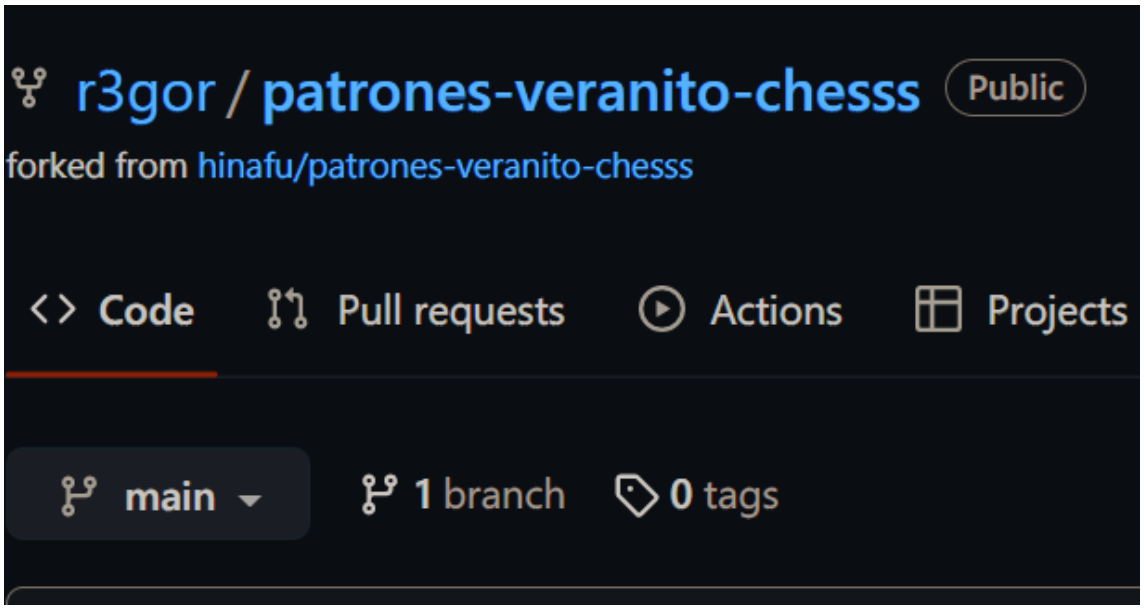
Examen Final – Ramos Paredes Roger Anthony (18200096)

Repositorio: https://github.com/r3gor/patrones-veranito-chesss

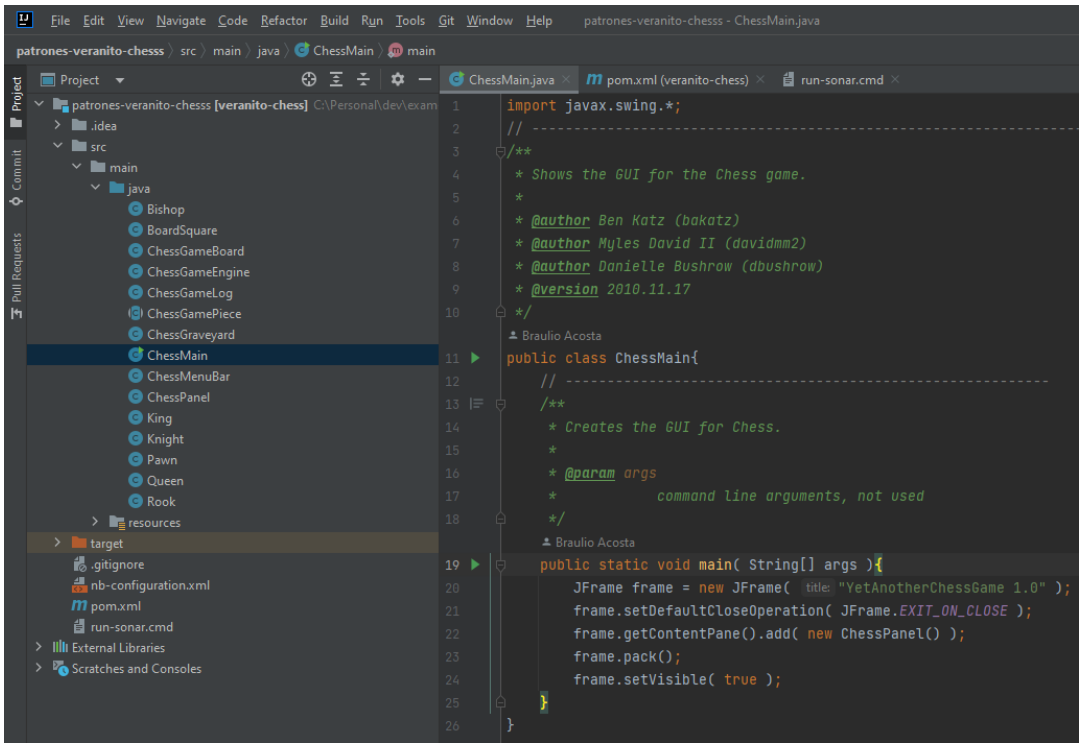
1. Fork al proyecto

Como primer paso hacemos fork al proyecto original para generar una copia en nuestra cuenta personal, en mi caso luego de hacer el fork este es el resultado (mi username en Github es r3gor):



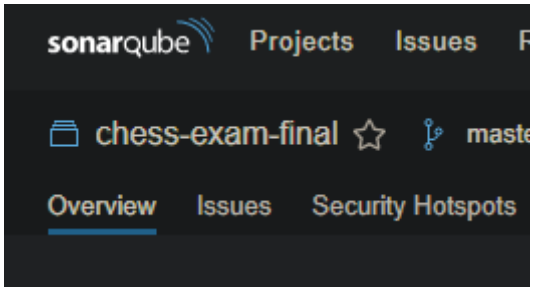
2. Clonación del proyecto

Procedemos a clonar el proyecto localmente para poder trabajar con nuestro editor favorito en este caso InteliJ Idea de Jetbrains, adicionalmente se usó Visual Studio Code ya que provee funcionalidades y pluggins que apoyan en el trabajo de la refactorización. Resultado:



3. Creación del proyecto en SonarQube

Levantamos el servidor de SonarQube en este caso con el script StartSonar que nos aparece en los binarios para Windows cuando descargamos el software. Luego ingresamos con nuestras credenciales y creamos el proyecto. Resultado:



4. SonarQube Análisis - Estado inicial

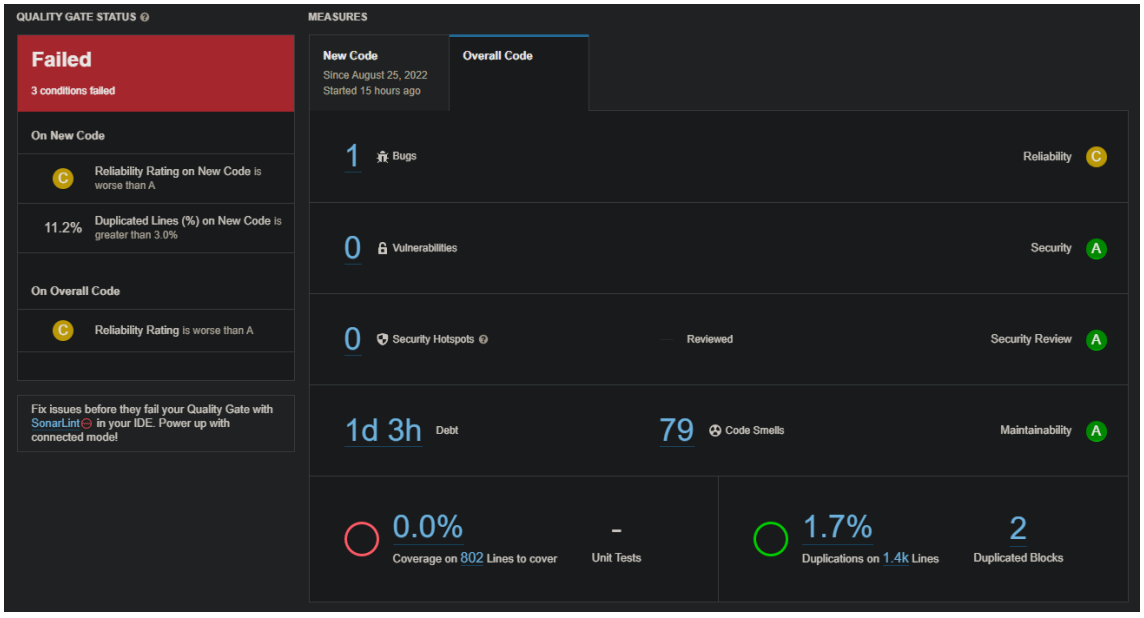
Una vez creado el proyecto procedemos a ejecutar el análisis de SonarQube a nuestro proyecto para esto corremos el comando de Maven que nos provee la interfaz de SonarQube, en este caso es:

```
mvn clean verify sonar:sonar -Dsonar.projectKey=chess-exam-final -
Dsonar.host.url=http://localhost:9000 -
Dsonar.login=sqp_0f829cf747154794fd169c0c2392c763304896a9
```

Resultado:

```
[INFO] Analysis report generated in 185ms, dir size=272.4 kB
[INFO] Analysis report compressed in 767ms, zip size=81.7 kB
[INFO] ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=chess-exam-final
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AYLT0D7la1hELhKW7YzD
[INFO] Analysis total time: 16.128 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  27.343 s
[INFO] Finished at: 2022-08-25T02:03:37-05:00
```

Una vez ejecutado podemos ir a la interfaz de SonarQube para observar los resultados el análisis realizado:

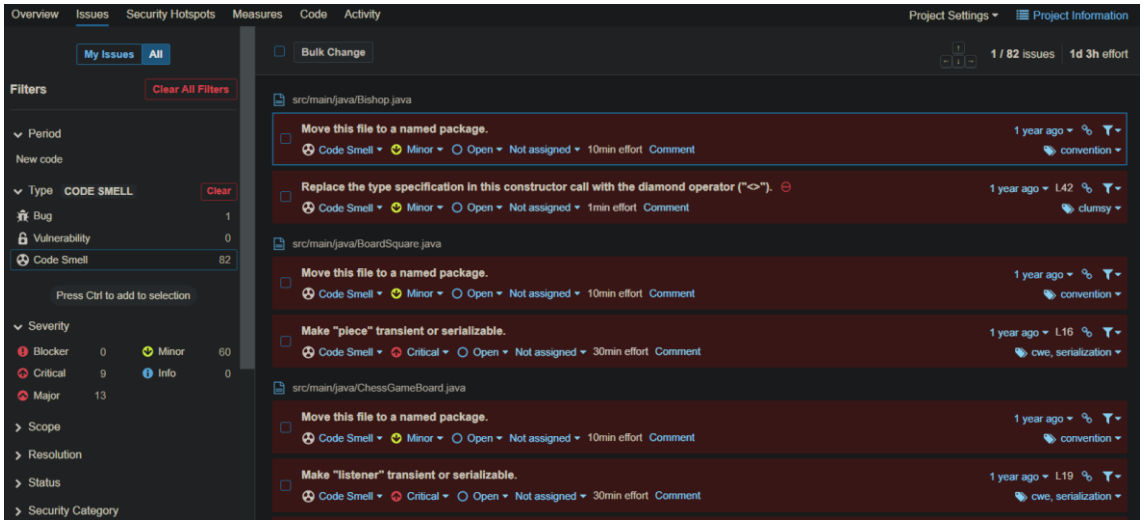


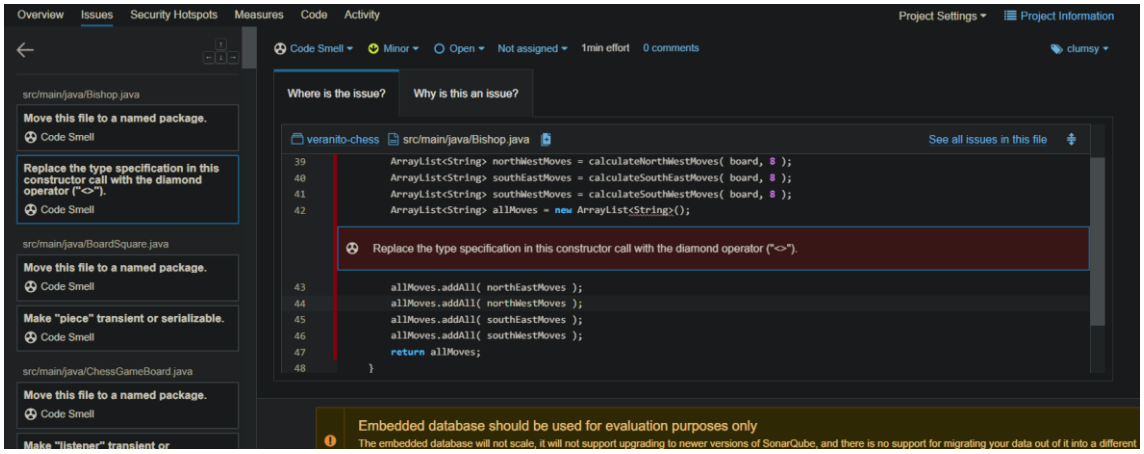
Como resultados de esta pasada inicial con sonar observamos que el test no ha pasado la prueba, tambien podemos observar que existen 79 code smells, 1 bug y 2 bloques duplicados

de código. En los siguientes pasos nos encargaremos de hacer las refactorizaciones necesarias para reducir la máxima cantidad de estos problemas.

5. Refactorización

Podemos navegar en sonar para ver cada detalle por ejemplo de los code smell:

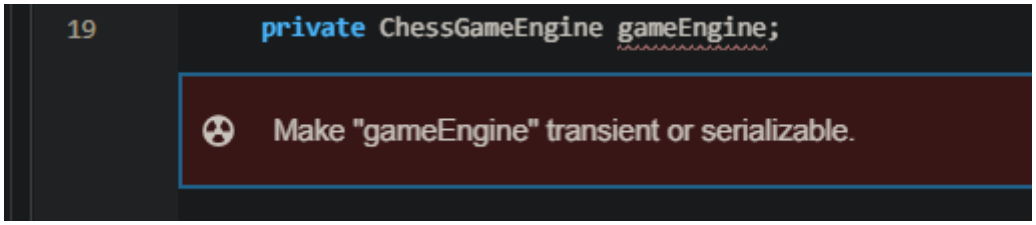




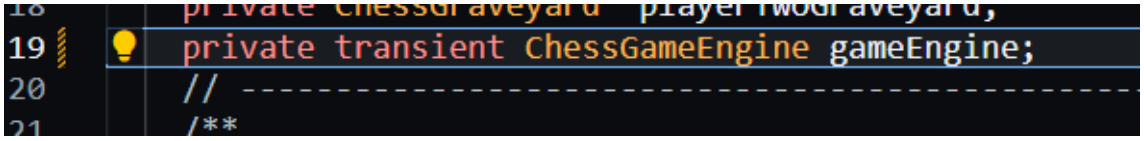
De esta forma podemos navegar en cada una de las observaciones del sonar he ir refactorizando una por una en nuestro editor de texto.

A continuación se muestran algunos ejemplos de la refactorización:

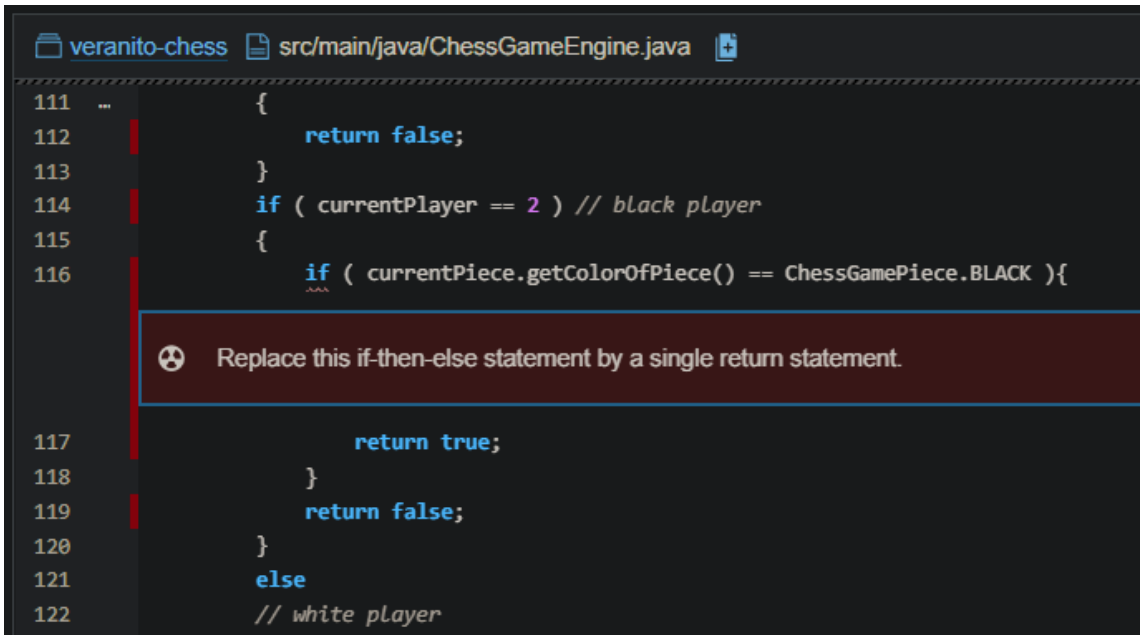
Code smell:



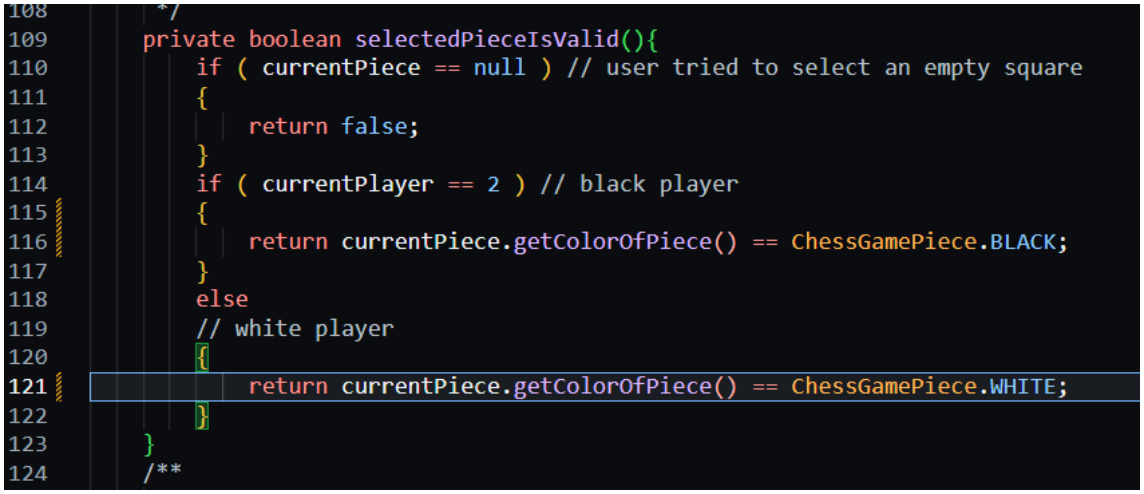
Refactorización:



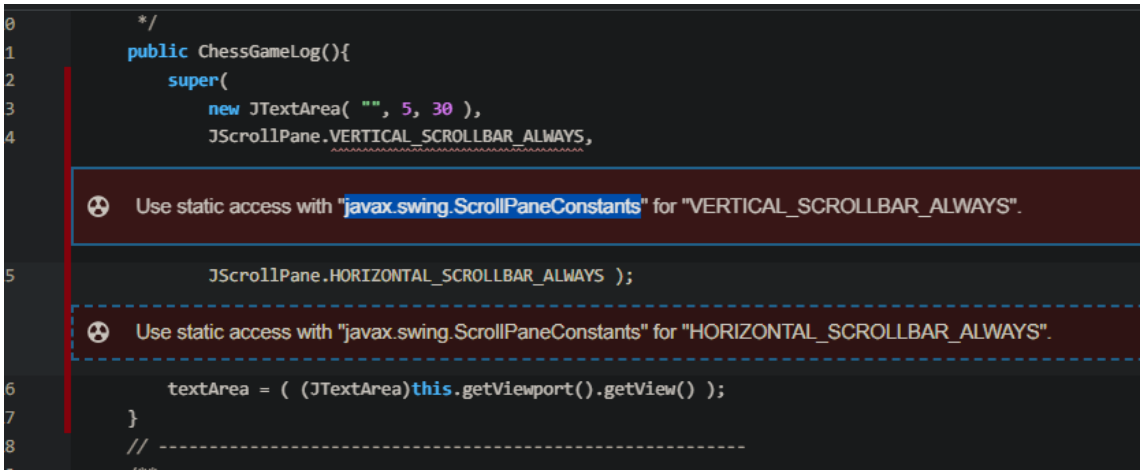
Code Smell:



Refactorización:



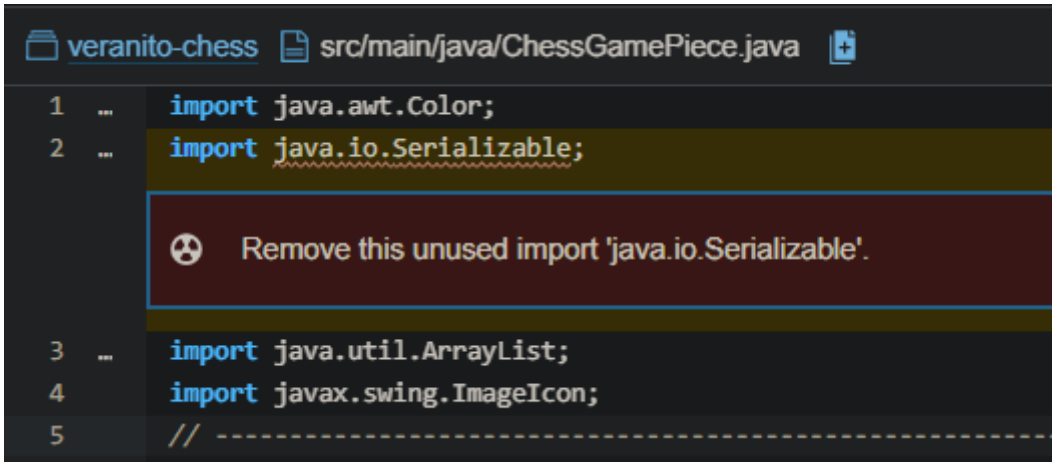
Code smell:



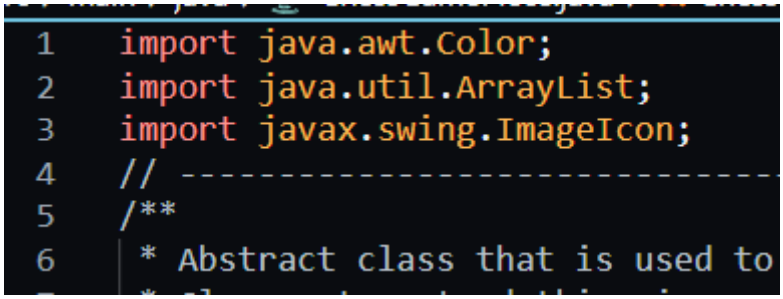
Refactor:

```
20      */
21     public ChessGameLog(){
22         super(
23             new JTextArea( text: "", rows: 5, columns: 30 ),
24             javax.swing.ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
25             javax.swing.ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS );
26         textArea = ( (JTextArea)this.getViewport().getView() );
27     }
28     // ----------------------------------------------------------
29     /**
30      * Adds a new line of text to the log.
```

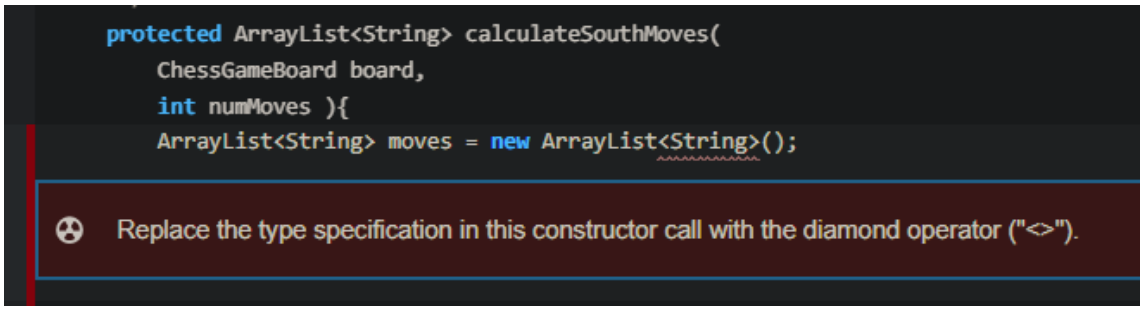
Code smell:



Refactor:



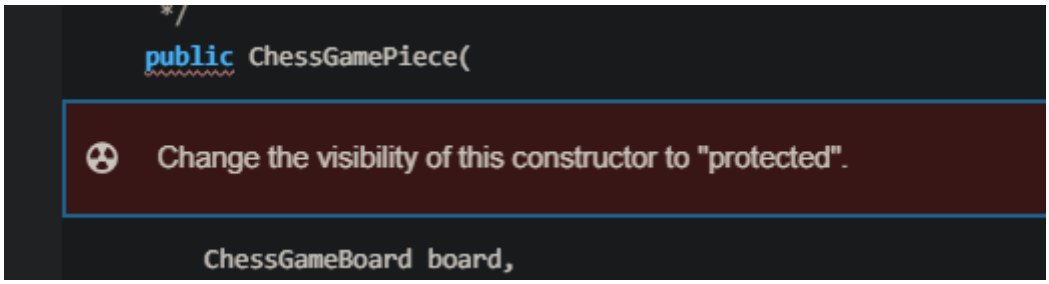
Code Smell:



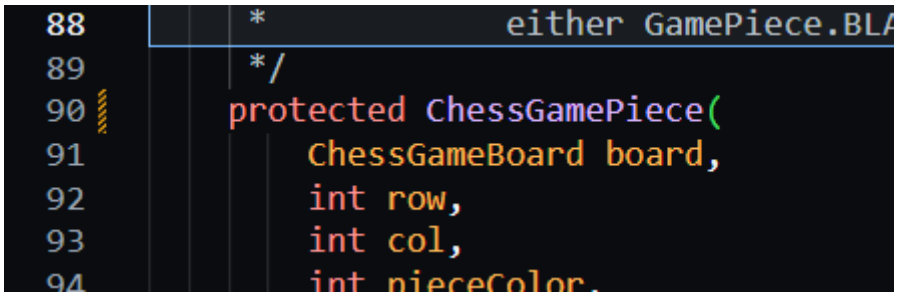
Refactor:

```
31      protected ArrayList<String> calculateSouthMoves(
32          ChessGameBoard board,
33          int numMoves ){
34          ArrayList<String> moves = new ArrayList<>();
35          int count = 0;
```

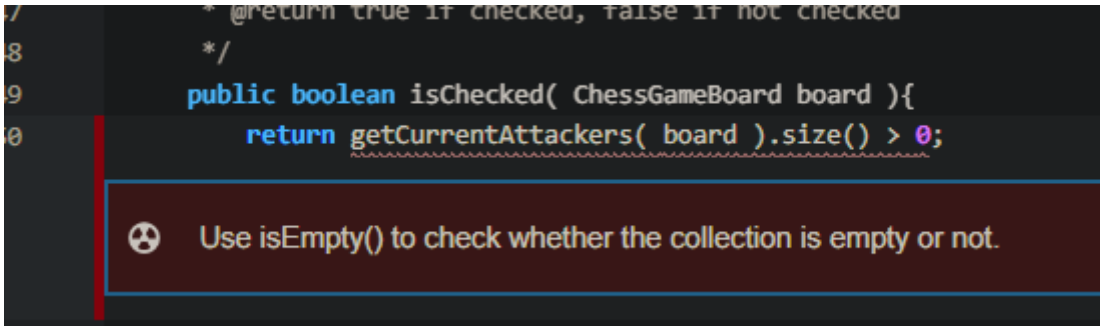
Code Smell:



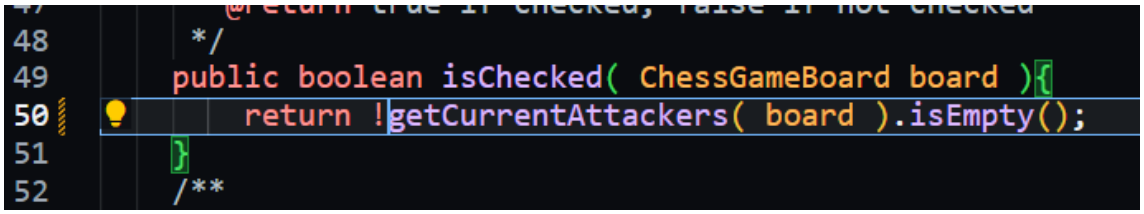
Refactor:



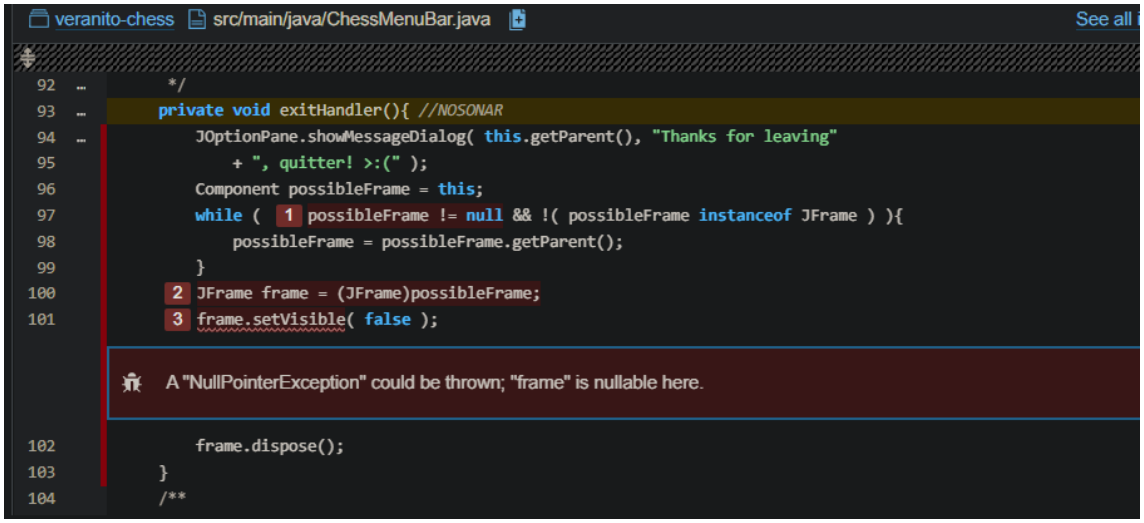
Code Smell:



Refactor:



Bug:



Refactor:

```
        }
        JFrame frame = (JFrame)possibleFrame;
        if(frame != null) {
          frame.setVisible( b: false );
          frame.dispose();
        }
    }
```

Duplicate code:

veranito-chess / src/main/java / King.java

```
38      @Override
39      protected ArrayList<String> calculatePossibleMoves( ChessGameBoard board ){
40          ArrayList<String> northEastMoves = calculateNorthEastMoves( board, 1 );
41          ArrayList<String> northWestMoves = calculateNorthWestMoves( board, 1 );
42          ArrayList<String> southEastMoves = calculateSouthEastMoves( board, 1 );
43          ArrayList<String> southWestMoves = calculateSouthWestMoves( board, 1 );
44          ArrayList<String> northMoves = calculateNorthMoves( board, 1 );
45          ArrayList<String> southMoves = calculateSouthMoves( board, 1 );
46          ArrayList<String> eastMoves = calculateEastMoves( board, 1 );
47          ArrayList<String> westMoves = calculateWestMoves( board, 1 );
```

eranito-chess / src/main/java / Queen.java

```
35      @Override
36      protected ArrayList<String> calculatePossibleMoves( ChessGameBoard board ){
37          ArrayList<String> northEastMoves = calculateNorthEastMoves( board, 8 );
38          ArrayList<String> northWestMoves = calculateNorthWestMoves( board, 8 );
39          ArrayList<String> southEastMoves = calculateSouthEastMoves( board, 8 );
40          ArrayList<String> southWestMoves = calculateSouthWestMoves( board, 8 );
41          ArrayList<String> northMoves = calculateNorthMoves( board, 8 );
42          ArrayList<String> southMoves = calculateSouthMoves( board, 8 );
43          ArrayList<String> eastMoves = calculateEastMoves( board, 8 );
44          ArrayList<String> westMoves = calculateWestMoves( board, 8 );
```

Refactor:

Movemos el código repetitivo a la clase padre para que de esta forma se pueda generalizar según el parámetro que cambia para cada uno de los casos (en uno debe mandase 1 y en el otro 8)

Método general:

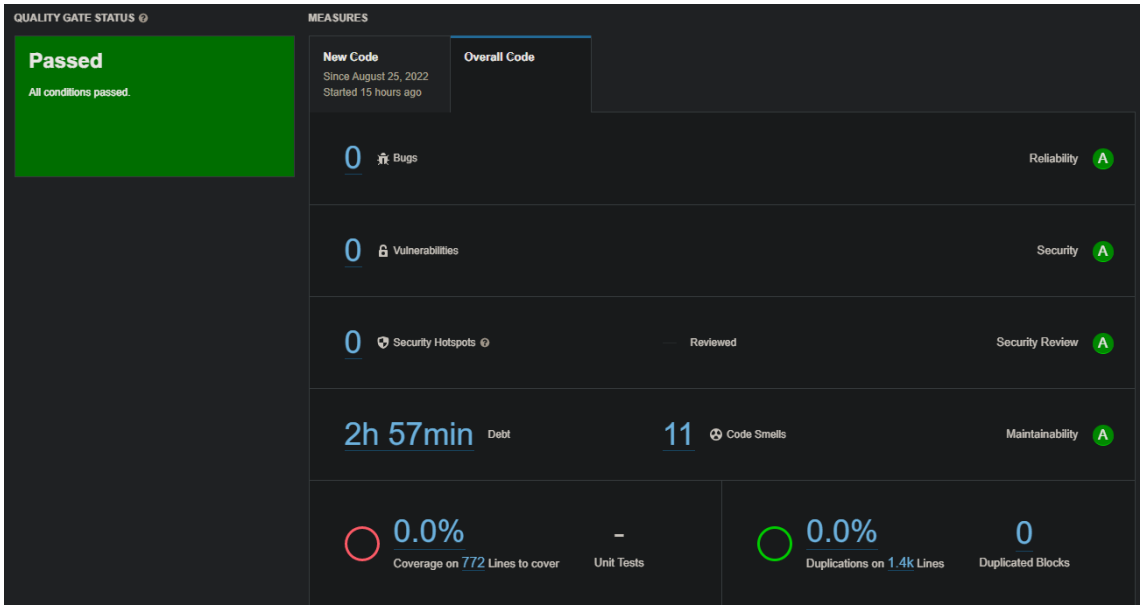
```
48
49  protected ArrayList<String> QueenKingPosibleMoves(ChessGameBoard board, Integer value) {
50      ArrayList<String> northEastMoves = calculateNorthEastMoves( board, value );
51      ArrayList<String> northWestMoves = calculateNorthWestMoves( board, value );
52      ArrayList<String> southEastMoves = calculateSouthEastMoves( board, value );
53      ArrayList<String> southWestMoves = calculateSouthWestMoves( board, value );
54      ArrayList<String> northMoves = calculateNorthMoves( board, value );
55      ArrayList<String> southMoves = calculateSouthMoves( board, value );
56      ArrayList<String> eastMoves = calculateEastMoves( board, value );
57      ArrayList<String> westMoves = calculateWestMoves( board, value );
58      ArrayList<String> allMoves = new ArrayList<>();
59      allMoves.addAll( northEastMoves );
60      allMoves.addAll( northWestMoves );
61      allMoves.addAll( southWestMoves );
62      allMoves.addAll( southEastMoves );
63      allMoves.addAll( northMoves );
64      allMoves.addAll( southMoves );
65      allMoves.addAll( westMoves );
66      allMoves.addAll( eastMoves );
67      return allMoves;
68  }
69  /**
```

Llamadas especificas para King y Queen:

```java
@Override
protected ArrayList<String> calculatePossibleMoves( ChessGameBoard board ){
  return this.QueenKingPosibleMoves(board, value: 1);
}
/**
```

```java
@Override
protected ArrayList<String> calculatePossibleMoves( ChessGameBoard board ){
    return this.QueenKingPosibleMoves(board, value: 8);
}
```

6. Resultados de la refactorización – Pasada final SonarQube



Tanto los duplicated blocks como los bugs se redujeron a cero, mientras que los code smells se redujeron a una cantidad de 11, además el estado ahora es de "Passed". Por lo tanto se concluye que el código tiene una mejor calidad ahora que antes de la refactorización.