FREECOL TACTITAL TEST PLAN

William Cosulich

Lauren Medley

Erica Muschelli

Jose Quinchuela

COSC 603

Table of Contents

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	ChoiceItem
17	net.sf.freecol.client.gui	FrameMontionListener
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: FrameMontionListener
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, Eclmma

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: ChoiceItem
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: FrameMontionListener
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

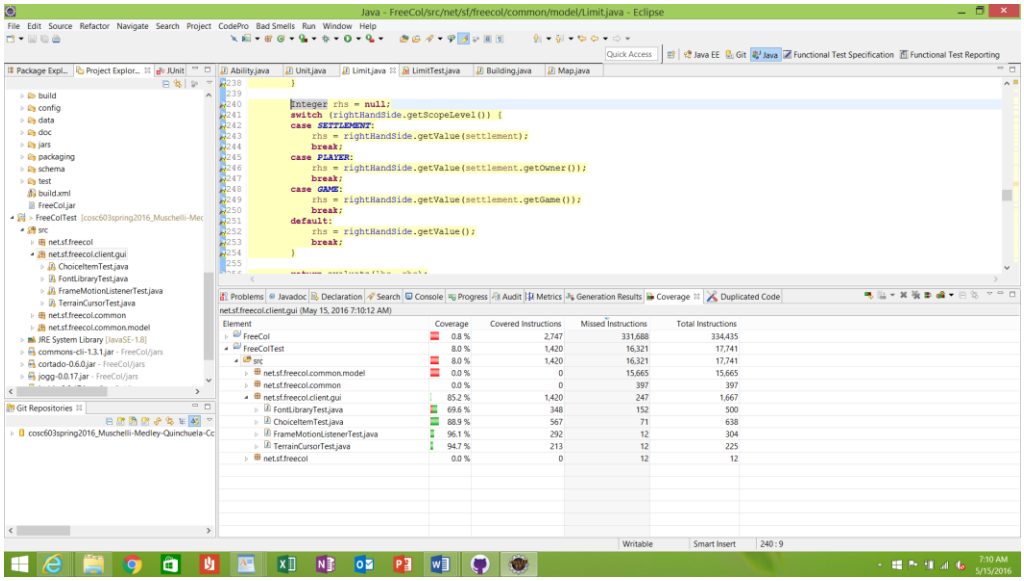
Erica: Setting up the repository and uploading the project on the repository for all to use.

Lauren/Erica:

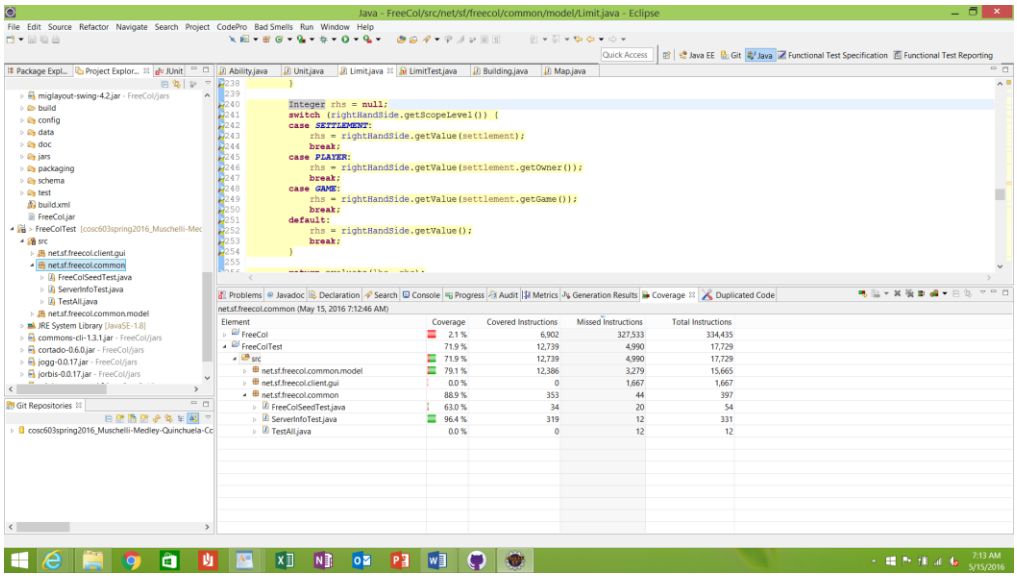
Erica with assistance from Lauren work on the majority of the JUnit test cases that were created for the project. They also worked to increase the coverage for a number of classes. They selected files where coverage was 0.0% to show code coverage improvement when developing our test cases. Some GUI files 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:

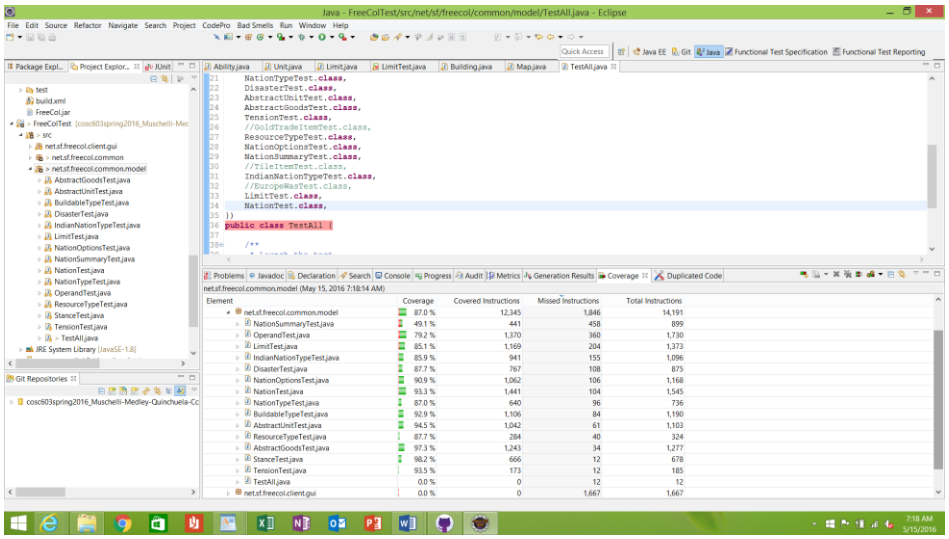
1. Net.sf.freecol.client.gui/ChoiceItem.java: Achieved 88.9% branch coverage.
2. Net.sf.freecol.client.gui/FontLibrary.java: Achieved 69.6.% branch coverage.
3. net.sf.freecol.client.gui/FrameMotionListener.java: Achieved 96.1% branch coverage.
4. net.sf.freecol.client.gui/TerrainCursor.java: Achieved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



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

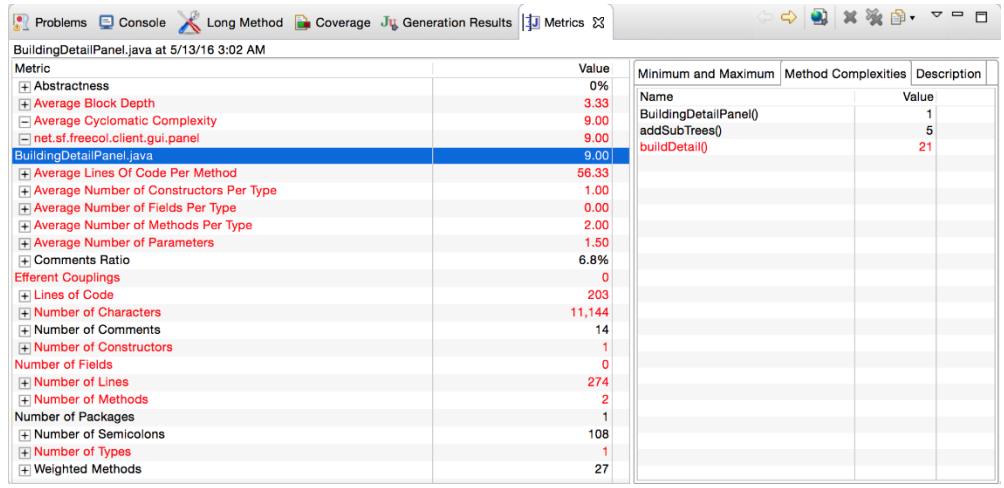


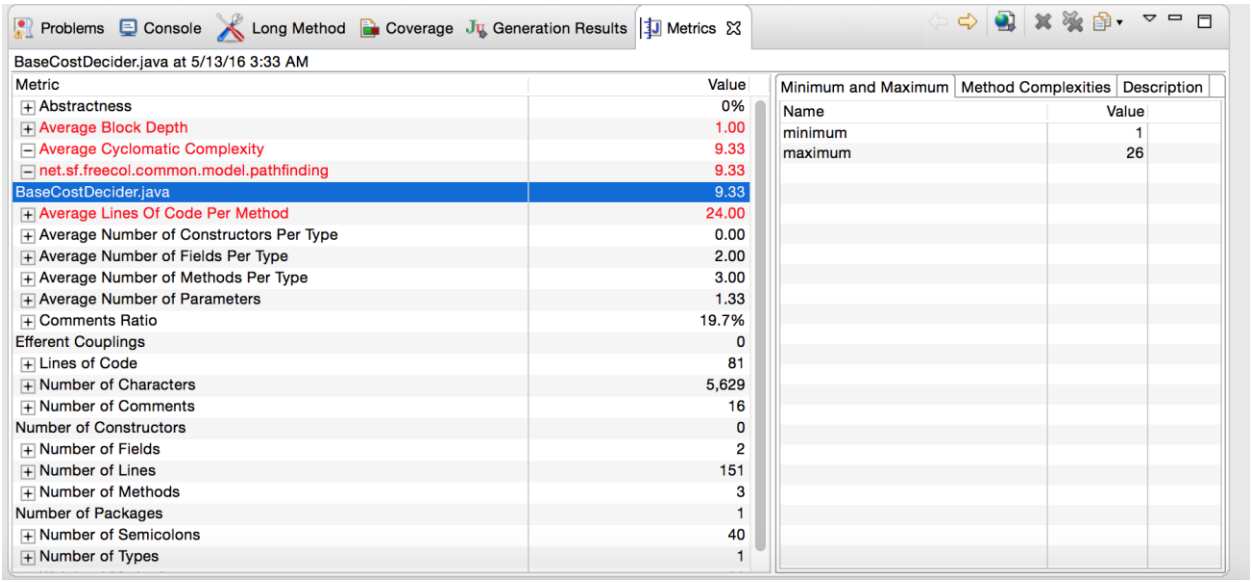
7. Net.sf.freecol.common.model/NationSummary.java: Achieved 49.1% code coverage.
8. Net.sf.freecol.common.model/Operand.java: Achieved 79.2% code coverage.
9. Net.sf.freecol.common.model/Limit.java: Achieved 85.1% code coverage.
10. Net.sf.freecol.common.model/IndianNationType.java: Achieved 85.9% code covereage.
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.

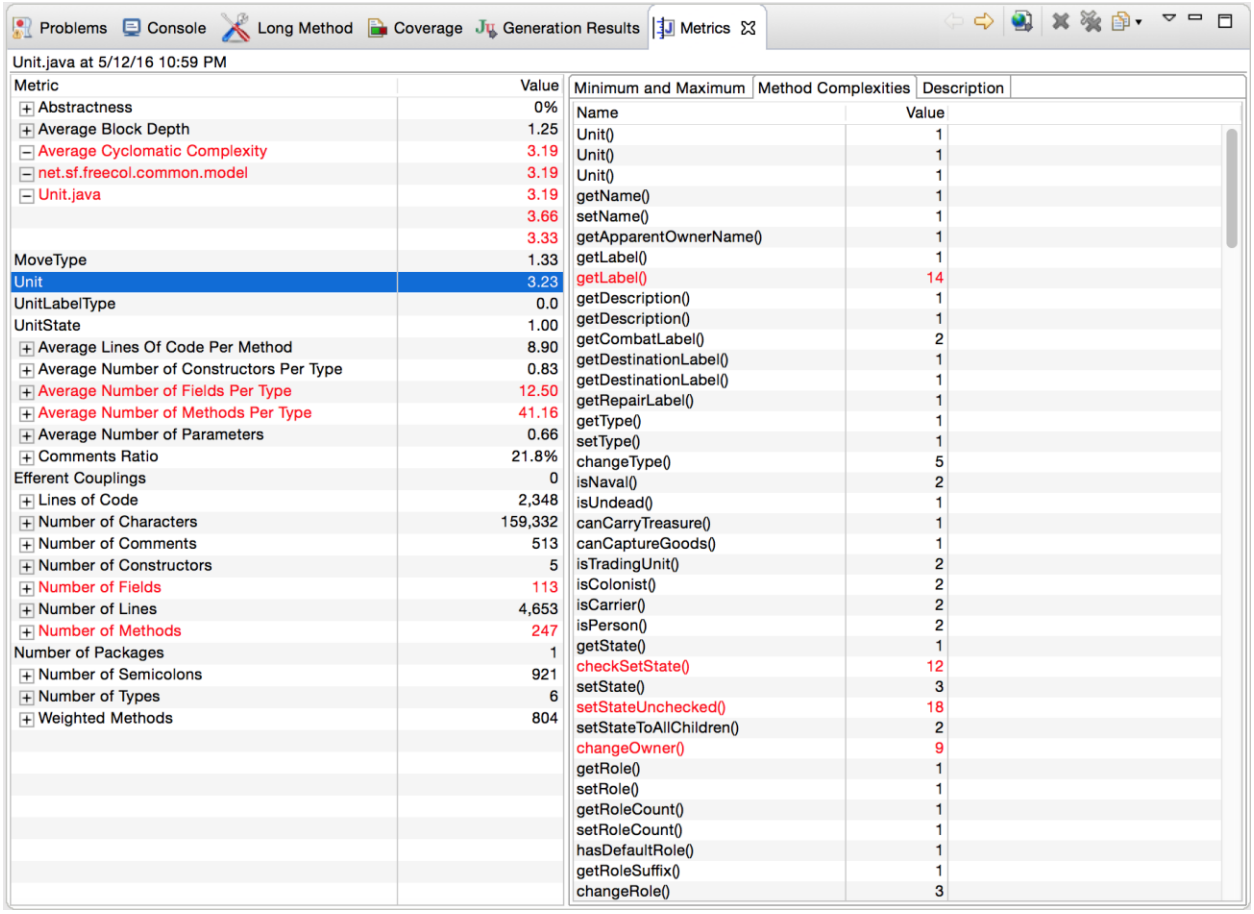


Will:

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.







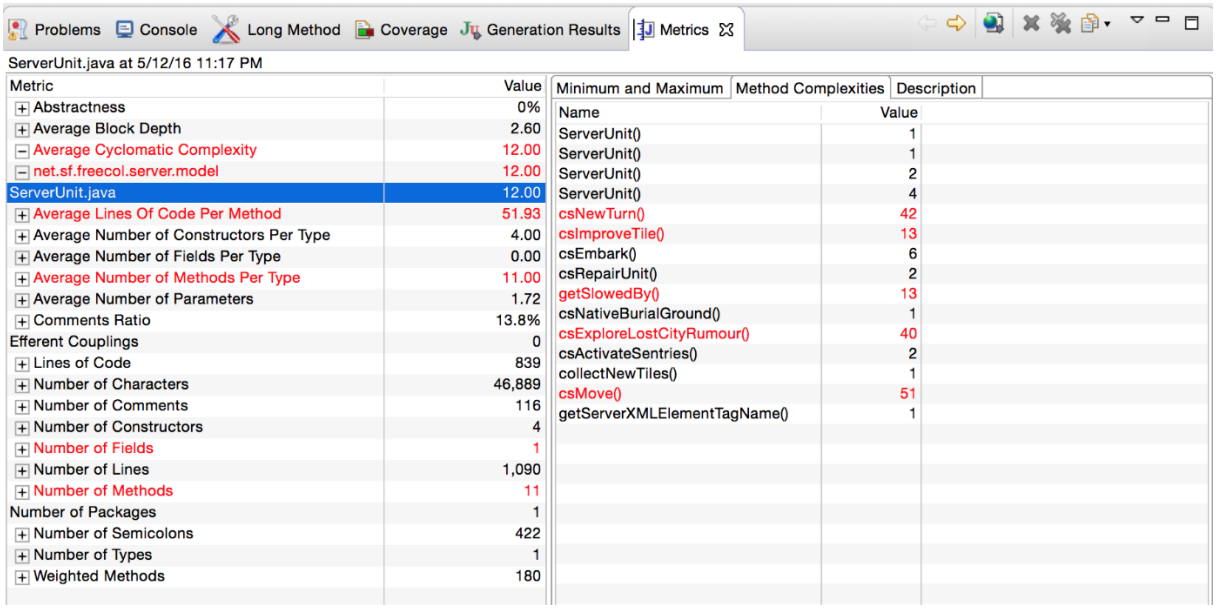
Problems Console Long Method Coverage Generation Results Metrics

Unit.java at 5/12/16 10:59 PM

Metric	Value
Abstractness	0%
Average Block Depth	1.25
Average Cyclomatic Complexity	3.19
net.sf.freecol.common.model	3.19
Unit.java	3.19
	3.66
	3.33
MoveType	1.33
Unit	3.23
UnitLabelType	0.0
UnitState	1.00
Average Lines Of Code Per Method	8.90
Average Number of Constructors Per Type	0.83
Average Number of Fields Per Type	12.50
Average Number of Methods Per Type	41.16
Average Number of Parameters	0.66
Comments Ratio	21.8%
Efferent Couplings	0
Lines of Code	2,348
Number of Characters	159,332
Number of Comments	513
Number of Constructors	5
Number of Fields	113
Number of Lines	4,653
Number of Methods	247
Number of Packages	1
Number of Semicolons	921
Number of Types	6
Weighted Methods	804

Minimum and Maximum | Method Complexities | Description

Name	Value
Unit()	1
Unit()	1
Unit()	1
getName()	1
setName()	1
getApparentOwnerName()	1
getLabel()	1
getLabel()	14
getDescription()	1
getDescription()	1
getCombatLabel()	2
getDestinationLabel()	1
getDestinationLabel()	1
getRepairLabel()	1
getType()	1
setType()	1
changeType()	5
isNaval()	2
isUndead()	1
canCarryTreasure()	1
canCaptureGoods()	1
isTradingUnit()	2
isColonist()	2
isCarrier()	2
isPerson()	2
getState()	1
checkSetState()	12
setState()	3
setStateUnchecked()	18
setStateToAllChildren()	2
changeOwner()	9
getRole()	1
setRole()	1
getRoleCount()	1
setRoleCount()	1
hasDefaultRole()	1
getRoleSuffix()	1
changeRole()	3



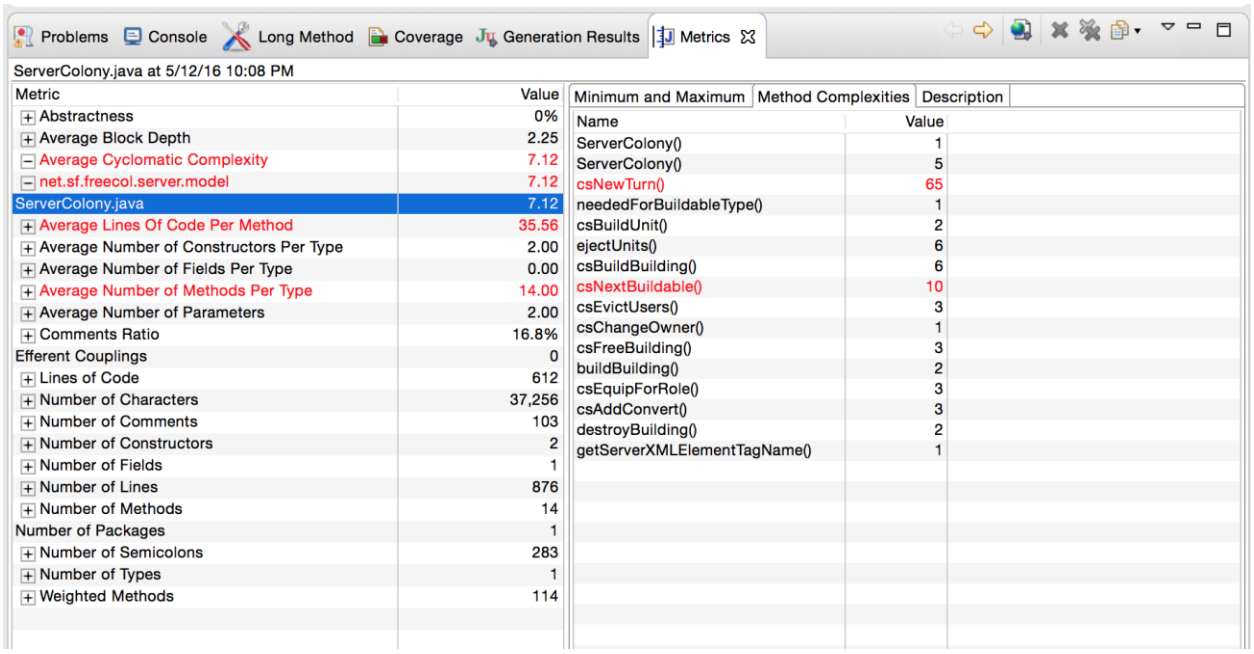
Problems Console Long Method Coverage Generation Results Metrics

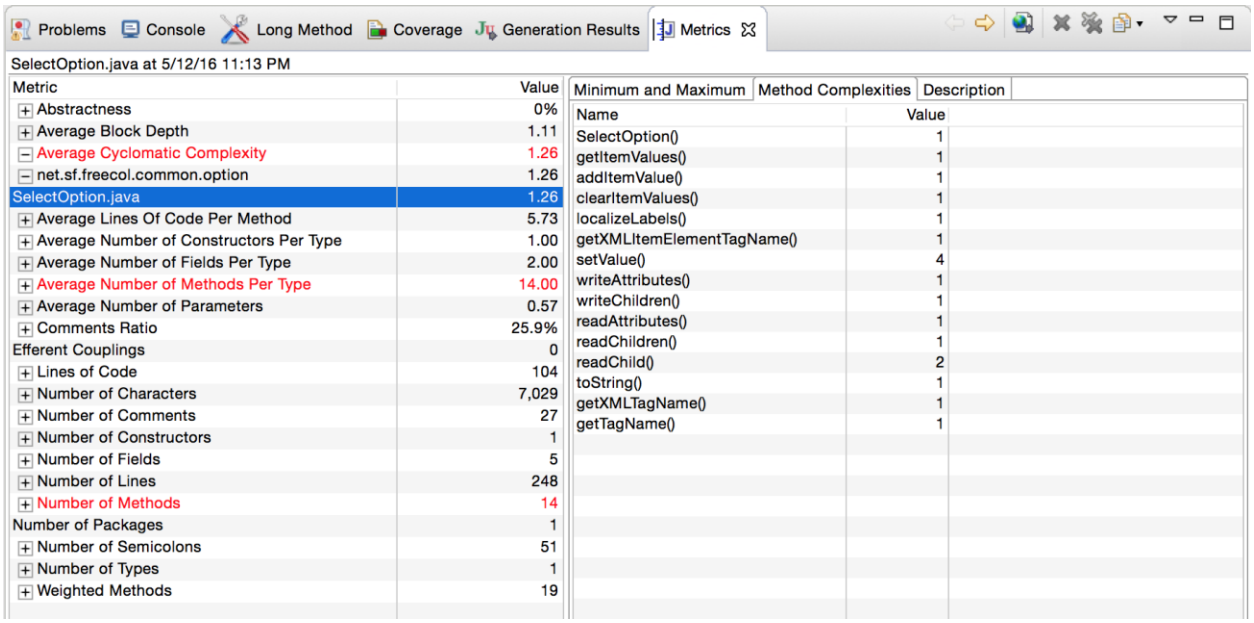
ServerUnit.java at 5/12/16 11:17 PM

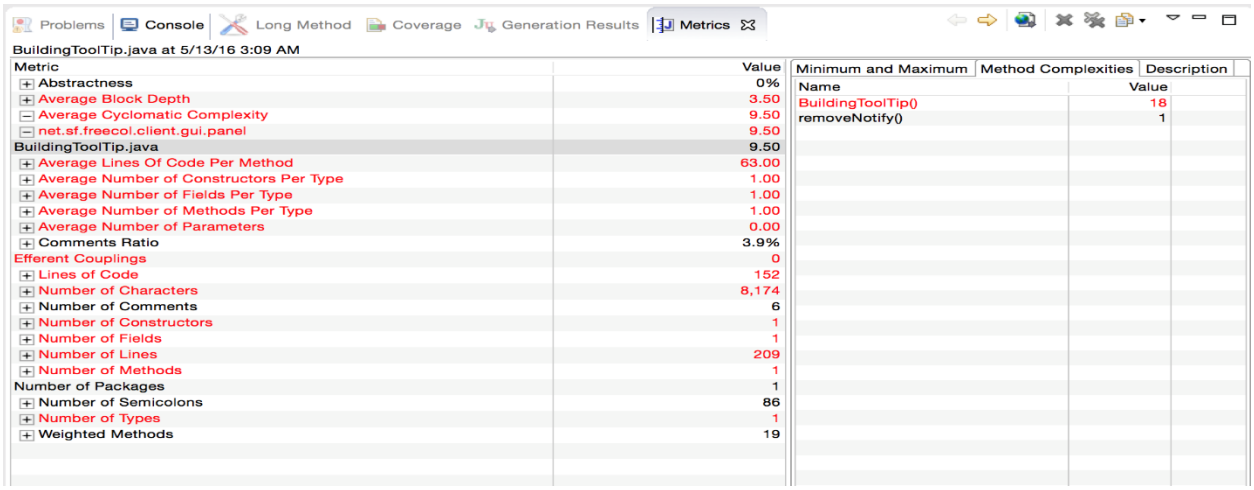
Metric	Value
Abstractness	0%
Average Block Depth	2.60
Average Cyclomatic Complexity	12.00
net.sf.freecol.server.model	12.00
ServerUnit.java	12.00
Average Lines Of Code Per Method	51.93
Average Number of Constructors Per Type	4.00
Average Number of Fields Per Type	0.00
Average Number of Methods Per Type	11.00
Average Number of Parameters	1.72
Comments Ratio	13.8%
Efferent Couplings	0
Lines of Code	839
Number of Characters	46,889
Number of Comments	116
Number of Constructors	4
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

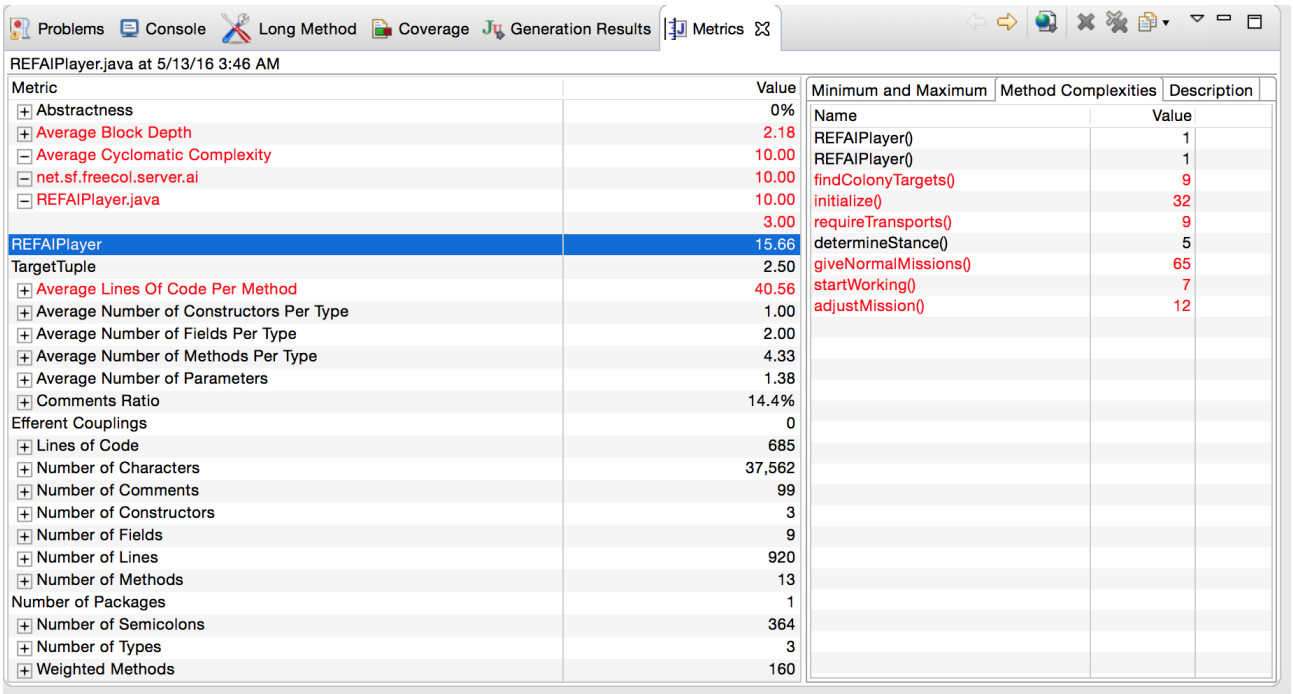
Minimum and Maximum | Method Complexities | Description

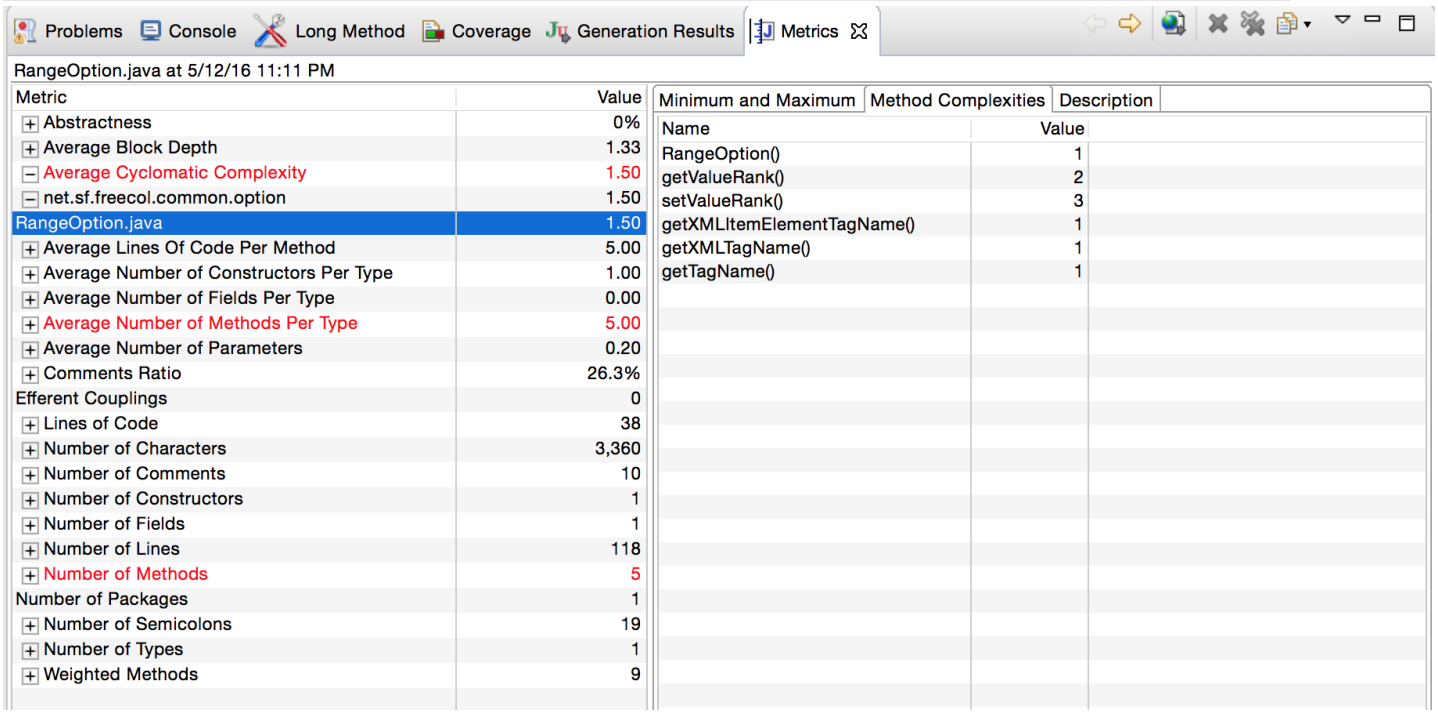
Name	Value
ServerUnit()	1
ServerUnit()	1
ServerUnit()	2
ServerUnit()	4
csNewTurn()	42
csImproveTile()	13
csEmbark()	6
csRepairUnit()	2
getSlowedBy()	13
csNativeBurialGround()	1
csExploreLostCityRumour()	40
csActivateSentries()	2
collectNewTiles()	1
csMove()	51
getServerXMLElementTagName()	1

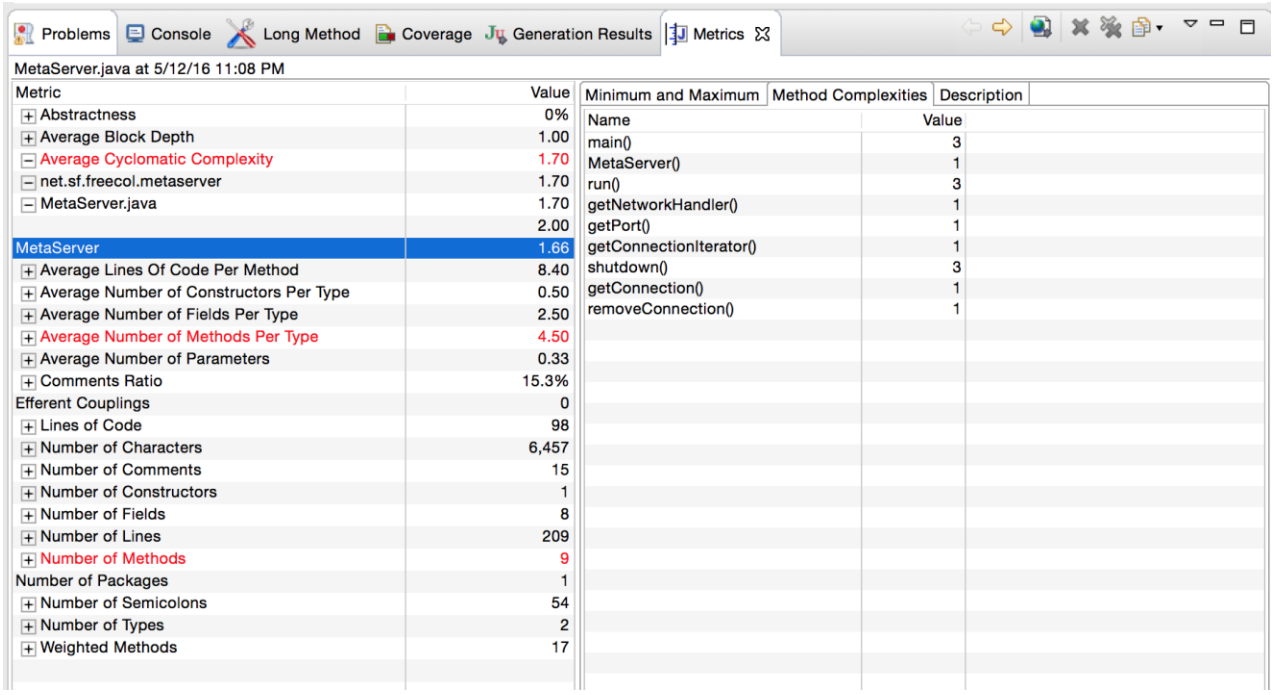

Problems
Console
Long Method
Coverage
Generation Results
Metrics
ServerColony.java at 5/12/16 10:08 PM
Metric
Value
Abstractness 0%
Average Block Depth 2.25
Average Cyclomatic Complexity 7.12
net.sf.freecol.server.model 7.12
ServerColony.java 7.12
Average Lines Of Code Per Method 35.56
Average Number 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
Minimum and Maximum
Method Complexities
Description
Name
Value
ServerColony() 1
ServerColony() 5
csNewTurn() 65
neededForBuildableType() 1
csBuildUnit() 2
ejectUnits() 6
csBuildBuilding() 6
csNextBuildable() 10
csEvictUsers() 3
csChangeOwner() 1
csFreeBuilding() 3
buildBuilding() 2
csEquipForRole() 3
csAddConvert() 3
destroyBuilding() 2
getServerXMLElementTagName() 1

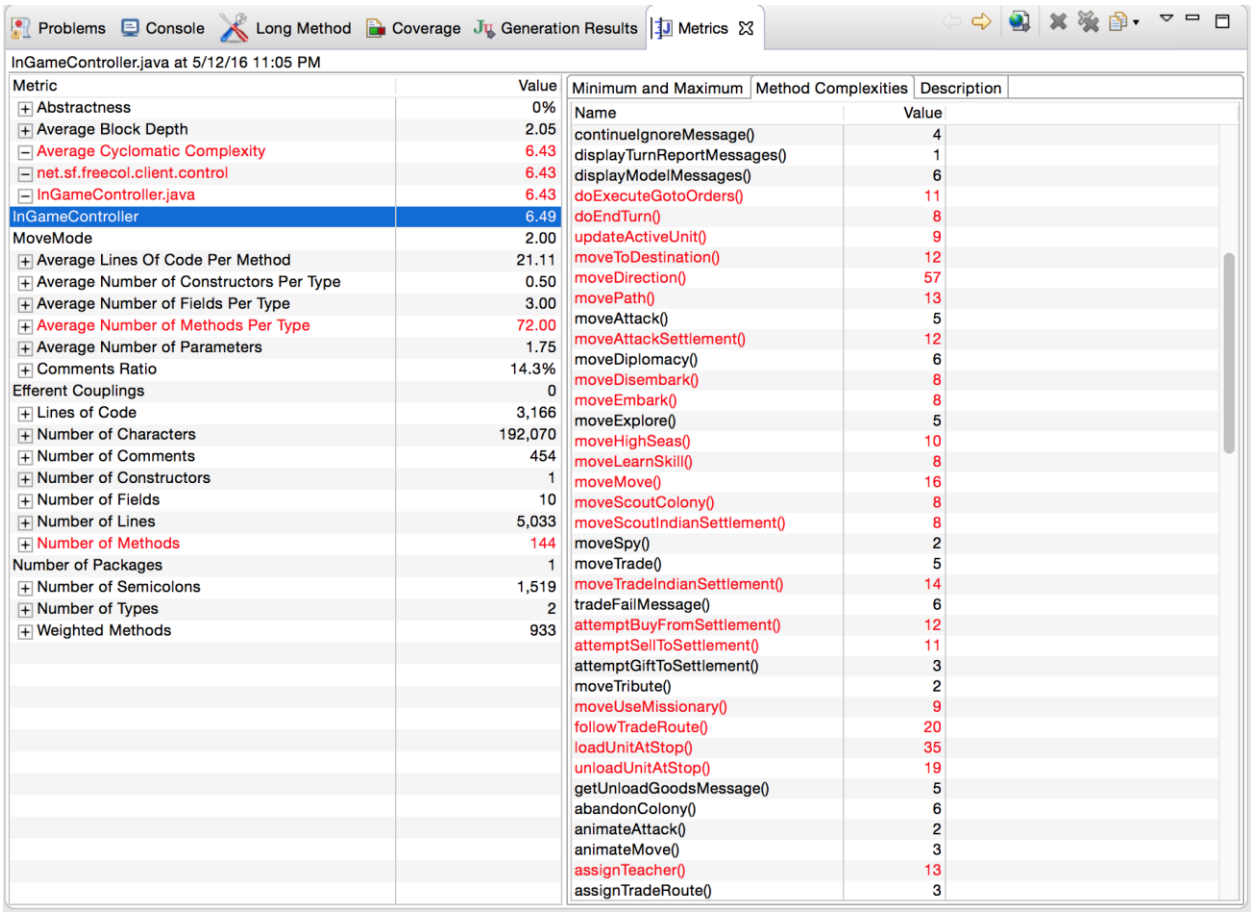

Problems
Console
Long Method
Coverage
Generation Results
Metrics
SelectOption.java at 5/12/16 11:13 PM
Metric
Value
Abstractness 0%
Average Block Depth 1.11
Average Cyclomatic Complexity 1.26
net.sf.freecol.common.option 1.26
SelectOption.java 1.26
Average Lines Of Code Per Method 5.73
Average Number 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
Minimum and Maximum
Method Complexities
Description
Name
Value
SelectOption() 1
getItemValues() 1
addItemValue() 1
clearItemValues() 1
localizeLabels() 1
getXMLItemElementTagName() 1
setValue() 4
writeAttributes() 1
writeChildren() 1
readAttributes() 1
readChildren() 1
readChild() 2
toString() 1
getXMLTagName() 1
getTagName() 1

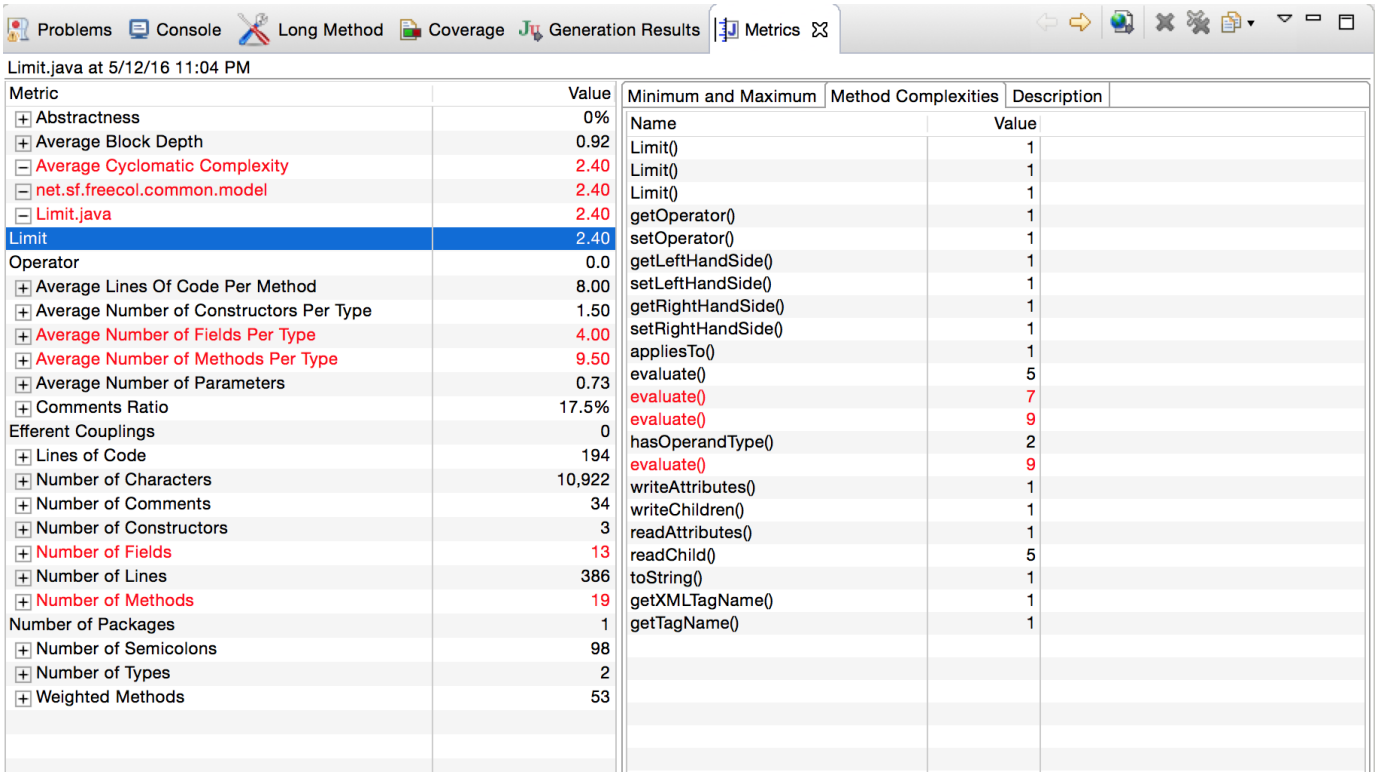

Problems
Console
Long Method
Coverage
Generation Results
Metrics
BuildingToolTip.java at 5/13/16 3:09 AM
Metric
Value
Abstractness 0%
Average Block Depth 3.50
Average Cyclomatic Complexity 9.50
net.sf.freecol.client.gui.panel 9.50
BuildingToolTip.java 9.50
Average Lines Of Code Per Method 63.00
Average 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
Minimum and Maximum
Method Complexities
Description
Name
Value
BuildingToolTip() 18
removeNotify() 1

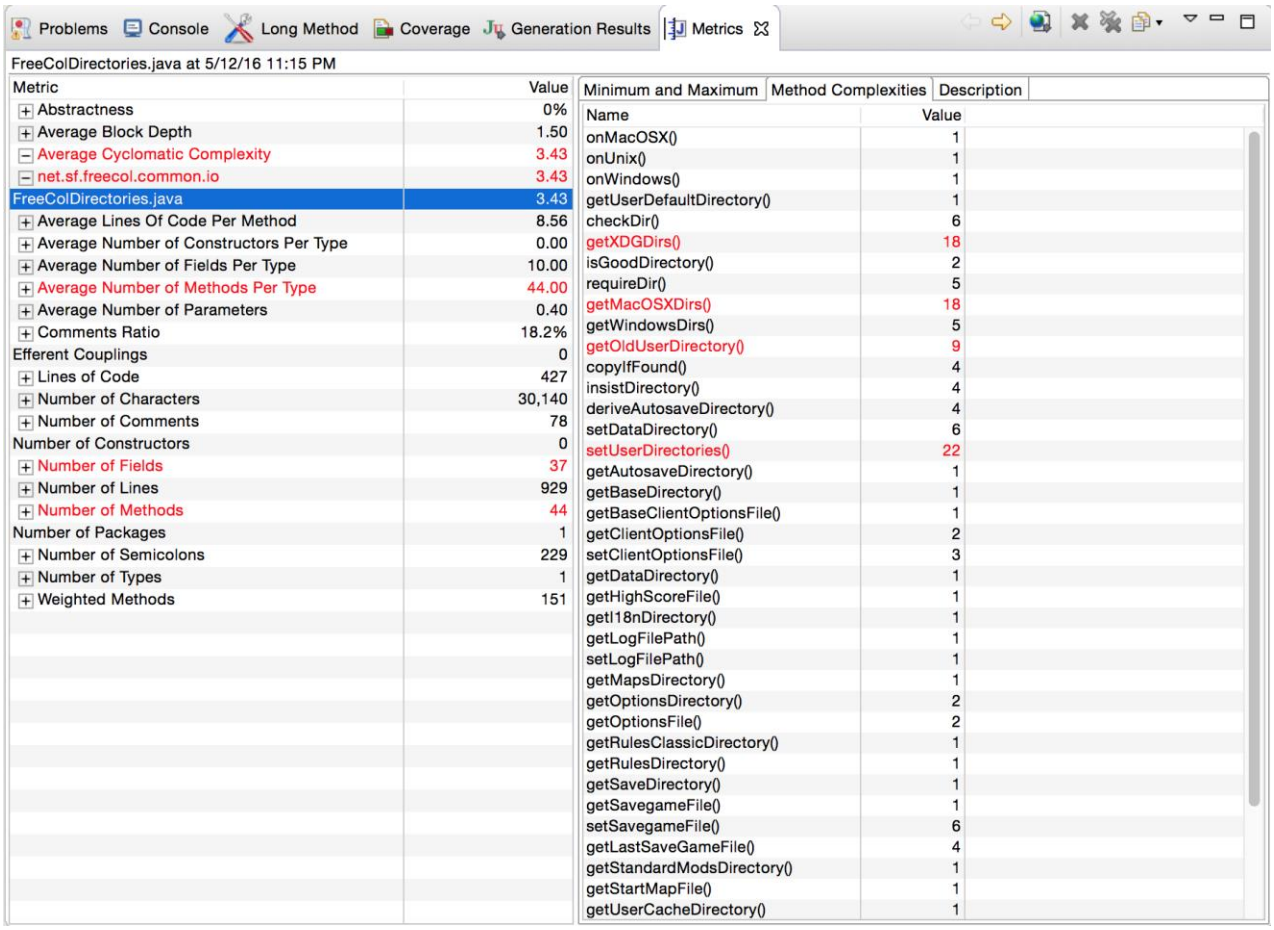

Problems
Console
Long Method
Coverage
Generation Results
Metrics
REFAIPlayer.java at 5/13/16 3:46 AM
Metric
Value
Abstractness 0%
Average Block Depth 2.18
Average Cyclomatic Complexity 10.00
net.sf.freecol.server.ai 10.00
REFAIPlayer.java 10.00
3.00
REFAIPlayer 15.66
TargetTuple 2.50
Average Lines Of 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
Minimum and Maximum
Method Complexities
Description
Name
Value
REFAIPlayer() 1
REFAIPlayer() 1
findColonyTargets() 9
initialize() 32
requireTransports() 9
determineStance() 5
giveNormalMissions() 65
startWorking() 7
adjustMission() 12

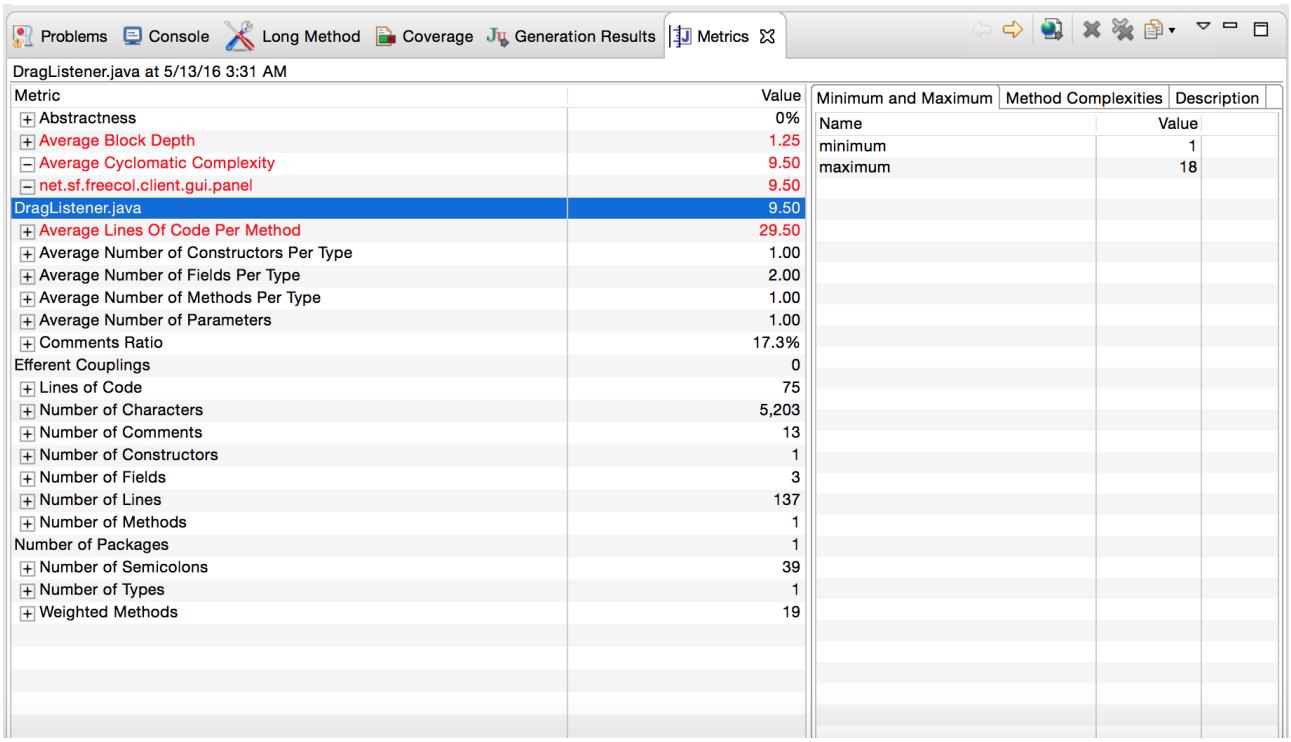

Problems
Console
Long Method
Coverage
Generation Results
Metrics
RangeOption.java at 5/12/16 11:11 PM
Metric
Value
Abstractness 0%
Average Block Depth 1.33
Average Cyclomatic Complexity 1.50
net.sf.freecol.common.option 1.50
RangeOption.java 1.50
Average Lines Of Code Per Method 5.00
Average Number of 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
Minimum and Maximum
Method Complexities
Description
Name
Value
RangeOption() 1
getValueRank() 2
setValueRank() 3
getXMLItemElementTagName() 1
getXMLTagName() 1
getTagName() 1


Problems
Console
Long Method
Coverage
Generation Results
Metrics
MetaServer.java at 5/12/16 11:08 PM
Metric
Value
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
Minimum and Maximum
Method Complexities
Description
Name
Value
main()
3
MetaServer()
1
run()
3
getNetworkHandler()
1
getPort()
1
getConnectionIterator()
1
shutdown()
3
getConnection()
1
removeConnection()
1


Problems
Console
Long Method
Coverage
Generation Results
Metrics
InGameController.java at 5/12/16 11:05 PM
Metric
Value
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
Minimum and Maximum
Method Complexities
Description
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

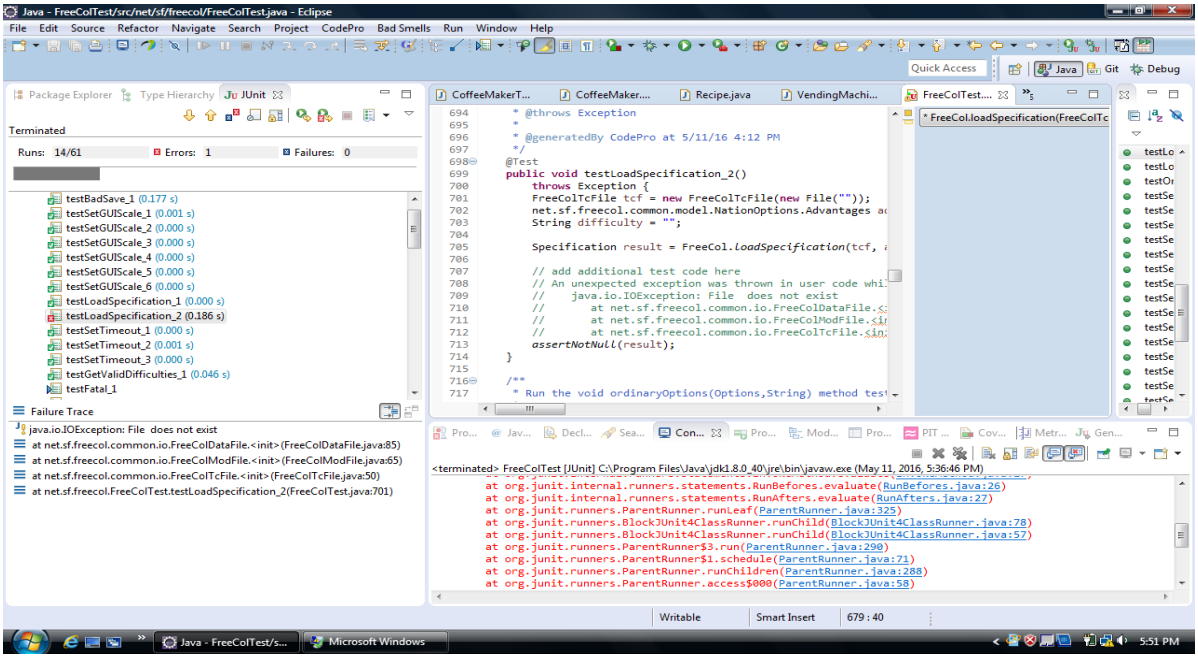

Problems
Console
Long Method
Coverage
Generation Results
Metrics
Limit.java at 5/12/16 11:04 PM
Metric
Value
Abstractness 0%
Average Block Depth 0.92
Average Cyclomatic Complexity 2.40
net.sf.freecol.common.model 2.40
Limit.java 2.40
Limit 2.40
Operator 0.0
Average Lines Of Code Per Method 8.00
Average 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
Minimum and Maximum
Method Complexities
Description
Name
Value
Limit() 1
Limit() 1
Limit() 1
getOperator() 1
setOperator() 1
getLeftHandSide() 1
setLeftHandSide() 1
getRightHandSide() 1
setRightHandSide() 1
appliesTo() 1
evaluate() 5
evaluate() 7
evaluate() 9
hasOperandType() 2
evaluate() 9
writeAttributes() 1
writeChildren() 1
readAttributes() 1
readChild() 5
toString() 1
getXMLTagName() 1
getTagName() 1

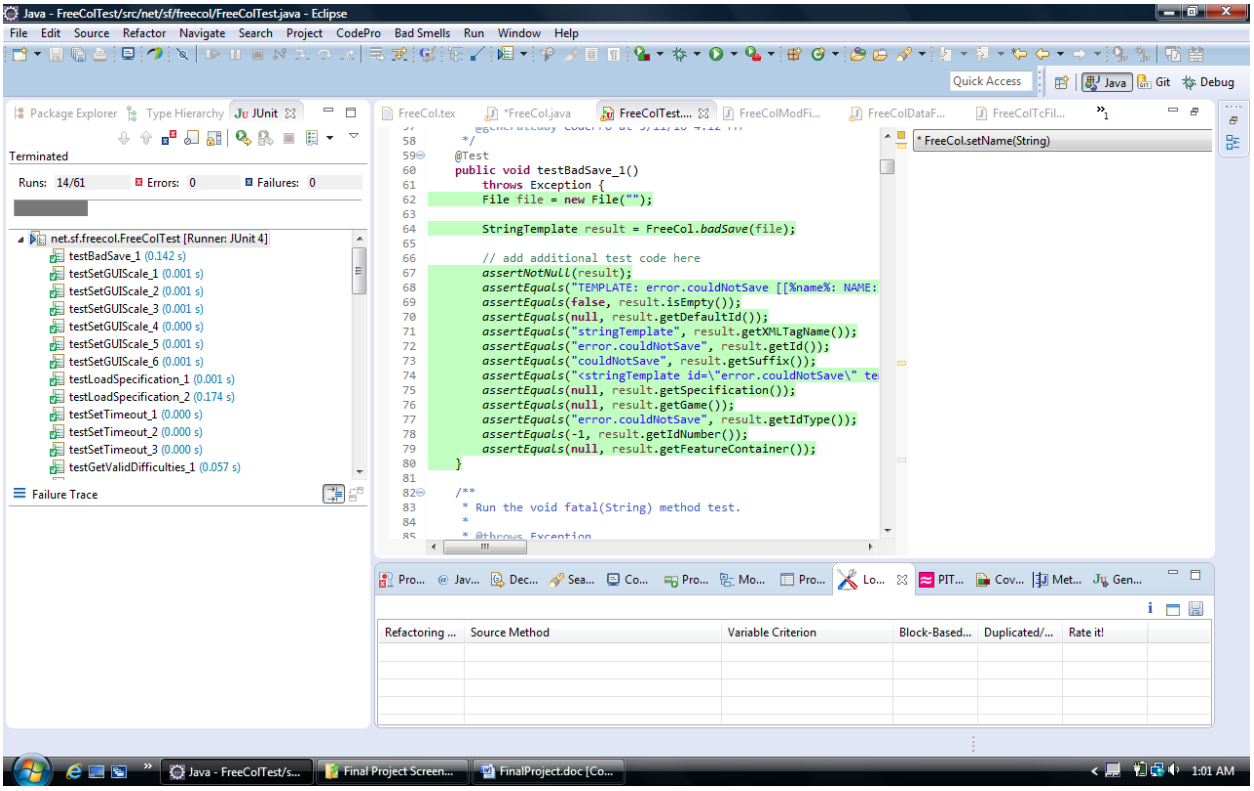

Problems
Console
Long Method
Coverage
Generation Results
Metrics
FreeColDirectories.java at 5/12/16 11:15 PM
Metric
Value
Abstractness 0%
Average Block Depth 1.50
Average Cyclomatic Complexity 3.43
net.sf.freecol.common.io 3.43
FreeColDirectories.java 3.43
Average Lines Of Code Per Method 8.56
Average 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
Minimum and Maximum
Method Complexities
Description
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



Jose:

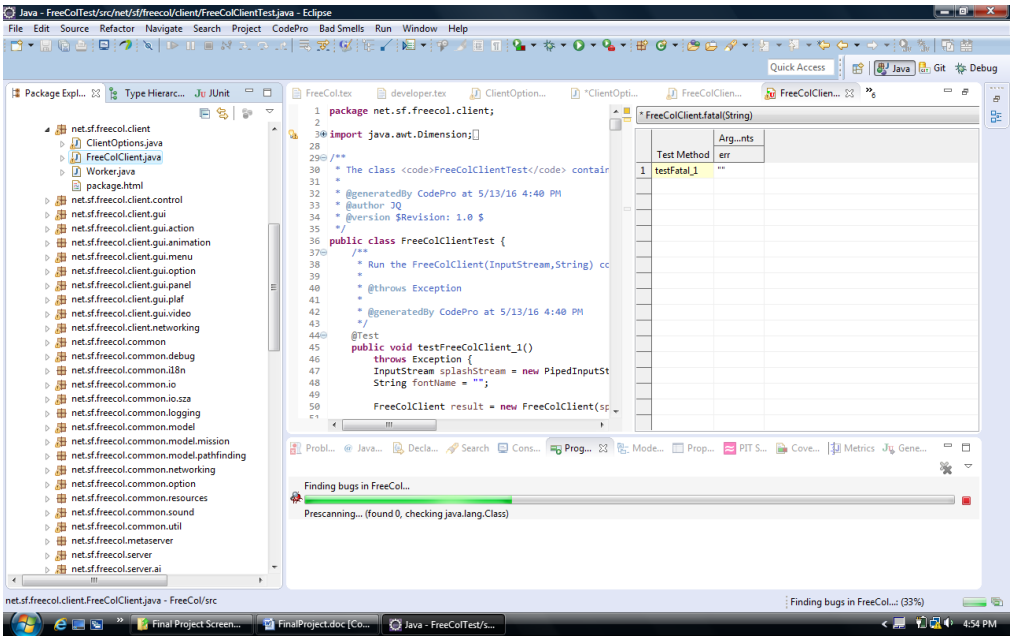
Jose is to assist with refactoring methods using JDeodorant, and Eclipse Refactoring tools, create JUnit test cases, fix style issues found with CheckStyle

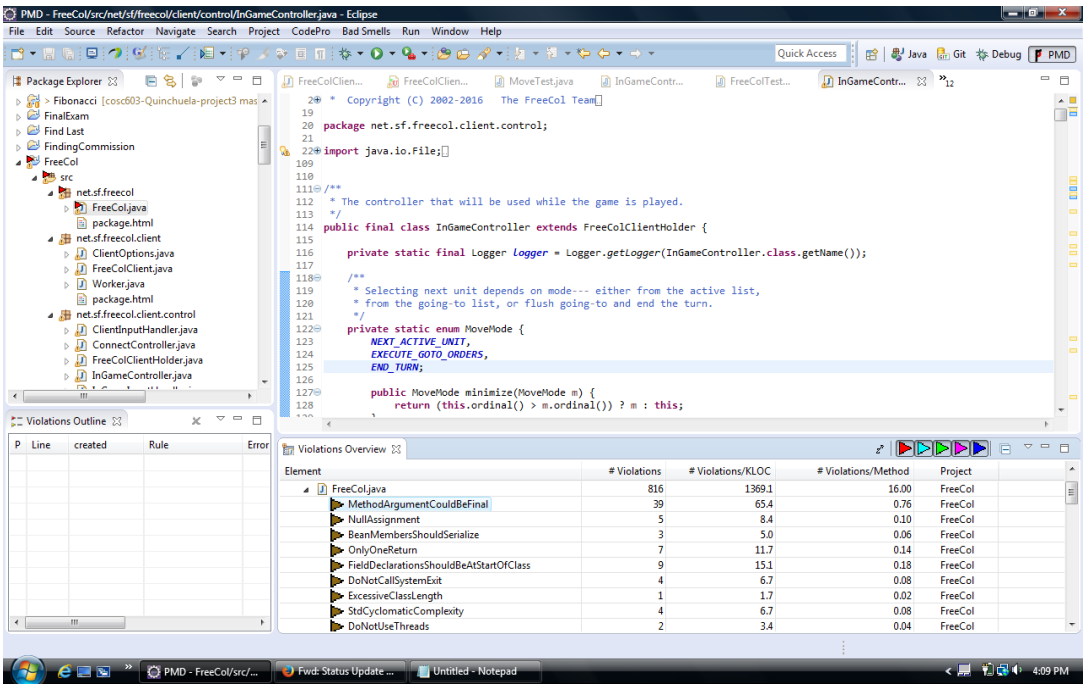




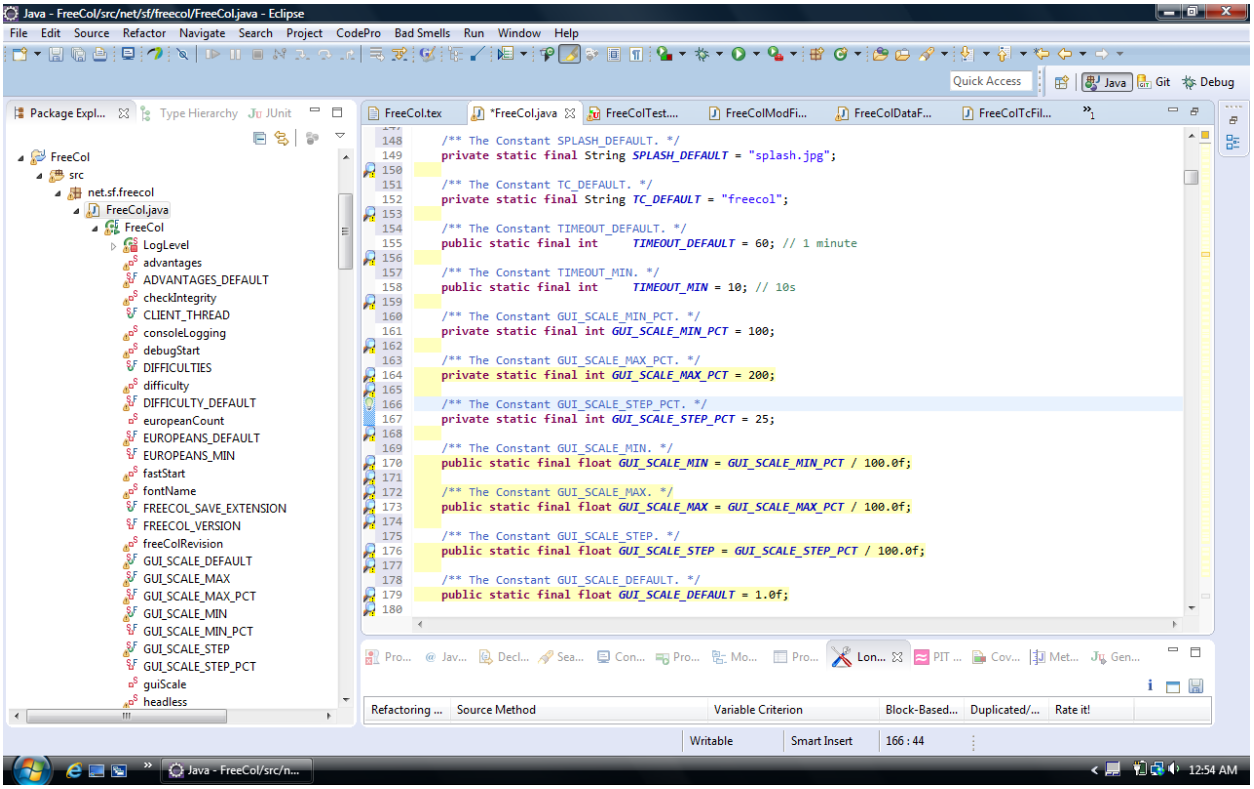
I had questions where it asked me to fix a null value but I left it there because it was part of a variable assignation.

No bugs were found with FindBugs

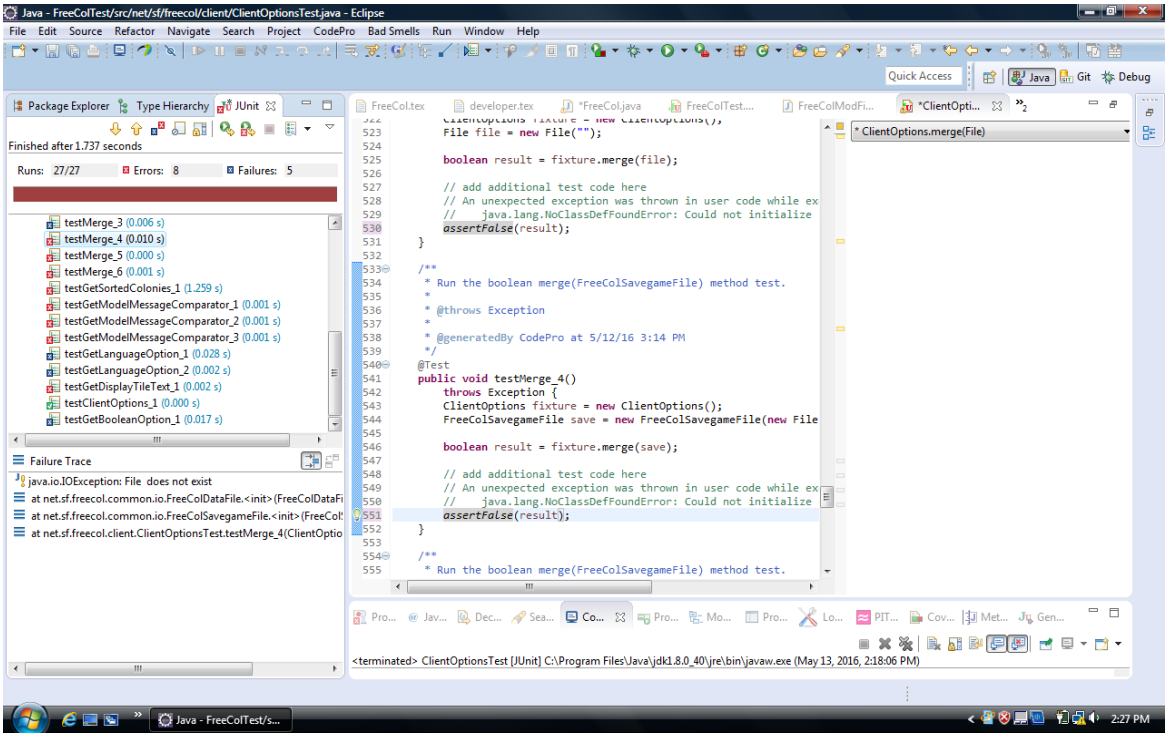


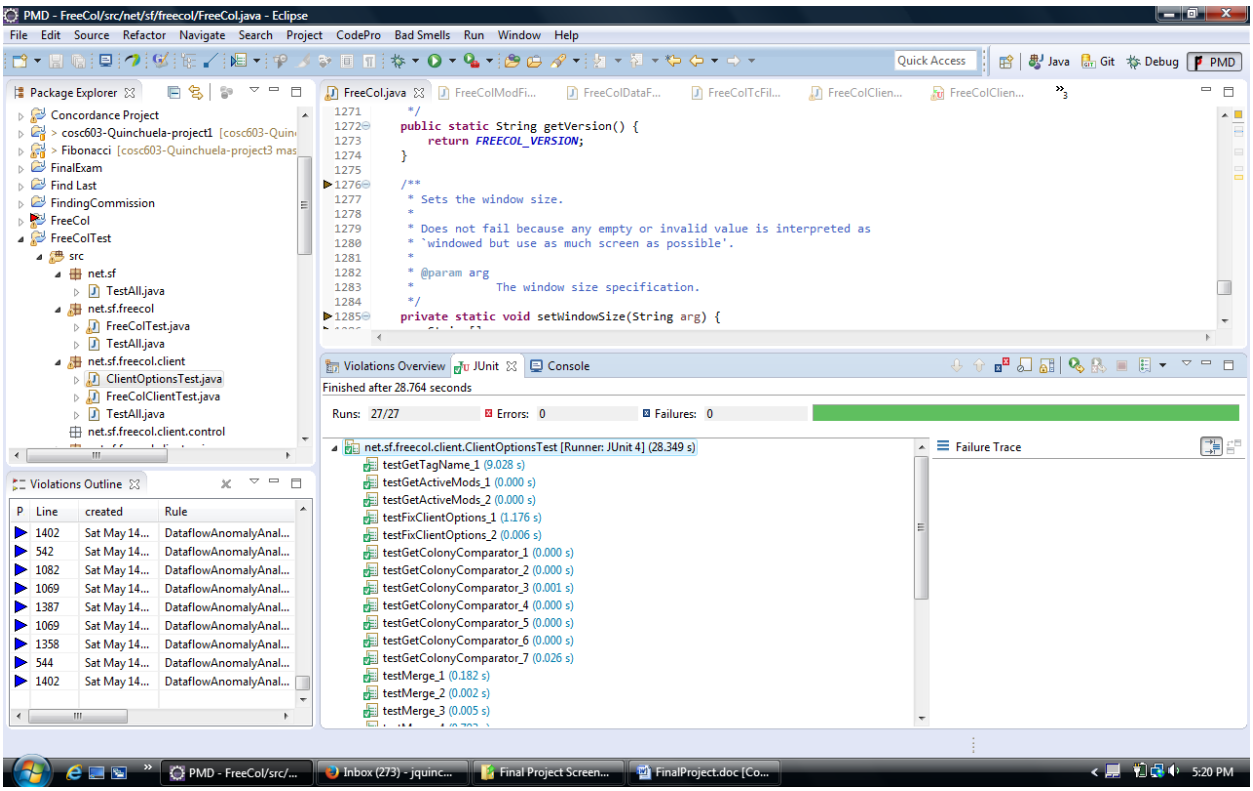


Checklist used. I used Eclipse Formatter tool to clean up most of the warnings.

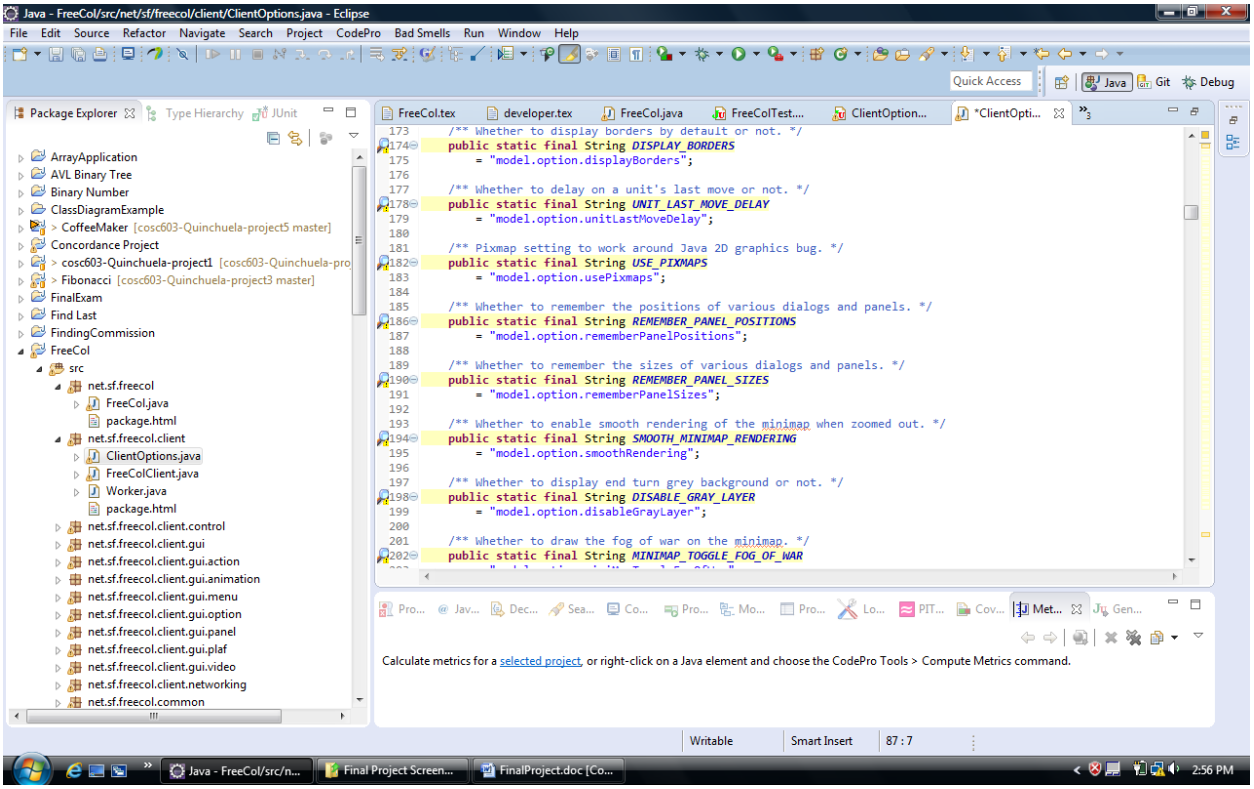


More test cases:





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



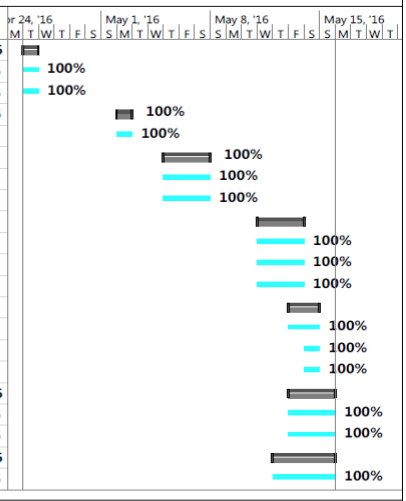
Identify the groups responsible for the environmental needs above.

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

Software Project Schedule

Project1_freecol

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Creating the GitHub repository**	**1 day**	**Tue 4/26/16**	**Tue 4/26/16**
2		Select the name for the repository	1 day	Tue 4/26/16	Tue 4/26/16
3		Setting up the Repository with team	1 day	Tue 4/26/16	Tue 4/26/16
4		**FreeCol initial upload**	**1 day**	**Mon 5/2/16**	**Mon 5/2/16**
5		Initial upload of freecol	1 day	Mon 5/2/16	Mon 5/2/16
6		**Setting up FreeCol**	**2 days**	**Thu 5/5/16**	**Sat 5/7/16**
7		Setting up and Running FreeCol	3 days	Thu 5/5/16	Sat 5/7/16
8		Setup and Execute FreeCol Successfi	3 days	Thu 5/5/16	Sat 5/7/16
9		**Will-Improving quality of FreeCol**	**3 days**	**Wed 5/11/16**	**Fri 5/13/16**
10		Reducing Cyclomatic Complexity, ref	3 days	Wed 5/11/16	Fri 5/13/16
11		Javadoc generation	3 days	Wed 5/11/16	Fri 5/13/16
12		Bugs found and use of PMD tool	3 days	Wed 5/11/16	Fri 5/13/16
13		**Erica-Improving quality of FreeCol**	**2 days**	**Fri 5/13/16**	**Sat 5/14/16**
14		Selecting FreeCol Components	2 days	Fri 5/13/16	Sat 5/14/16
15		Syncing repository	1 day	Sat 5/14/16	Sat 5/14/16
16		Merge remote-tracking branch	1 day	Sat 5/14/16	Sat 5/14/16
17		**Erica/Lauren- Improving quality of Fre**	**2 days**	**Fri 5/13/16**	**Sun 5/15/16**
18		Test cases created	2 days	Fri 5/13/16	Sun 5/15/16
19		Improving quality and testing	2 days	Fri 5/13/16	Sun 5/15/16
20		**Jose-Improving quality of FreeCol**	**3 days**	**Thu 5/12/16**	**Sun 5/15/16**
21		Test cases created	3 days	Thu 5/12/16	Sun 5/15/16

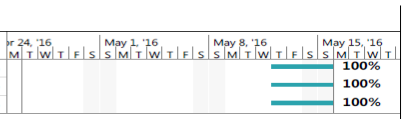


Critical, Critical Split, Critical Progress, Task, Split, Task Progress, Manual Task, Start-only
Finish-only, Duration-only, Baseline, Baseline Split, Baseline Milestone, Milestone, Summary Progress, Summary
Manual Summary, Project Summary, External Tasks, External Milestone, Inactive Task, Inactive Milestone, Inactive Summary, Deadline

Page 1

Project1_freecol

ID	Task Mode	Task Name	Duration	Start	Finish
22		Javadoc generation	3 days	Thu 5/12/16	Sun 5/15/16
23		Checkstyle used	3 days	Thu 5/12/16	Sun 5/15/16
24		Bugs not found	3 days	Thu 5/12/16	Sun 5/15/16



Critical, Critical Split, Critical Progress, Task, Split, Task Progress, Manual Task, Start-only
Finish-only, Duration-only, Baseline, Baseline Split, Baseline Milestone, Milestone, Summary Progress, Summary
Manual Summary, Project Summary, External Tasks, External Milestone, Inactive Task, Inactive Milestone, Inactive Summary, Deadline

Page 2