**SER 216 – Summer 2018 B – Group 3**

**List of Bugs**

**Bugs found with FindBugs (Eclipse plug-in)**

*Category: Troubling - Normal Confidence*

1. **Bug Type:** Possible null pointer dereference of listener in edu.asu.stratego.Server.main(String[]) on exception path.

**Explanation**: A reference value which is null on some exception control path is dereferenced here. This may lead to a NullPointerException when the code is executed.

**Where:** Server.java, line 44.

**Possible Fix:** Add an if-clause to check if the reference value is null.

1. **Bug Type:** Unconditional wait in edu.asu.stratego.game.ClientGameManager.playGame(), and similar bugs.

**Explanation**: This method contains a call to java.lang.Object.wait() which is not guarded by conditional control flow. The code should verify that condition it intends to wait for is not already satisfied before calling wait; any previous notifications will be ignored.

**Where:**

* ClientGameManager.java, line 203;
* ClientGameManager.java, line 136;
* BoardTurnIndicator.java, line 66;
* SetupPanel.java, line 240;
* ConnectionScene.java, line 149;
* ConnectionScene.java, line 109;
* ClientGameManager.java, line 136 (wait-not-in-loop variation of the same bug);
* ClientGameManager.java, line 109 (wait-not-in-loop variation of the same bug).

**Possible Fix:** Add an if-clause to check if the condition it intends to wait for is not already satisfied.

*Category: Of Concern - High Confidence*

1. **Bug Type:** Dead store to indicator in new edu.asu.stratego.gui.BoardScene().

**Explanation**: This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error because the value computed is never used.

**Where:** BoardScene.java, line 66.

**Possible Fix:** Eliminate the unused local variable or used it.

1. **Bug Type:** edu.asu.stratego.util.HashTables.PIECE\_MAP is a mutable collection.

**Explanation**: A mutable collection instance is assigned to a final static field, thus can be changed by malicious code or by accident from another package.

**Where:** HashTables.java, line 12.

**Possible Fix:** Since the collection has to be modified to be initialized with the proper values, eliminating the final attribute at its declaration should eliminate the problem.

1. **Bug Type:** edu.asu.stratego.util.HashTables.SOUND\_MAP is a mutable collection.

**Explanation**: A mutable collection instance is assigned to a final static field, thus can be changed by malicious code or by accident from another package.

**Where:** HashTables.java, line 45.

**Possible Fix:** Since the collection has to be modified to be initialized with the proper values, eliminating the final attribute at its declaration should eliminate the problem.

1. **Bug Type:** Write to static field edu.asu.stratego.game.Game.move from instance method new edu.asu.stratego.game.Game(), and similar bugs.

**Explanation**: This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.

**Where:**

* Game.java, line 30;
* Game.java, line 28;
* Game.java, line 27

**Possible Fix:** Make the static variable volatile at its declaration or add a new static constructor for the Game class and use it instead of the classic one.

*Category: Of Concern - Normal Confidence:*

1. **Bug Type:** Integral division result cast to double or float in new edu.asu.stratego.gui.ClientStage().

**Explanation**: This code casts the result of an integral division (e.g., int or long division) operation to double or float. Doing division on integers truncates the result to the integer value closest to zero. The fact that the result was cast to double suggests that this precision should have been retained. What was probably meant was to cast one or both of the operands to double before performing the division. Here is an example:

int x = 2;

int y = 5;

// Wrong: yields result 0.0

double value1 = x / y;

// Right: yields result 0.4

double value2 = x / (double) y;

**Where:** ClienStage.java, line 29.

**Possible Fix:** Cast one or both of the operands to double before performing the division or eliminate the INT cast on line 28.

1. **Bug Type:** Should edu.asu.stratego.game.ClientGameManager$ResetImageVisibility be a \_static\_ inner class? (and similar bugs).

**Explanation**: This class is an inner class but does not use its embedded reference to the object which created it. This reference makes the instances of the class larger and may keep the reference to the creator object alive longer than necessary. If possible, the class should be made static.

**Where:**

* ClientGameManager.java, line 430;
* ClientGameManager.java, line 415;
* BoardSquareEventPane.java, line 86;
* BoardTurnIndicator.java, line 59;
* SetupPanel.java, line 230;
* SetupPiece.java, line 95;
* SetupTimer.java, line 77.

**Possible Fix:** Make the method static when possible.

1. **Bug Type:** Unread field: edu.asu.stratego.game.board.ClientBoard.size; should this field be static? (and similar bugs).

**Explanation**: This class contains an instance final field that is initialized to a compile-time static value. Consider making the field static.

**Where:**

* ClientBoard.java, line 14;
* ServerBoard.java, line 8;
* ConnectionScene.java, line 36;
* ConnectionScene.java, line 35;
* WaitingScene.java, line 17;
* WaitingScene.java, line 16.

**Possible Fix:** Make the field static when possible.

1. **Bug Type:** Write to static field edu.asu.stratego.game.Game.board from instance method new edu.asu.stratego.game.Game(), and similar bugs.

**Explanation**: This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.

**Where:**

* Game.java, line 36;
* Game.java, line 31;
* Game.java, line 33;
* Game.java, line 34;
* BoardTurnIndicator.java, line 30;
* SetupPiece.java, line 48;
* Board Scene.java, line 97;
* Board Scene.java, line 86;
* ClientStage.java, line 28;
* ClientStage.java, line 29.

**Possible Fix:** Make the static variable volatile at its declaration or add a new static constructor for the Game class and use it instead of the classic one.

**Possible enhancements produced by using FindBugs (Eclipse plug-in):**

1. Handling and solving possible bugs;
2. Eliminating unnecessary code and/or variables for a better adherence to the Java coding standards;
3. Creating console messages for the unhandled exceptions of the try-catch blocks within the ServerGameManager.java and the ClientGameManager.java files;
4. Add a catch to the try-catch block in the ServerGameManager.java’s main missing the catch part of the block.