Robert and Pandora

FreeCol

Software Quality Assurance Plan

Version: (1) Date: (05/16/2016)

Document History and Distribution

Revision #	Revision Date	Description of Change	Author
001	29 Apr 2016	Establish Initial template	R. Bathmann
002	30 Apr 2016	Removed extraneous items from Plan and updated sections as appropriate	R. Bathmann
003	07 May 2016	Updated Section 4: Approach	J. Podier
004	12 May 2016	Reviewed entire document and made necessary updates to various sections	J. Podier
005	14 May 2016	Revised and Updated Sections 2, 3 and 5.	R. Bathmann & J. Podier
006	15 May 2016	Made modifications to the entire document	R. Bathmann & J. Podier

TABLE OF CONTENTS

1. INTRODUCTION

FreeCol is an open-source implementation of a Colonization-style game in which the New World must be settled. A group of contributors are building the game with the intent to release version 1.0.

The software is hosted on SourceForge using Git for version control. The SourceForge issue tracker is used to track bugs and feature requests.

The existing test strategy for FreeCol combines unit testing, regression testing, and community-driven user interface and gameplay manual testing. This test plan will present a path forward to improve the current test plan with an emphasis on ten classes for improvement.

1.1 Objectives

The scope of additional testing and code quality improvement activities will be limited to enhancing code coverage of unit test cases, reducing average cyclomatic complexity of classes, and refactoring to eliminate typical code smells.

Test artifacts are listed in the deliverables section. These artifacts will be completed by May 16, 2016.

1.2 Testing Strategy

The first step in testing and improving FreeCol will be to identify a subset of ten (10) classes that need improvement. We will use tools to perform static analysis. We will accomplish that by analyzing the source code for statement coverage, average cyclomatic complexity, and code smells.

Due to the large, complex code base and limited resources, we will not impose a strict process for class selection. We will use our best judgment to determine realistic changes to the source code to make improvements in line with our objectives.

In the course of testing exercises, we determined that we will not limit ourselves to 10 classes in order to best improve the product. We will identify 10 overall improvements, each of which may correspond to updating one or more classes. For instance, to make an improvement in which the “scariest” FindBugs results are addressed, several classes will be updated to fix only those problems.

1.3 Scope

This test plan is a living document. We will make updates throughout the lifecycle of our testing activities. This document is maintained as a shared resource using Google Drive to increase collaboration and eliminate merge conflicts. Our last draft of this test plan will be posted in PDF format to GitHub with the modified source code.

1.5 Definitions and Acronyms

There are no definitions or Acronyms.

2. TEST ITEMS

FreeCol does not have a format requirements document. However, its overall goal is to recreate the functionality that existed in the original Colonization game.

The user manual for FreeCol is available at http://www.freecol.org/docs/FreeCol.pdf.

Issues with the software are tracked using SourceForge. It is available at https://sourceforge.net/p/freecol/_list/tickets.

2.1 Program Modules

The following classes will be tested and improved in accordance with the test plan:

1. net.sf.freecol.server.model.ServerColony
2. net.sf.freecol.server.model.ServerPlayer
3. net.sf.freecol.server.control.ChangeSet
4. net.sf.freecol.server.model.TransactionSession
5. net.sf.freecol.common.model.Limit
6. net.sf.freecol.common.option.RangeOption
7. net.sf.freecol.common.SelectOption
8. net.sf.freecol.metaserver.MetaServer
9. net.sf.freecol.server.networking.Server
10. net.sf.freecol.client.control.InGameController
11. net.sf.freecol.server.ai.GoodsWish
12. net.sf.freecol.common.networking.Connection

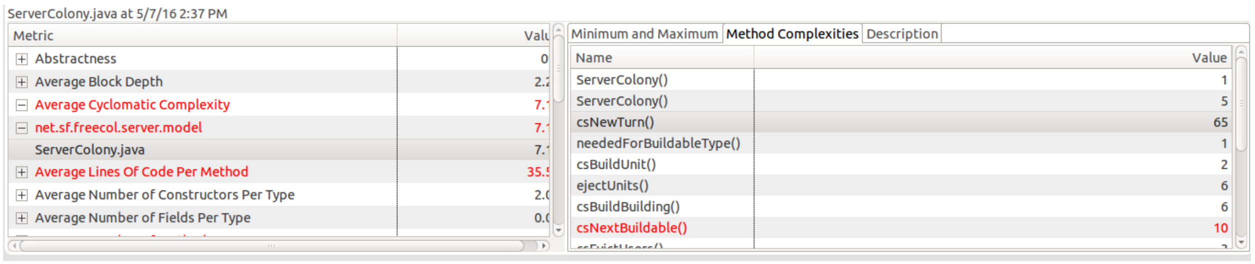
2.2 User Procedures

3. FEATURES TO BE TESTED

This section will be used to document the ten (10) classes we improved and the changes we made.

1. Complexity of csNewTurn method in net.sf.freecol.server.model.ServerColony

The cyclomatic complexity of this method was originally sixty-five (65). We reduced the complexity by refactoring via extracting methods. We were able to achieve a final cyclomatic complexity of 10. Although this is slightly higher than ideal, further reductions would require additional data types to store changes to variables currently defined within method scope. The screenshots below show the before and after complexities as reported by Google CodePro.

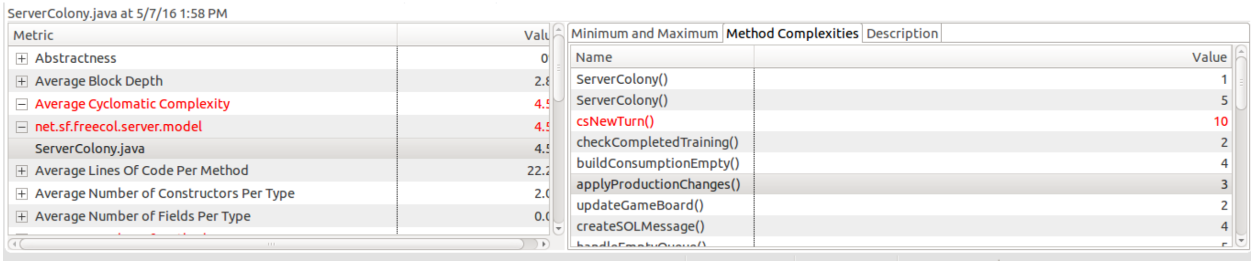


ServerColony.java at 5/7/16 2:37 PM

Metric	Valu
⊞ Abstractness	0
⊞ Average Block Depth	2.2
⊟ Average Cyclomatic Complexity	7.1
⊟ net.sf.freecol.server.model	7.1
ServerColony.java	7.1
⊞ Average Lines Of Code Per Method	35.5
⊞ Average Number of Constructors Per Type	2.0
⊞ Average Number of Fields Per Type	0.0

Minimum and Maximum | Method Complexities | Description

Name	Value
ServerColony()	1
ServerColony()	5
csNewTurn()	65
neededForBuildableType()	1
csBuildUnit()	2
ejectUnits()	6
csBuildBuilding()	6
csNextBuildable()	10



ServerColony.java at 5/7/16 1:58 PM

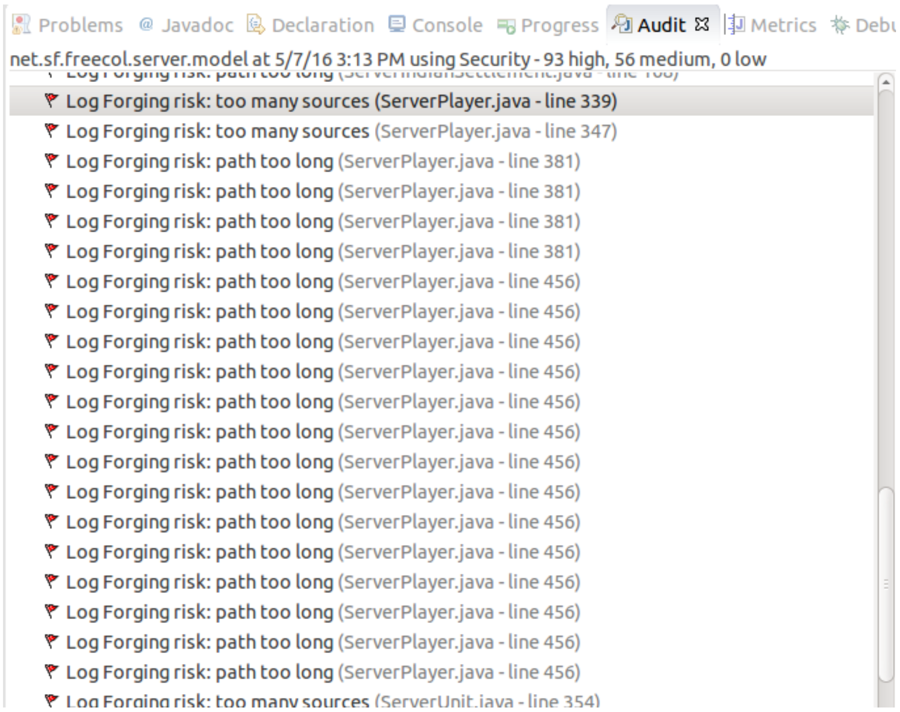
Metric	Valu
⊞ Abstractness	0
⊞ Average Block Depth	2.8
⊟ Average Cyclomatic Complexity	4.5
⊟ net.sf.freecol.server.model	4.5
ServerColony.java	4.5
⊞ Average Lines Of Code Per Method	22.2
⊞ Average Number of Constructors Per Type	2.0
⊞ Average Number of Fields Per Type	0.0

Minimum and Maximum | Method Complexities | Description

Name	Value
ServerColony()	1
ServerColony()	5
csNewTurn()	10
checkCompletedTraining()	2
buildConsumptionEmpty()	4
applyProductionChanges()	3
updateGameBoard()	2
createSOLMessage()	4

2. Log forging vulnerability in net.sf.freecol.server.model.ServerPlayer

A log forging vulnerability existed in the ServerPlayer class that could allow not validated user input to be included in log files. This could allow a malicious user to include false information by forging new log lines. We mitigated this by first sanitizing the strings to be logged in this class.



```
net.sf.freecol.server.model at 5/7/16 3:13 PM using Security - 93 high, 56 medium, 0 low
Log Forging risk: too many sources (ServerPlayer.java - line 339)
Log Forging risk: too many sources (ServerPlayer.java - line 347)
Log Forging risk: path too long (ServerPlayer.java - line 381)
Log Forging risk: path too long (ServerPlayer.java - line 381)
Log Forging risk: path too long (ServerPlayer.java - line 381)
Log Forging risk: path too long (ServerPlayer.java - line 381)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: path too long (ServerPlayer.java - line 456)
Log Forging risk: too many sources (ServerUnit.java - line 354)
```

3. Javadoc repair in several classes in the net.st.freecol.server.model package

The classes that extend TransactionSession had incorrect or incomplete JavaDoc. We repaired the JavaDoc for these classes and added additional comments as needed. In particular, several method in the TradeSession class did not include any explanation to indicate the responsibilities of the method.

4. Feature envy for methods in net.sf.freecol.server.control.ChangeSet

Some methods in the ServerColony and ServerPlayer classes use an excessive amount of methods from the ChangeSet class. That is, these classes have feature envy for ChangeSet. To resolve this code smell, we moved two methods from ServerColony and ServerPlayer to the ChangeSet class.

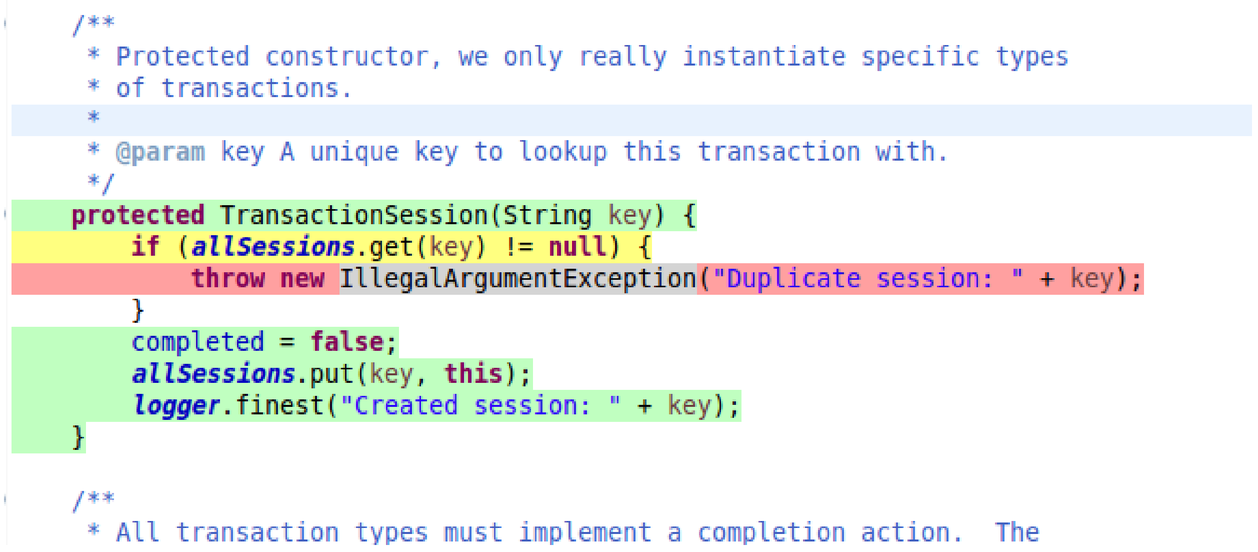
5. Improve code coverage for net.sf.freecol.server.model.TransactionSession

TransactionSession did not have code coverage for several statements. Since this is an abstract class, we created a hollow implementation of a class that inherited from it. We called this the FakeTransactionSession class.

We added a test case to test a thrown exception from the constructor to handle a situation in which duplicate session keys are managed in the same game instance. Since these keys are stored in a static field inside the TransactionSession class, we created two instances of FakeTransactionSession with the same key to produce the exception and test for it.

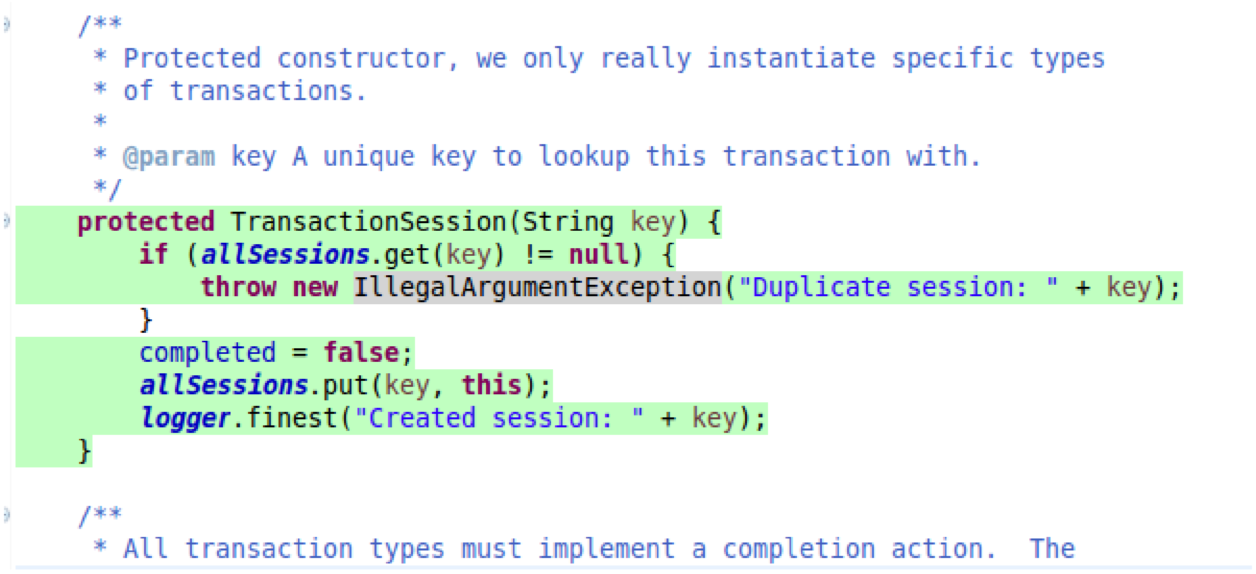
Static fields present an interesting problem for unit testing since they can break the model of independent unit tests. If a unit test does not “clean up” after itself and restore the state of the static field, it could have consequences for other test cases. We observed this when we attempted to add another unit test with same key value as another test case and the code correctly produced an exception.

This class is difficult to test because many methods do not return a value and only change the internal state not accessible to other classes.



```
    /**
     * Protected constructor, we only really instantiate specific types
     * of transactions.
     *
     * @param key A unique key to lookup this transaction with.
     */
    protected TransactionSession(String key) {
        if (allSessions.get(key) != null) {
            throw new IllegalArgumentException("Duplicate session: " + key);
        }
        completed = false;
        allSessions.put(key, this);
        logger.finest("Created session: " + key);
    }

    /**
     * All transaction types must implement a completion action.  The
```

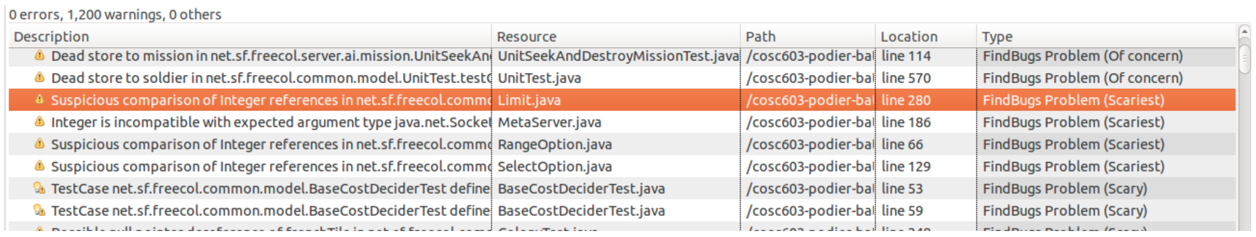


```
    /**
     * Protected constructor, we only really instantiate specific types
     * of transactions.
     *
     * @param key A unique key to lookup this transaction with.
     */
    protected TransactionSession(String key) {
        if (allSessions.get(key) != null) {
            throw new IllegalArgumentException("Duplicate session: " + key);
        }
        completed = false;
        allSessions.put(key, this);
        logger.finest("Created session: " + key);
    }

    /**
     * All transaction types must implement a completion action.  The
```

6. Review and correct the “scariest” FindBugs problems

We used the FindBugs Eclipse plug-in on the entire FreeCol project. The results showed the project contained four problems with the highest criticality level: scariest. The problems were grouped into categories: Incompatible arguments and incorrect Integer comparison.



0 errors, 1,200 warnings, 0 others

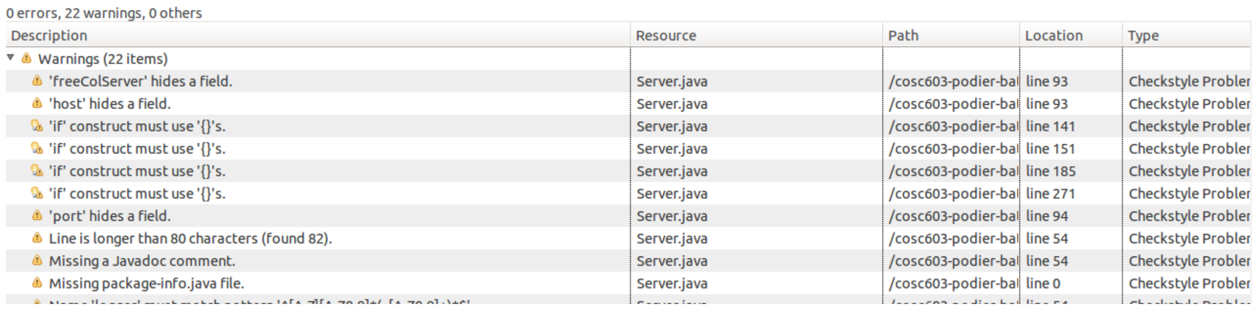
Description	Resource	Path	Location	Type
Dead store to mission in net.sf.freecol.server.ai.mission.UnitSeekAn	UnitSeekAndDestroyMissionTest.java	/cosc603-podier-bal	line 114	FindBugs Problem (Of concern)
Dead store to soldier in net.sf.freecol.common.model.UnitTest.testC	UnitTest.java	/cosc603-podier-bal	line 570	FindBugs Problem (Of concern)
Suspicious comparison of Integer references in net.sf.freecol.comm	Limit.java	/cosc603-podier-bal	line 280	FindBugs Problem (Scariest)
Integer is incompatible with expected argument type java.net.Socket	MetaServer.java	/cosc603-podier-bal	line 186	FindBugs Problem (Scariest)
Suspicious comparison of Integer references in net.sf.freecol.comm	RangeOption.java	/cosc603-podier-bal	line 66	FindBugs Problem (Scariest)
Suspicious comparison of Integer references in net.sf.freecol.comm	SelectOption.java	/cosc603-podier-bal	line 129	FindBugs Problem (Scariest)
TestCase net.sf.freecol.common.model.BaseCostDeciderTest define	BaseCostDeciderTest.java	/cosc603-podier-bal	line 53	FindBugs Problem (Scary)
TestCase net.sf.freecol.common.model.BaseCostDeciderTest define	BaseCostDeciderTest.java	/cosc603-podier-bal	line 59	FindBugs Problem (Scary)

There was one bug in the code in the first category. The shutdown method in net.sf.freecol.metaserver.MetaServer class was passing ‘0’ to the remove method on a HashMap. This method should take the key of type Object, not a primitive integer. We re-wrote this concept by iterating over the HashMap and closing the connections one at a time. Then, we used the clear method to empty the HashMap.

There were three bugs in the code for the second category. The Limit, RangeOption, and SelectOption classes used ‘==’ to compare Integer instances. While this is acceptable for integer primitives, the equals method must be used when comparing Integer objects. We updated these three classes to correctly use the equals method.

7. Resolve Checkstyle problems in net.sf.freecol.server.networking.Server

We selected this class as a Checkstyle candidate because it has a moderate number of lines and does not contain complex gameplay logic. There were a total of 22 style violations according to the Eclipse Checkstyle rules.



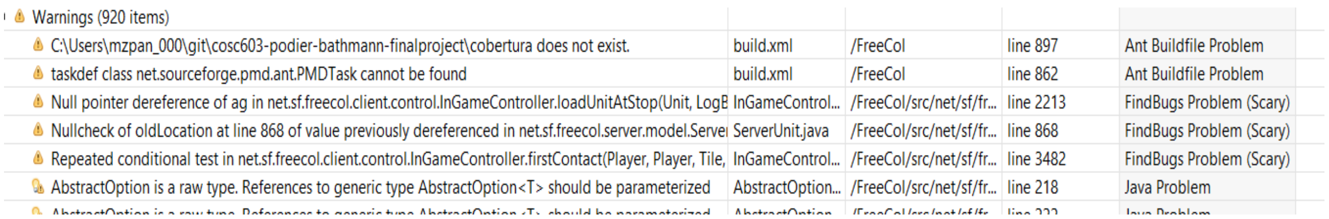
0 errors, 22 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (22 items)				
'freeColServer' hides a field.	Server.java	/cosc603-podier-ba	line 93	Checkstyle Probler
'host' hides a field.	Server.java	/cosc603-podier-ba	line 93	Checkstyle Probler
'if' construct must use '{}'s.	Server.java	/cosc603-podier-ba	line 141	Checkstyle Probler
'if' construct must use '{}'s.	Server.java	/cosc603-podier-ba	line 151	Checkstyle Probler
'if' construct must use '{}'s.	Server.java	/cosc603-podier-ba	line 185	Checkstyle Probler
'if' construct must use '{}'s.	Server.java	/cosc603-podier-ba	line 271	Checkstyle Probler
'port' hides a field.	Server.java	/cosc603-podier-ba	line 94	Checkstyle Probler
Line is longer than 80 characters (found 82).	Server.java	/cosc603-podier-ba	line 54	Checkstyle Probler
Missing a Javadoc comment.	Server.java	/cosc603-podier-ba	line 54	Checkstyle Probler
Missing package-info.java file.	Server.java	/cosc603-podier-ba	line 0	Checkstyle Probler

Most violations were related to missing *final* keywords on method arguments, missing Javadoc, and missing curly braces. We left 5 violations for a missing package-info.java file, hidden field violations, and incorrect casing on the static logger field name. These are common practices and should be removed from the Checkstyle rules for the project.

8. Resolve FindBugs problems in net.sf.freecol.client.control.InGameController

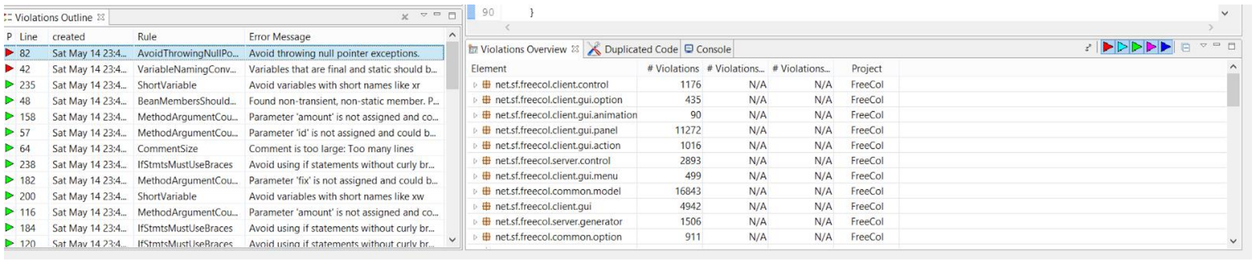
We selected the InGameController class because of the number of lines of code and did not impact the game logic. There were three FindBugs errors in this class which was related to repeated conditional test, null pointer dereference, and a nullcheck of old location. We corrected two of the three errors because the nullcheck error was in another class. In the repeated condition it wanted to check if players were null or other but the initial condition statement prior to the correction was evaluating if player equal null twice versus if one player was null and the other player was other. In the other error it wanted to check for unknown values but if the code was to ever hit this line of code, an exception would be thrown, therefore the statement had to be corrected to ensure that an exception is not thrown.



Warnings (920 items)				
C:\Users\mzpan_000\git\cosc603-podier-bathmann-finalproject\cobertura does not exist.	build.xml	/FreeCol	line 897	Ant Buildfile Problem
taskdef class net.sourceforge.pmd.ant.PMDTask cannot be found	build.xml	/FreeCol	line 862	Ant Buildfile Problem
Null pointer dereference of ag in net.sf.freecol.client.control.InGameController.loadUnitAtStop(Unit, LogB	InGameControl...	/FreeCol/src/net/sf/fr...	line 2213	FindBugs Problem (Scary)
Nullcheck of oldLocation at line 868 of value previously dereferenced in net.sf.freecol.server.model.Serve	ServerUnit.java	/FreeCol/src/net/sf/fr...	line 868	FindBugs Problem (Scary)
Repeated conditional test in net.sf.freecol.client.control.InGameController.firstContact(Player, Player, Tile,	InGameControl...	/FreeCol/src/net/sf/fr...	line 3482	FindBugs Problem (Scary)
AbstractOption is a raw type. References to generic type AbstractOption<T> should be parameterized	AbstractOption...	/FreeCol/src/net/sf/fr...	line 218	Java Problem

9. Resolve PMD violations in net.sf.freecol.server.ai.GoodsWish

We decided to run the PMD tool against the FreeCol program and identified two critical blockers and decided to fix those. One of the identified blockers was a violation for naming conventions in which variables that are static or final should all be capitalized, This error appear to be throughout the program and would potentially require a global change to many variables throughout the code. Therefore to illustrate how to fix the naming convention error we modified the logger variable to be capitalized. To adjust the other critical blocker, we addressed the NullPointerException by changing it to an IllegalArgumentException.



Violations Outline

P	Line	created	Rule	Error Message
🟥	82	Sat May 14 23:4...	AvoidThrowingNullPo...	Avoid throwing null pointer exceptions.
🟥	42	Sat May 14 23:4...	VariableNamingConv...	Variables that are final and static should b...
🟩	235	Sat May 14 23:4...	ShortVariable	Avoid variables with short names like xr
🟩	48	Sat May 14 23:4...	BeanMembersShould...	Found non-transient, non-static member. P...
🟩	158	Sat May 14 23:4...	MethodArgumentCou...	Parameter 'amount' is not assigned and co...
🟩	57	Sat May 14 23:4...	MethodArgumentCou...	Parameter 'id' is not assigned and could b...
🟩	64	Sat May 14 23:4...	CommentSize	Comment is too large: Too many lines
🟩	238	Sat May 14 23:4...	IfStmtsMustUseBraces	Avoid using if statements without curly br...
🟩	182	Sat May 14 23:4...	MethodArgumentCou...	Parameter 'fix' is not assigned and could b...
🟩	200	Sat May 14 23:4...	ShortVariable	Avoid variables with short names like xw
🟩	116	Sat May 14 23:4...	MethodArgumentCou...	Parameter 'amount' is not assigned and co...
🟩	184	Sat May 14 23:4...	IfStmtsMustUseBraces	Avoid using if statements without curly br...
🟩	120	Sat May 14 23:4...	IfStmtsMustUseBraces	Avoid using if statements without curly br...

```
}
```

Violations Overview | Duplicated Code | Console

Element	# Violations	# Violations...	# Violations...	Project
net.sf.freecol.client.control	1176	N/A	N/A	FreeCol
net.sf.freecol.client.gui.option	435	N/A	N/A	FreeCol
net.sf.freecol.client.gui.animation	90	N/A	N/A	FreeCol
net.sf.freecol.client.gui.panel	11272	N/A	N/A	FreeCol
net.sf.freecol.client.gui.action	1016	N/A	N/A	FreeCol
net.sf.freecol.server.control	2893	N/A	N/A	FreeCol
net.sf.freecol.client.gui.menu	499	N/A	N/A	FreeCol
net.sf.freecol.common.model	16843	N/A	N/A	FreeCol
net.sf.freecol.client.gui	4942	N/A	N/A	FreeCol
net.sf.freecol.server.generator	1506	N/A	N/A	FreeCol
net.sf.freecol.common.option	911	N/A	N/A	FreeCol

10. Several improvements to net.sf.freecol.common.networking.Connection

We determined that the Connection class is central to multiplayer gameplay and contained several defects. First, we fixed all the Javadoc in the class. Several methods were missing block comments explaining parameters, return values, and exceptions.

Second, we used CodePro to audit the code and fixed a use of a platform-dependant line separator. There were other instances of this problem in this class, but they occurred in the text that is transferred between game instances. Since these instances could be on separate platforms, the chosen line separator must not be changed to the platform-independent value.

Last, we corrected a “troubling” FindBugs error where a thread was started from the constructor. This is not a recommended practice since the object being constructed may not be fully initialized. We created a new start method in the class and updated all usages of the class to call the new method after the constructor.

4. APPROACH

The sections below describe how each of our software improvements are evaluated by different types of testing activities.

4.1 Component Testing

We will use the existing unit tests in the FreeCol project to validate our changes. If unit tests do not exist for code we are changing, we will create additional unit tests where feasible.

4.2 Integration Testing

We will perform limited integration testing. We will validate the game can start without error on our development systems.

4.3 Interface Testing

We do not have the resources to perform interface testing. Besides use of operating system and graphics APIs, the only interface in the game is to itself for multiplayer gaming.

4.4 Security Testing

We will perform limited security testing. Using static analysis tools, we will inspect the source code for common vulnerabilities.

4.5 Performance Testing

We will not perform performance testing.

4.6 Regression Testing

We will perform regression testing by executing the entire test suite after each change to the source code.

4.7 Acceptance Testing

We will not perform acceptance testing. Acceptance tests are the responsibility of the larger FreeCol community.

4.8 Beta Testing

FreeCol is currently beta software and available to the general public.

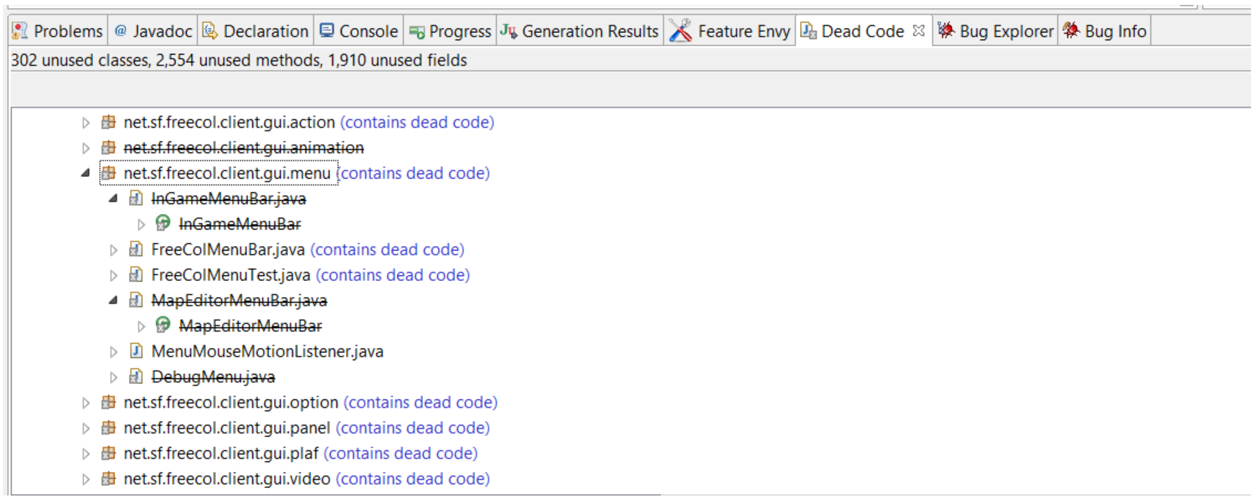
5. PASS/FAIL CRITERIA

Changes to the software will be exclusively verified by the JUnit test suite.

5.1 Suspension Criteria

Testing will be suspended when a particular avenue for improvement does not work out.

We encountered this situation when we selected the net.sf.freecol.client.gui.menu and determined three classes contained dead code, InGameMenuBar, MapEditorMenuBar, and DebugMenu. We removed the dead code from the InGameMenuBar class and its relevant dependencies. When the dependencies were removed it cause an error and therefore we had to revert this change and conduct further investigation and ensure only appropriate references were removed.



302 unused classes, 2,554 unused methods, 1,910 unused fields

```
net.sf.freecol.client.gui.action (contains dead code)
~~net.sf.freecol.client.gui.animation~~
net.sf.freecol.client.gui.menu (contains dead code)
    ~~InGameMenuBar.java~~
        ~~InGameMenuBar~~
    FreeColMenuBar.java (contains dead code)
    FreeColMenuTest.java (contains dead code)
    ~~MapEditorMenuBar.java~~
        ~~MapEditorMenuBar~~
    MenuMouseMotionListener.java
    ~~DebugMenu.java~~
net.sf.freecol.client.gui.option (contains dead code)
net.sf.freecol.client.gui.panel (contains dead code)
net.sf.freecol.client.gui.plaf (contains dead code)
net.sf.freecol.client.gui.video (contains dead code)
```

5.2 Resumption Criteria

Testing will resume when the improvement is reverted back to its original state and the activities have been properly documented as lessons learned.

5.2 Approval Criteria

To approve our changes to FreeCol, all existing unit tests must pass and the game must start on each of our development configurations.

6. ENVIRONMENTAL REQUIREMENTS

6.1 Hardware

At least 512MB of RAM and a 1024x768 screen is required to run FreeCol.

6.2 Software

The Java Runtime Environment (JRE) 8 is required to run FreeCol. To develop FreeCol, the Java Development Kit (JDK) 8 and Eclipse are required.

6.3 Security

We will modify a snapshot of FreeCol and host it on GitHub. Any developer can read the source code, but committing requires users to be added as collaborators.

6.4 Tools

We will use the following tools to test FreeCol and make changes:

- Eclipse as our Integrated Development Environment (IDE)

- EclEmma to assess code coverage
- jDeodorant to identify code smells
- Google CodePro Tools to analyze the source code and compute metrics
- FindBugs to identify common mistakes
- Checkstyle to find errors in documentation and style
- PMD to do a source code analysis to find common programming flaws
- CodePro Tools to Find Dead Code and Unnecessary Code Detector to find unnecessary (dead) code
- JDepend to determine the package dependency analysis to ensure if we made a change it did not impact something else

7. DELIVERABLES

We will deliver the following items as test artifacts:

- This Tactical Test Plan in PDF and Microsoft Office formats
- Source code changes committed to our GitHub repository
- Individual peer reviews submitted to Blackboard

8. RESPONSIBILITIES

We worked collaboratively to evaluate the FreeCol program to gain an understanding of the functionality of the program and its classes. We then divided the testing and maintenance activities as follows:

- Robert
    - Identify initial set of improvements
    - Find additional tools and plug-ins for static analysis
    - Perform initial improvements and document in Tactical Test Plan
- Jlaja
    - Identify additional Eclipse plug-ins to utilize to improve code quality
    - Begin Tactical Test Plan sections
    - Perform second iteration of improvements
    - Finish and review Tactical Test Plan