34	20	54
ServerInfoTest.java	96.4 %	319	12	331
TestAll.java	0.0 %	0	12	12

7:13 AM 5/15/2016

7. Net.sf.freecol.common.model/NationSummary.java: Achieved 49.1% code coverage.
8. Net.sf.freecol.common.model/Operand.java: Achieved 79.2% code coverage.
9. Net.sf.freecol.common.model/Limit.java: Achieved 85.1% code coverage.
10. Net.sf.freecol.common.model/IndianNationType.java: Achieved 85.9% code coverage.
11. Net.sf.freecol.common.model/Disaster.java: Achieved 87.7% code coverage.
12. Net.sf.freecol.common.model/NationOptions.java: Achieved 90.9% code coverage.
13. Net.sf.freecol.common.model/Nation.java: Achieved 93.3% code coverage.
14. Net.sf.freecol.common.model/NationType.java: Achieved 87.0% code coverage.
15. Net.sf.freecol.common.model/BuidableType.java: Achieved 92.9% code coverage.
16. Net.sf.freecol.common.model/AbstractUnit.java: Achieved 94.5% code coverage.
17. Net.sf.freecol.common.model/ResourceType.java: Achieved 87.47% code coverage.
18. Net.sf.freecol.common.model/AbstractGoods.java: Achieved 94.5% code coverage.
19. Net.sf.freecol.common.model/Stance.java: Achieved 98.2% code coverage.
20. Net.sf.freecol.common.model/Tension.java: Achieved 93.5% code coverage.

Java - FreeColTest/src/net/sf/freecol/common/model/TestAll.java - Eclipse

File Edit Source Refactor Navigate Search Project CodePro Bad Smells Run Window Help

Quick Access | Java EE | Git | Java | Functional Test Specification | Functional Test Reporting

Package Explorer | Project Explorer | JUnit | UnitTest | Limit.java | LimitTest.java | Building.java | Map.java | TestAll.java

```

11  public class TestAll {
12
13     @Test
14     public void testAll() {
15         NationTypeTest.class;
16         DisasterTest.class;
17         AbstractUnitTest.class;
18         AbstractGoodsTest.class;
19         ResourceTest.class;
20         ResourceTypeTest.class;
21         DisastersTest.class;
22         IndianNationTypeTest.java;
23         LimitTest.java;
24         NationOptionsTest.java;
25         OperandTest.java;
26         NationTest.java;
27         ResourceTypeTest.java;
28         StanceTest.java;
29         TensionTest.java;
30         TestAll.java;
31     }
32 }
33 
```

Problems | Javadoc | Declaration | Search | Console | Progress | Audit | Metrics | Generation Results | Coverage | Duplicated Code

net.sf.freecol.common (May 15, 2016 7:18:14 AM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
net.sf.freecol.common.model	87.0 %	12,345	1,846	14,191
StanceOptionsTest.java	49.1 %	441	458	899
OperandTest.java	79.2 %	1,370	360	1,730
LimitTest.java	85.1 %	1,169	204	1,373
AbstractUnitTest.java	85.9 %	941	155	1,096
DisasterTest.java	87.7 %	707	106	875
IndianNationTypeTest.java	90.9 %	1,062	106	1,168
NationOptionsTest.java	93.3 %	1,441	104	1,545
NationTypeTest.java	87.0 %	640	96	736
BuidableTypeTest.java	92.9 %	1,106	84	1,190
AbstractGoodsTest.java	84.4 %	1,042	121	1,163
ResourceTypeTest.java	87.7 %	284	40	324
AbstractGoodsTest.java	97.3 %	1,243	34	1,277
StanceTest.java	98.2 %	666	12	678
TensionTest.java	93.5 %	173	12	185
TestAll.java	0.0 %	0	12	12
net.sf.freecol.client.gui	0.0 %	0	1,667	1,667

7:19 AM 5/15/2016

Will:

Will's responsibility is to fix all the severe bugs that were found with the FindBug tool as well as all the high code violations found with the PMD tool for the classes picked. He is to also refactor method using JDeodorant, and Eclipse Refactoring tools where possible. He is to lower the cyclomatic complexity on the classes that it is possible for. He is also to generate all the JDocs documentation for all the classes.

The image shows two side-by-side screenshots of the Eclipse Metrics view. Both windows have a toolbar at the top with icons for Problems, Console, Long Method, Coverage, Generation Results, and Metrics. The left window is for 'BuildingDetailPanel.java at 5/13/16 3:02 AM' and the right window is for 'BaseCostDecider.java at 5/13/16 3:33 AM'. Each window has a 'Metrics' tab selected.

BuildingDetailPanel.java Metrics:

Metric	Value
+ Abstractness	0%
+ Average Block Depth	3.33
- Average Cyclomatic Complexity	9.00
net.sf.freecol.client.gui.panel	9.00
BuildingDetailPanel.java	9.00
+ Average Lines Of Code Per Method	56.33
+ Average Number of Constructors Per Type	1.00
+ Average Number of Fields Per Type	0.00
+ Average Number of Methods Per Type	2.00
+ Average Number of Parameters	1.50
+ Comments Ratio	6.8%
Efferent Couplings	0
+ Lines of Code	203
+ Number of Characters	11,144
+ Number of Comments	14
+ Number of Constructors	1
Number of Fields	0
+ Number of Lines	274
+ Number of Methods	2
Number of Packages	1
+ Number of Semicolons	108
+ Number of Types	1
+ Weighted Methods	27

BaseCostDecider.java Metrics:

Name	Value
minimum	1
maximum	26

Unit.java at 5/12/16 10:59 PM			
Metric	Value	Minimum and Maximum Method Complexities	
		Name	Value
+ Abstractness	0%	Unit()	1
+ Average Block Depth	1.25	Unit()	1
- Average Cyclomatic Complexity	3.19	Unit()	1
- net.sf.freecol.common.model	3.19	getName()	1
- Unit.java	3.19	setName()	1
	3.66	getApparentOwnerName()	1
	3.33	getLabel()	1
MoveType	1.33	getLabel()	14
Unit	3.23	getDescription()	1
UnitLabelType	0.0	getDescription()	1
UnitState	1.00	getCombatLabel()	2
+ Average Lines Of Code Per Method	8.90	getDestinationLabel()	1
+ Average Number of Constructors Per Type	0.83	getDestinationLabel()	1
+ Average Number of Fields Per Type	12.50	getRepairLabel()	1
+ Average Number of Methods Per Type	41.16	getType()	1
+ Average Number of Parameters	0.66	setType()	1
+ Comments Ratio	21.8%	changeType()	5
Efferent Couplings	0	isNaval()	2
+ Lines of Code	2,348	isUndead()	1
+ Number of Characters	159,332	canCarryTreasure()	1
+ Number of Comments	513	canCaptureGoods()	1
+ Number of Constructors	5	isTradingUnit()	2
+ Number of Fields	113	isColonist()	2
+ Number of Lines	4,653	isCarrier()	2
+ Number of Methods	247	isPerson()	2
Number of Packages	1	getState()	1
+ Number of Semicolons	921	checkSetState()	12
+ Number of Types	6	setState()	3
+ Weighted Methods	804	setStateUnchecked()	18
		setStateToAllChildren()	2
		changeOwner()	9
		getRole()	1
		setRole()	1
		getRoleCount()	1
		setRoleCount()	1
		hasDefaultRole()	1
		getRoleSuffix()	1
		changeRole()	3

Metric	Value	Minimum and Maximum	Method Complexities	Description
Name	Value			
+ Abstractness	0%			
+ Average Block Depth	2.60			
- Average Cyclomatic Complexity	12.00			
- net.sf.freecol.server.model	12.00			
ServerUnit.java	12.00			
+ Average Lines Of Code Per Method	51.93	csNewTurn()	42	
+ Average Number of Constructors Per Type	4.00	csImproveTile()	13	
+ Average Number of Fields Per Type	0.00	csEmbark()	6	
+ Average Number of Methods Per Type	11.00	csRepairUnit()	2	
+ Average Number of Parameters	1.72	getSlowedBy()	13	
+ Comments Ratio	13.8%	csNativeBurialGround()	1	
Efferent Couplings	0	csExploreLostCityRumour()	40	
+ Lines of Code	839	csActivateSentries()	2	
+ Number of Characters	46,889	collectNewTiles()	1	
+ Number of Comments	116	csMove()	51	
+ Number of Constructors	4	getServerXMLElementTagName()	1	
+ Number of Fields	1			
+ Number of Lines	1,090			
+ Number of Methods	11			
Number of Packages	1			
+ Number of Semicolons	422			
+ Number of Types	1			
+ Weighted Methods	180			

ServerColony.java at 5/12/16 10:08 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	2.25			
- Average Cyclomatic Complexity	7.12			
- net.sf.freecol.server.model	7.12			
ServerColony.java	7.12			
+ Average Lines Of Code Per Method	35.56			
+ Average Number of Constructors Per Type	2.00			
+ Average Number of Fields Per Type	0.00			
+ Average Number of Methods Per Type	14.00			
+ Average Number of Parameters	2.00			
+ Comments Ratio	16.8%			
Efferent Couplings	0			
+ Lines of Code	612			
+ Number of Characters	37,256			
+ Number of Comments	103			
+ Number of Constructors	2			
+ Number of Fields	1			
+ Number of Lines	876			
+ Number of Methods	14			
Number of Packages	1			
+ Number of Semicolons	283			
+ Number of Types	1			
+ Weighted Methods	114			

SelectOption.java at 5/12/16 11:13 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	1.11			
- Average Cyclomatic Complexity	1.26			
- net.sf.freecol.common.option	1.26			
SelectOption.java	1.26			
+ Average Lines Of Code Per Method	5.73			
+ Average Number of Constructors Per Type	1.00			
+ Average Number of Fields Per Type	2.00			
+ Average Number of Methods Per Type	14.00			
+ Average Number of Parameters	0.57			
+ Comments Ratio	25.9%			
Efferent Couplings	0			
+ Lines of Code	104			
+ Number of Characters	7,029			
+ Number of Comments	27			
+ Number of Constructors	1			
+ Number of Fields	5			
+ Number of Lines	248			
+ Number of Methods	14			
Number of Packages	1			
+ Number of Semicolons	51			
+ Number of Types	1			
+ Weighted Methods	19			

Problems | Console | Long Method | Coverage | Generation Results | Metrics

BuildingToolTip.java at 5/13/16 3:09 AM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	3.50			
- Average Cyclomatic Complexity	9.50			
- net.sf.freecol.client.gui.panel	9.50			
BuildingToolTip.java	9.50			
+ Average Lines Of Code Per Method	63.00			
+ Average Number of Constructors Per Type	1.00			
+ Average Number of Fields Per Type	1.00			
+ Average Number of Methods Per Type	1.00			
+ Average Number of Parameters	0.00			
+ Comments Ratio	3.9%			
Efferent Couplings	0			
+ Lines of Code	152			
+ Number of Characters	8,174			
+ Number of Comments	6			
+ Number of Constructors	1			
+ Number of Fields	1			
+ Number of Lines	209			
+ Number of Methods	1			
Number of Packages	1			
+ Number of Semicolons	86			
+ Number of Types	1			
+ Weighted Methods	19			

Problems Console Long Method Coverage Generation Results Metrics

REFAIPlayer.java at 5/13/16 3:46 AM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	2.18			
- Average Cyclomatic Complexity	10.00			
- net.sf.freecol.server.ai	10.00			
- REFAIPlayer.java	10.00			
REFAIPlayer	15.66			
TargetTuple	2.50			
+ Average Lines Of Code Per Method	40.56			
+ Average Number of Constructors Per Type	1.00			
+ Average Number of Fields Per Type	2.00			
+ Average Number of Methods Per Type	4.33			
+ Average Number of Parameters	1.38			
+ Comments Ratio	14.4%			
Efferent Couplings	0			
+ Lines of Code	685			
+ Number of Characters	37,562			
+ Number of Comments	99			
+ Number of Constructors	3			
+ Number of Fields	9			
+ Number of Lines	920			
+ Number of Methods	13			
Number of Packages	1			
+ Number of Semicolons	364			
+ Number of Types	3			
+ Weighted Methods	160			

Problems Console Long Method Coverage Generation Results Metrics

RangeOption.java at 5/12/16 11:11 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	1.33			
- Average Cyclomatic Complexity	1.50			
- net.sf.freecol.common.option	1.50			
RangeOption.java	1.50			
+ Average Lines Of Code Per Method	5.00			
+ Average Number of Constructors Per Type	1.00			
+ Average Number of Fields Per Type	0.00			
+ Average Number of Methods Per Type	5.00			
+ Average Number of Parameters	0.20			
+ Comments Ratio	26.3%			
Efferent Couplings	0			
+ Lines of Code	38			
+ Number of Characters	3,360			
+ Number of Comments	10			
+ Number of Constructors	1			
+ Number of Fields	1			
+ Number of Lines	118			
+ Number of Methods	5			
Number of Packages	1			
+ Number of Semicolons	19			
+ Number of Types	1			
+ Weighted Methods	9			

Problems Console Long Method Coverage Generation Results Metrics

MetaServer.java at 5/12/16 11:08 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	1.00			
- Average Cyclomatic Complexity	1.70			
- net.sf.freecol.metaserver	1.70			
- MetaServer.java	1.70			
	2.00			
MetaServer	1.66			
+ Average Lines Of Code Per Method	8.40			
+ Average Number of Constructors Per Type	0.50			
+ Average Number of Fields Per Type	2.50			
+ Average Number of Methods Per Type	4.50			
+ Average Number of Parameters	0.33			
+ Comments Ratio	15.3%			
Efferent Couplings	0			
+ Lines of Code	98			
+ Number of Characters	6,457			
+ Number of Comments	15			
+ Number of Constructors	1			
+ Number of Fields	8			
+ Number of Lines	209			
+ Number of Methods	9			
Number of Packages	1			
+ Number of Semicolons	54			
+ Number of Types	2			
+ Weighted Methods	17			

Problems Console Long Method Coverage Generation Results Metrics

InGameController.java at 5/12/16 11:05 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	2.05			
- Average Cyclomatic Complexity	6.43			
- net.sf.freecol.client.control	6.43			
- InGameController.java	6.43			
InGameController	6.49			
MoveMode	2.00			
+ Average Lines Of Code Per Method	21.11			
+ Average Number of Constructors Per Type	0.50			
+ Average Number of Fields Per Type	3.00			
+ Average Number of Methods Per Type	72.00			
+ Average Number of Parameters	1.75			
+ Comments Ratio	14.3%			
Efferent Couplings	0			
+ Lines of Code	3,166			
+ Number of Characters	192,070			
+ Number of Comments	454			
+ Number of Constructors	1			
+ Number of Fields	10			
+ Number of Lines	5,033			
+ Number of Methods	144			
Number of Packages	1			
+ Number of Semicolons	1,519			
+ Number of Types	2			
+ Weighted Methods	933			
		Name	Value	
		continueIgnoreMessage()	4	
		displayTurnReportMessages()	1	
		displayModelMessages()	6	
		doExecuteGotoOrders()	11	
		doEndTurn()	8	
		updateActiveUnit()	9	
		moveToDestination()	12	
		moveDirection()	57	
		movePath()	13	
		moveAttack()	5	
		moveAttackSettlement()	12	
		moveDiplomacy()	6	
		moveDisembark()	8	
		moveEmbark()	8	
		moveExplore()	5	
		moveHighSeas()	10	
		moveLearnSkill()	8	
		moveMove()	16	
		moveScoutColony()	8	
		moveScoutIndianSettlement()	8	
		moveSpy()	2	
		moveTrade()	5	
		moveTradeIndianSettlement()	14	
		tradeFailMessage()	6	
		attemptBuyFromSettlement()	12	
		attemptSellToSettlement()	11	
		attemptGiftToSettlement()	3	
		moveTribute()	2	
		moveUseMissionary()	9	
		followTradeRoute()	20	
		loadUnitAtStop()	35	
		unloadUnitAtStop()	19	
		getUnloadGoodsMessage()	5	
		abandonColony()	6	
		animateAttack()	2	
		animateMove()	3	
		assignTeacher()	13	
		assignTradeRoute()	3	

Problems Console Long Method Coverage Generation Results Metrics

Limit.java at 5/12/16 11:04 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	0.92			
- Average Cyclomatic Complexity	2.40			
- net.sf.freecol.common.model	2.40			
- Limit.java	2.40			
Limit	2.40			
Operator	0.0			
+ Average Lines Of Code Per Method	8.00			
+ Average Number of Constructors Per Type	1.50			
+ Average Number of Fields Per Type	4.00			
+ Average Number of Methods Per Type	9.50			
+ Average Number of Parameters	0.73			
+ Comments Ratio	17.5%			
Efferent Couplings	0			
+ Lines of Code	194			
+ Number of Characters	10,922			
+ Number of Comments	34			
+ Number of Constructors	3			
+ Number of Fields	13			
+ Number of Lines	386			
+ Number of Methods	19			
Number of Packages	1			
+ Number of Semicolons	98			
+ Number of Types	2			
+ Weighted Methods	53			

Problems Console Long Method Coverage Generation Results Metrics

FreeColDirectories.java at 5/12/16 11:15 PM

Metric	Value	Minimum and Maximum	Method Complexities	Description
+ Abstractness	0%			
+ Average Block Depth	1.50			
- Average Cyclomatic Complexity	3.43			
- net.sf.freecol.common.io	3.43			
FreeColDirectories.java	3.43			
+ Average Lines Of Code Per Method	8.56			
+ Average Number of Constructors Per Type	0.00			
+ Average Number of Fields Per Type	10.00			
+ Average Number of Methods Per Type	44.00			
+ Average Number of Parameters	0.40			
+ Comments Ratio	18.2%			
Efferent Couplings	0			
+ Lines of Code	427			
+ Number of Characters	30,140			
+ Number of Comments	78			
Number of Constructors	0			
+ Number of Fields	37			
+ Number of Lines	929			
+ Number of Methods	44			
Number of Packages	1			
+ Number of Semicolons	229			
+ Number of Types	1			
+ Weighted Methods	151			
		Name	Value	
		onMacOSX()	1	
		onUnix()	1	
		onWindows()	1	
		getUserDefaultDirectory()	1	
		checkDir()	6	
		getXDGDirs()	18	
		isGoodDirectory()	2	
		requireDir()	5	
		getMacOSXDirs()	18	
		getWindowsDirs()	5	
		getOldUserDirectory()	9	
		copyIfFound()	4	
		insistDirectory()	4	
		deriveAutosaveDirectory()	4	
		setDataDirectory()	6	
		setUserDirectories()	22	
		getAutosaveDirectory()	1	
		getBaseDirectory()	1	
		getBaseClientOptionsFile()	1	
		getClientOptionsFile()	2	
		setClientOptionsFile()	3	
		getDataDirectory()	1	
		getHighScoreFile()	1	
		getI18nDirectory()	1	
		getLogFilePath()	1	
		setLogFilePath()	1	
		getMapsDirectory()	1	
		getOptionsDirectory()	2	
		getOptionsFile()	2	
		getRulesClassicDirectory()	1	
		getRulesDirectory()	1	
		getSaveDirectory()	1	
		getSavegameFile()	1	
		setSavegameFile()	6	
		getLastSaveGameFile()	4	
		getStandardModsDirectory()	1	
		getStartMapFile()	1	
		getUserCacheDirectory()	1	

DragListener.java at 5/13/16 3:31 AM		
Metric	Value	Minimum and Maximum Method Complexities Description
+ Abstractness	0%	
+ Average Block Depth	1.25	
- Average Cyclomatic Complexity	9.50	
- net.sf.freecol.client.gui.panel	9.50	
DragListener.java	9.50	
+ Average Lines Of Code Per Method	29.50	
+ Average Number of Constructors Per Type	1.00	
+ Average Number of Fields Per Type	2.00	
+ Average Number of Methods Per Type	1.00	
+ Average Number of Parameters	1.00	
+ Comments Ratio	17.3%	
Efferent Couplings	0	
+ Lines of Code	75	
+ Number of Characters	5,203	
+ Number of Comments	13	
+ Number of Constructors	1	
+ Number of Fields	3	
+ Number of Lines	137	
+ Number of Methods	1	
Number of Packages	1	
+ Number of Semicolons	39	
+ Number of Types	1	
+ Weighted Methods	19	

Jose:

Jose is to assist with refactoring methods using JDeodorant, and Eclipse Refactoring tools, create JUnit test cases, fix style issues found with CheckStyle

The screenshot shows the Eclipse IDE interface with the following details:

- Java - FreeColTest/src/net/sf/freecol/FreeColTest.java - Eclipse**: The active Java file.
- File Edit Source Refactor Navigate Search Project CodePro Bad Smells Run Window Help**: The menu bar.
- Quick Access**: A search bar.
- Package Explorer**: Shows 14/61 runs, 1 error, and 0 failures.
- JUnit**: A view showing test results: testBadSave_1 (0.177 s), testSetGUIScale_1 (0.001 s), testSetGUIScale_2 (0.000 s), testSetGUIScale_3 (0.000 s), testSetGUIScale_4 (0.000 s), testSetGUIScale_5 (0.000 s), testSetGUIScale_6 (0.000 s), testLoadSpecification_1 (0.000 s), testLoadSpecification_2 (0.186 s), testSetTimeout_3 (0.000 s), testSetTimeout_2 (0.001 s), testSetTimeout_3 (0.000 s), testGetValidDifficulties_1 (0.046 s), and testFatal_1.
- Editor**: Displays Java code for `FreeColTest`, specifically the `testLoadSpecification_2` method.
- Failure Trace**: Shows a stack trace for a `java.io.IOException: File does not exist` exception.
- Console**: Displays the command-line output of the test run, including the Java command and the stack trace of the exception.

Java - FreeColTest/src/net/sf/freecol/FreeColTest.java - Eclipse

```

File Edit Source Refactor Navigate Search Project CodePro Bad Smells Run Window Help
Quick Access Java Git Debug
Package Explorer Type Hierarchy JUnit
Terminated
Runs: 14/61 Errors: 0 Failures: 0
net.sf.freecol.FreeColTest [Runner: JUnit 4]
  testBadSave_1 (0.142 s)
  testSetGUIScale_1 (0.001 s)
  testSetGUIScale_2 (0.001 s)
  testSetGUIScale_3 (0.001 s)
  testSetGUIScale_4 (0.000 s)
  testSetGUIScale_5 (0.001 s)
  testSetGUIScale_6 (0.001 s)
  testLoadSpecification_1 (0.001 s)
  testLoadSpecification_2 (0.174 s)
  testSetTimeout_1 (0.000 s)
  testSetTimeout_2 (0.000 s)
  testSetTimeout_3 (0.000 s)
  testGetValidDifficulties_1 (0.057 s)
Failure Trace

```

FreeCol.tex *FreeCol.java FreeColTest... FreeColModF... FreeColDataF... FreeColTcFil... FreeColSetName(String)

```

58 */
59 @Test
60 public void testBadSave_1()
61     throws Exception {
62     File file = new File("");
63
64     StringTemplate result = FreeCol.badSave(file);
65
66     // add additional test code here
67     assertNotNull(result);
68     assertEquals("TEMPLATE: error.couldNotSave [%name%: NAME:",
69                 result.isEmpty());
70     assertEquals(false, result.getDefaultId());
71     assertEquals("null", result.getXMLTagName());
72     assertEquals("error.couldNotSave", result.getId());
73     assertEquals("couldNotSave", result.getSuffixId());
74     assertEquals("stringTemplate id=\"error.couldNotSave\" te",
75                 result.getSpecification());
76     assertEquals(null, result.getGame());
77     assertEquals("error.couldNotSave", result.getIdType());
78     assertEquals(-1, result.getIdNumber());
79     assertEquals(null, result.getFeatureContainer());
80 }
81
82 /**
83  * Run the void fatal(String) method test.
84  *
85  * @throws Exception
86 */

```

Pro... Java... Dec... Sea... Co... Pro... Mo... Pro... Lo... PIT... Cov... Met... J... Gen... Refactoring ... Source Method Variable Criterion Block-Based... Duplicated/... Rate it!

I had questions where it asked me to fix a null value but I left it there because it was part of a variable assignment.

No bugs were found with FindBugs

Java - FreeColTest/src/net/sf/freecol/client/FreeColClientTest.java - Eclipse

```

File Edit Source Refactor Navigate Search Project CodePro Bad Smells Run Window Help
Quick Access Java Git Debug
Package Explorer Type Hierarchy JUnit
FreeCol.tex developer.tex ClientOption... ClientOption... FreeColClient... FreeColClient... FreeColClient.fatal(String)
  package net.sf.freecol.client;
  ...
  #import java.awt.Dimension
  ...
  * The class <code>FreeColClientTest</code> contains
  * @generatedBy CodePro at 5/13/16 4:40 PM
  * @author JQ
  * @version $Revision: 1.0 $
  * @generatedBy CodePro at 5/13/16 4:40 PM
  * Run the FreeColClient(InputStream, String) constructor
  * @throws Exception
  * @generatedBy CodePro at 5/13/16 4:40 PM
  */
  @Test
  public void testFreeColClient_1()
      throws Exception {
      InputStream splashStream = new PipedInputStream();
      String fontName = "";
      FreeColClient result = new FreeColClient(sp
      ...

```

Finding bugs in FreeCol...

Prescanning... (found 0, checking java.lang.Class)

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Displays the `FreeCol` project structure under `src`, including packages like `net.sf.freecol` and `net.sf.freecol.client`, and files such as `FreeCol.java`, `InGameController.java`, and `MoveTest.java`.
- Code Editor:** Shows the `InGameController.java` file with PMD annotations. Annotations include:
 - Line 109: `109 */**`
 - Line 110: `110 * The controller that will be used while the game is played.`
 - Line 111: `111 */`
 - Line 112: `112 * Selecting next unit depends on mode--- either from the active list,`
 - Line 113: `113 * from the going-to list, or flush going-to and end the turn.`
 - Line 114: `114 */`
 - Line 115: `115 public final class InGameController extends FreeColClientHolder {`
 - Line 116: `116 private static final Logger logger = Logger.getLogger(InGameController.class.getName());`
 - Line 117: `117 /**`
 - Line 118: `118 * Selecting next unit depends on mode--- either from the active list,`
 - Line 119: `119 * from the going-to list, or flush going-to and end the turn.`
 - Line 120: `120 */`
 - Line 121: `121 *`
 - Line 122: `122 private static enum MoveMode {`
 - Line 123: `123 NEXT_ACTIVE_UNIT,`
 - Line 124: `124 EXECUTE_GOTO_ORDERS,`
 - Line 125: `125 END_TURN;`
 - Line 126: `126 }`
 - Line 127: `127 public MoveMode minimize(MoveMode m) {`
 - Line 128: `128 return (this.ordinal() > m.ordinal()) ? m : this;`
- Violations Outline:** A table showing violations across the project, with columns for P, Line, created, Rule, and Error.
- Violations Overview:** A detailed table showing violations per element, with columns for Element, # Violations, # Violations/KLOC, # Violations/Method, and Project.

P	Line	created	Rule	Error

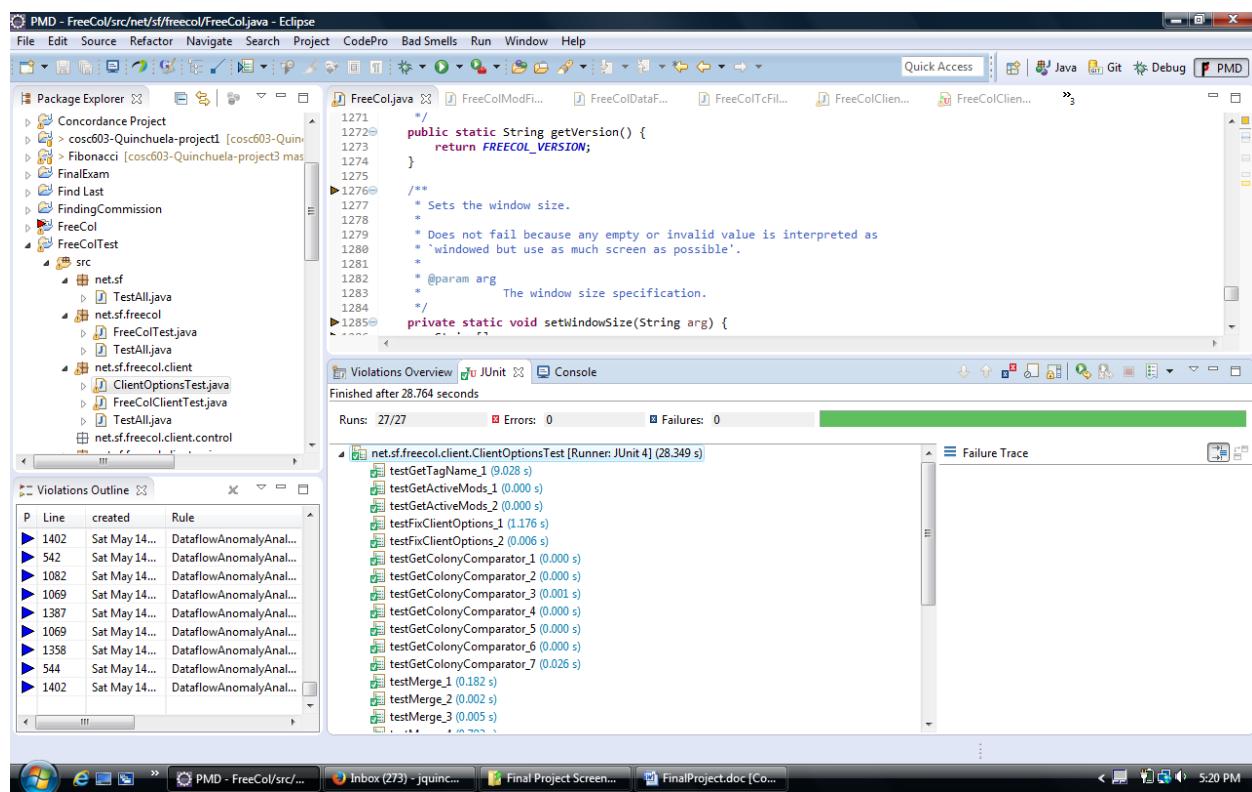
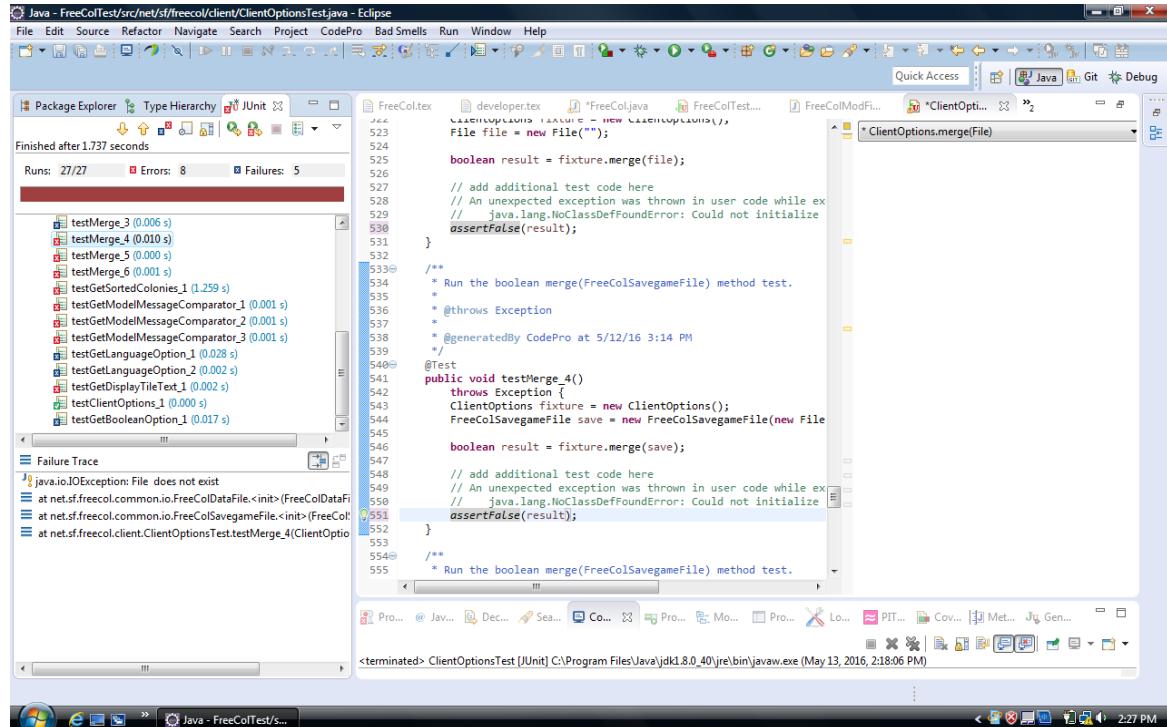
Element	# Violations	# Violations/KLOC	# Violations/Method	Project
FreeCol.java	816	1369.1	16.00	FreeCol
MethodArgumentCouldBeFinal	39	65.4	0.76	FreeCol
NullAssignment	5	8.4	0.10	FreeCol
BeanMembersShouldSerialize	3	5.0	0.06	FreeCol
OnlyOneReturn	7	11.7	0.14	FreeCol
FieldDeclarationsShouldBeAtStartOfClass	9	15.1	0.18	FreeCol
DoNotCallSystemExit	4	6.7	0.08	FreeCol
ExcessiveClassLength	1	1.7	0.02	FreeCol
StdCyclomaticComplexity	4	6.7	0.08	FreeCol
DoNotUseThreads	2	3.4	0.04	FreeCol

Checklist used. I used Eclipse Formatter tool to clean up most of the warnings.

The screenshot shows the Eclipse IDE interface with the title "Java - FreeCol/src/net/sf/freecol/FreeColJava - Eclipse". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, CodePro, Bad Smells, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The left sidebar shows the "Package Explorer" with the project structure: FreeCol (src, net.sf.freecol, FreeColJava, FreeCol, LogLevel, advantages, ADVANTAGES_DEFAULT, checkIntegrity, CLIENT_THREAD, consoleLogging, debugStart, DIFFICULTIES, difficulty, DIFFICULTY_DEFAULT, europeansCount, EUROPEANS_DEFAULT, europeansMin, fastStart, fontName, FREECOL_SAVE_EXTENSION, FREECOL_VERSION, freeColRevision, GUI_SCALE_DEFAULT, GUI_SCALE_MAX, GUI_SCALE_MAX_PCT, GUI_SCALE_MIN, GUI_SCALE_MIN_PCT, GUI_SCALE_STEP, GUI_SCALE_STEP_PCT, guiScale, headless). The right panel displays the code editor for "FreeCol.java" with the following content:

```
148     /** The Constant SPLASH_DEFAULT. */
149     private static final String SPLASH_DEFAULT = "splash.jpg";
150
151     /** The Constant TC_DEFAULT. */
152     private static final String TC_DEFAULT = "freecol";
153
154     /** The Constant TIMEOUT_DEFAULT. */
155     public static final int TIMEOUT_DEFAULT = 60; // 1 minute
156
157     /** The Constant TIMEOUT_MIN. */
158     public static final int TIMEOUT_MIN = 10; // 10s
159
160     /** The Constant GUI_SCALE_MIN_PCT. */
161     private static final int GUI_SCALE_MIN_PCT = 100;
162
163     /** The Constant GUI_SCALE_MAX_PCT. */
164     private static final int GUI_SCALE_MAX_PCT = 200;
165
166     /** The Constant GUI_SCALE_STEP_PCT. */
167     private static final int GUI_SCALE_STEP_PCT = 25;
168
169     /** The Constant GUI_SCALE_MIN. */
170     public static final float GUI_SCALE_MIN = GUI_SCALE_MIN_PCT / 100.0f;
171
172     /** The Constant GUI_SCALE_MAX. */
173     public static final float GUI_SCALE_MAX = GUI_SCALE_MAX_PCT / 100.0f;
174
175     /** The Constant GUI_SCALE_STEP. */
176     public static final float GUI_SCALE_STEP = GUI_SCALE_STEP_PCT / 100.0f;
177
178     /** The Constant GUI_SCALE_DEFAULT. */
179     public static final float GUI_SCALE_DEFAULT = 1.0f;
```

More test cases:



Checklist used. I used Eclipse Formatter tool to clean up most of the warnings.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java - FreeCol/src/net/sf/freecol/client/ClientOptions.java - Eclipse
- Menu Bar:** File Edit Source Refactor Navigate Search Project CodePro Bad Smells Run Window Help
- Toolbars:** Standard, Java, Git, Debug
- Quick Access:** Java, Git, Debug
- Package Explorer:** Shows the project structure under FreeCol, including src, net.sf.freecol, net.sf.freecol.client, and net.sf.freecol.common packages.
- Code Editor:** Displays the Java code for ClientOptions.java, specifically handling static final String constants related to borders, move delays, pixmaps, panel positions, and sizes.
- Bottom Status Bar:** Writable, Smart Insert, 87:7
- Bottom Taskbar:** Java - FreeCol/src/n...

```
173     /** Whether to display borders by default or not. */
174     public static final String DISPLAY_BORDERS
175         = "model.option.displayBorders";
176
177     /** Whether to delay on a unit's last move or not. */
178     public static final String UNIT_LAST_MOVE_DELAY
179         = "model.option.unitLastMoveDelay";
180
181     /** Pixmap setting to work around Java 2D graphics bug. */
182     public static final String USE_PIXMAPS
183         = "model.option.usePixmaps";
184
185     /** Whether to remember the positions of various dialogs and panels. */
186     public static final String REMEMBER_PANEL_POSITIONS
187         = "model.option.rememberPanelPositions";
188
189     /** Whether to remember the sizes of various dialogs and panels. */
190     public static final String REMEMBER_PANEL_SIZES
191         = "model.option.rememberPanelSizes";
192
193     /** Whether to enable smooth rendering of the minimap when zoomed out. */
194     public static final String SMOOTH_MINIMAP_RENDERING
195         = "model.option.smoothRendering";
196
197     /** Whether to display end turn grey background or not. */
198     public static final String DISABLE_GRAY_LAYER
199         = "model.option.disableGrayLayer";
200
201     /** Whether to draw the fog of war on the minimap. */
202     public static final String MINIMAP_TOGGLE_FOG_OF_WAR
203         = "model.option.toggleFogOfWar";
```

Identify the groups responsible for the environmental needs above.

Everyone on the team would be responsible for their own environmental needs above cited.

Software Project Schedule

