

## Static Code Analysis Exercise

For this exercise, you will run SonarQube to analyze your Java project code and SonarScanner to analyze your TypeScript code. There is a deliverable due by end of class and another for Sprint 4.

With the generated reports, you will put together a plan to address issues that were flagged in your Java or TypeScript code.

**Note:** Make SonarQube is **running**, which you should have verified in the setup document.

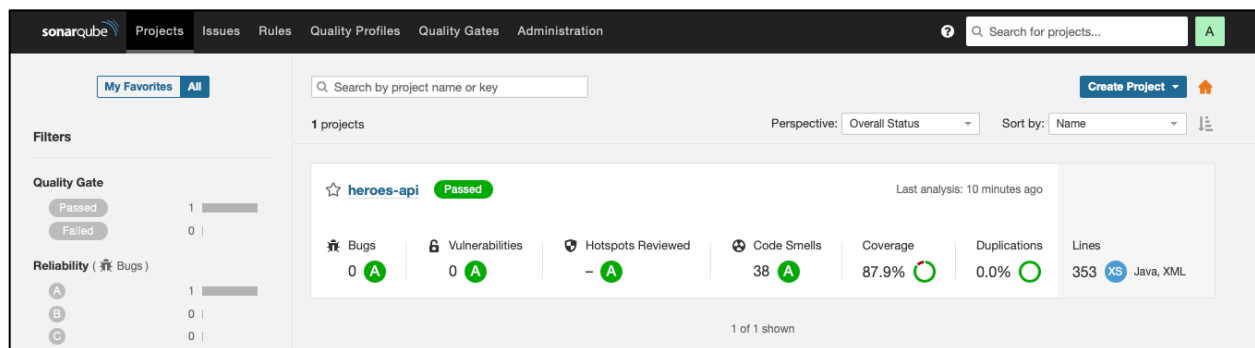
### Analyzing code with SonarQube

We will use SonarQube to analyze our Java code

1. Go to your project directory where you would run maven to build and test your code
2. Run `mvn clean test sonar:sonar -D sonar.login=admin -D sonar.password=admin password`
3. This will execute all unit tests followed by SonarQube. You should see something similar to below:

```
[INFO] Analysis report generated in 68ms, dir size=173.1 kB
[INFO] Analysis report compressed in 68ms, zip size=43.3 kB
[INFO] Analysis report uploaded in 73ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=com.heroes.api%3Aheroes-api
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AX3JowUyfDrwyKT3wXX9
[INFO] Analysis total time: 5.940 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15.774 s
[INFO] Finished at: 2021-12-17T13:22:17-05:00
[INFO] -----
```

4. Log in to <http://localhost:9000> and you should see your SonarQube report:



The screenshot shows the SonarQube web interface. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is present on the right. The main content area shows a list of projects, with 'heroes-api' selected and marked as 'Passed'. The dashboard for 'heroes-api' displays the following metrics:

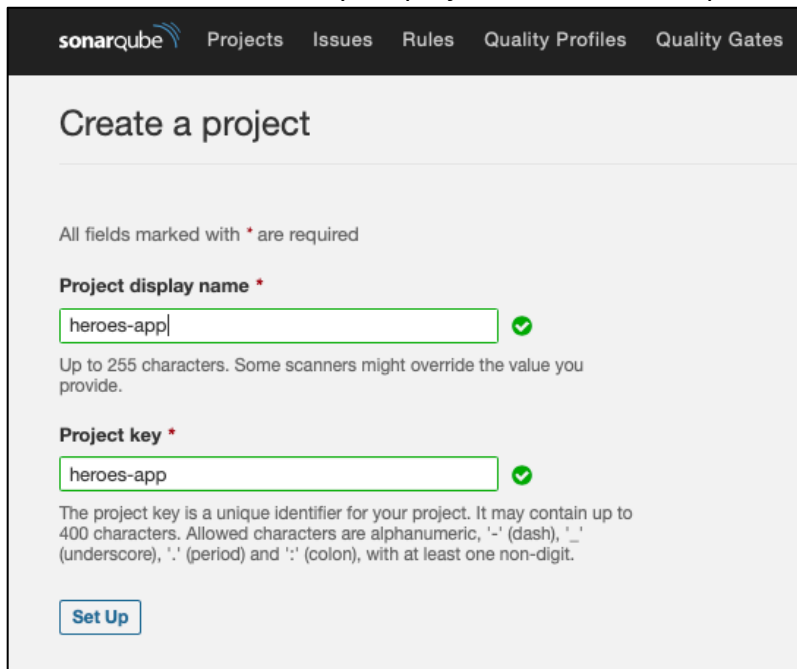
Metric	Value	Status
Bugs	0	A
Vulnerabilities	0	A
Hotspots Reviewed	-	A
Code Smells	38	A
Coverage	87.9%	C
Duplications	0.0%	C
Lines	353	XS

The 'Lines' metric is further detailed as '353 XS Java, XML'. The interface also shows '1 projects' in the list and '1 of 1 shown' at the bottom.

## Running SonarScanner

Next, we will run SonarScanner to analyze our TypeScript code

1. In SonarQube, click the “Create Project” button in the upper right corner and select “Manually”
2. Enter a name for your project and click “Set Up”



The screenshot shows the SonarQube interface for creating a project. The top navigation bar includes 'sonarqube', 'Projects', 'Issues', 'Rules', 'Quality Profiles', and 'Quality Gates'. The main heading is 'Create a project'. Below this, a note states 'All fields marked with \* are required'. There are two input fields: 'Project display name \*' and 'Project key \*'. Both fields contain the text 'heroes-app' and have a green checkmark to their right. Below the 'Project display name' field, there is a note: 'Up to 255 characters. Some scanners might override the value you provide.' Below the 'Project key' field, there is a note: 'The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '\_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.' At the bottom left of the form is a blue button labeled 'Set Up'.

5. Select “Locally”



The screenshot shows a dialog box with the text 'Are you just testing or have an advanced use-case? Analyze your project locally.' Below this text is a white box containing a blue icon of a hand pointing down to a computer monitor, with three arrows pointing down from the hand. Below the icon is the word 'Locally' in blue text.

6. For **#1 Provide a Token**, Enter any name for your token and click “Generate”

**1** Provide a token

Generate a token

Use existing token

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#).

7. Click “Continue”

8. For **#2 Run analysis on your project**, select Other (for JS, TS, Go, Python, PHP, ...)

9. Select your OS

10. You should see something similar to below:

**2** Run analysis on your project

What option best describes your build?

What is your OS?

Download and unzip the Scanner for macOS

Visit the [official documentation of the Scanner](#) to download the latest version, and add the `bin` directory to the `PATH` environment variable

**Execute the Scanner**

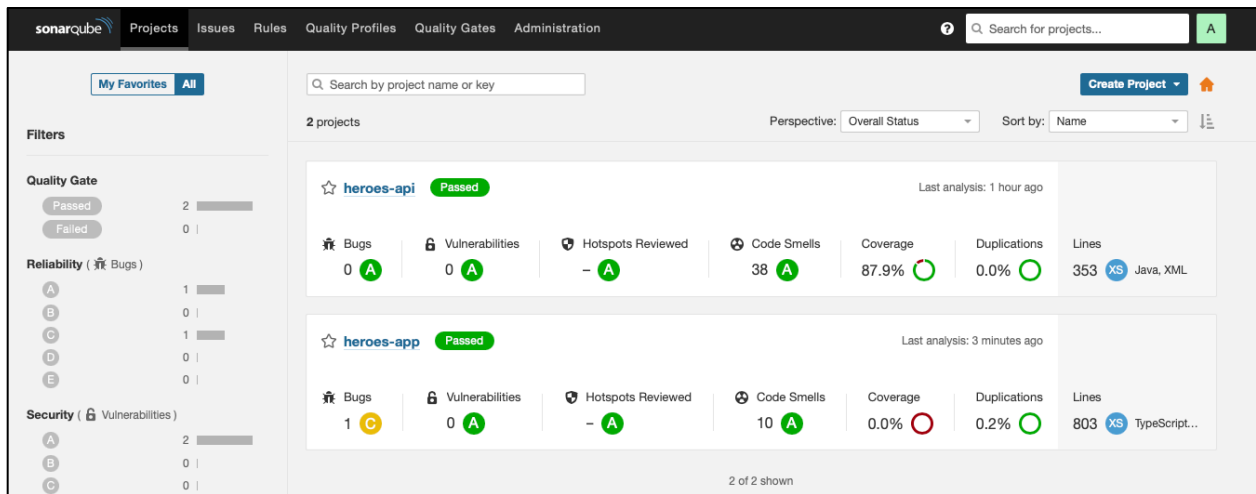
Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner \  
-Dsonar.projectKey=heroes-app \  
-Dsonar.sources=. \  
-Dsonar.host.url=http://localhost:9000 \  
-Dsonar.login=5d40923f41d923f31a0a084694a73e1123d52a9b
```

11. Copy the command under “Execute the Scanner” and go to the directory where your TypeScript code resides and paste this to run. You should see something similar to below:

```
INFO: CPD Executor 9 files had no CPD blocks  
INFO: CPD Executor Calculating CPD for 24 files  
INFO: CPD Executor CPD calculation finished (done) | time=19ms  
INFO: Analysis report generated in 93ms, dir size=1.2 MB  
INFO: Analysis report compressed in 197ms, zip size=317.4 kB  
INFO: Analysis report uploaded in 181ms  
INFO: ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=heroes-app  
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report  
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AX3J217VfDrwyKT3wXYQ  
INFO: Analysis total time: 42.204 s  
INFO: -----  
INFO: EXECUTION SUCCESS  
INFO: -----  
INFO: Total time: 43.154s  
INFO: Final Memory: 14M/57M  
INFO: -----
```

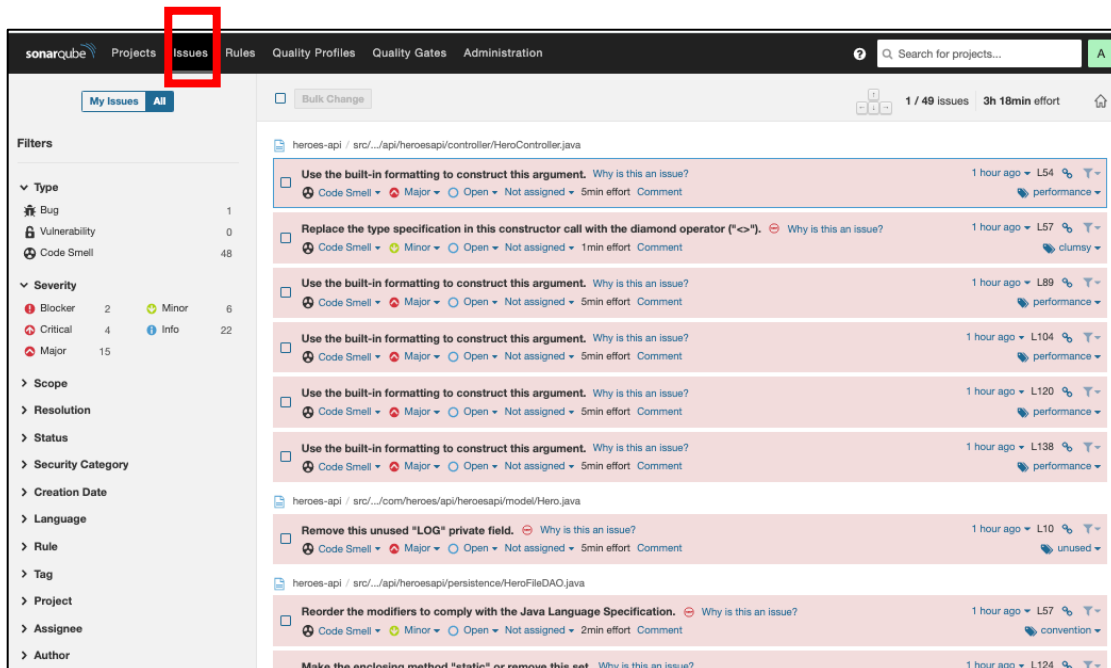
12. Go back to the SonarQube web page and you should see both your projects listed.



Take a screen shot of the projects (similar to above) and deposit it in the *Static Code Analysis - individual* in the myCourses Assignments by the date shown on your section's schedule.

## Explore and Analyze Your Reports

Every report will be different based on what the code analysis identified. Clicking on “Issues” (top of screen – see below) will list all the issues across your projects, which can then be categorized by severity (e.g. major, critical, blocker) and type (e.g. bug, vulnerability).



Depending on the complexity of your code, certain metrics like **Cognitive Complexity** might get triggered, requiring attention for potentially refactoring:

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Refactor this method to reduce its Cognitive Complexity from 20 to the 15 allowed. Why is this an issue? 2 years ago L434

Code Smell Critical Open Not assigned 10min effort Comment brain-overload

435 **1** if(getPieceByPosition(currPos) == null) {  
 436     return false;  
 437 }  
 438  
 439 3197... **2** if(getPieceByPosition(currPos).getType() == Type.SINGLE) {  
 440     **3** if (pColor == Color.RED) {  
 441         **4** if (isJumpNE(currPos, pColor) **5** || isJumpNW(currPos, pColor)) {  
 442             return true;  
 443         **6** else {  
 444             return false;  
 445         }  
 446         **7** else {  
 447             **8** if (isJumpSE(currPos, pColor) **9** || isJumpSW(currPos, pColor)) {  
 448                 return true;  
 449             **10** else {  
 450                 return false;  
 451             }  
 452         }  
 453     }  
 454 }  
 455 3197... **11** else {  
 456     **12** if (isJumpNE(currPos, pColor) **13** || isJumpNW(currPos, pColor) || isJumpSE(currPos, pColor) ||  
 457     isJumpSW(currPos, pColor)) {  
 458         return true;  
 459     } **14** else {  
 460         return false;  
 461     }  
 462 }  
 463 }  
 464 }  
 465 }  
 466 }

Refactor this method to reduce its Cognitive Complexity from 20 to the 15 allowed.  
 Code Smell +14

1 +1  
 2 +1  
 3 +2 (incl 1 for nesting)  
 4 +3 (incl 2 for nesting)  
 5 +1  
 6 +1  
 7 +1  
 8 +3 (incl 2 for nesting)  
 9 +1  
 10 +1  
 11 +1  
 12 +2 (incl 1 for nesting)  
 13 +1  
 14 +1

...src/main/java/com/webcheckers/model/Co  
 Rename this constant name to match the regular expression `^[A-Z][A-Z0-9]*([A-Z0-9]+)?$`.  
 Code Smell

Other “code smells” may get flagged due to issues with readability or other factors. This may indicate a problem or possibly a false flag.

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

Remove this unused “jumped” private field. Code Smell

Replace this if-then-else statement by a single return statement. Code Smell

This block of commented-out lines of code should be removed. Code Smell

Replace this if-then-else statement by a single return statement. Code Smell

Immediately return this expression instead of assigning it to the temporary variable “the\_space”. Code Smell

Rename this local variable to match the regular expression `^[a-z][a-zA-Z0-9]*$`. Code Smell

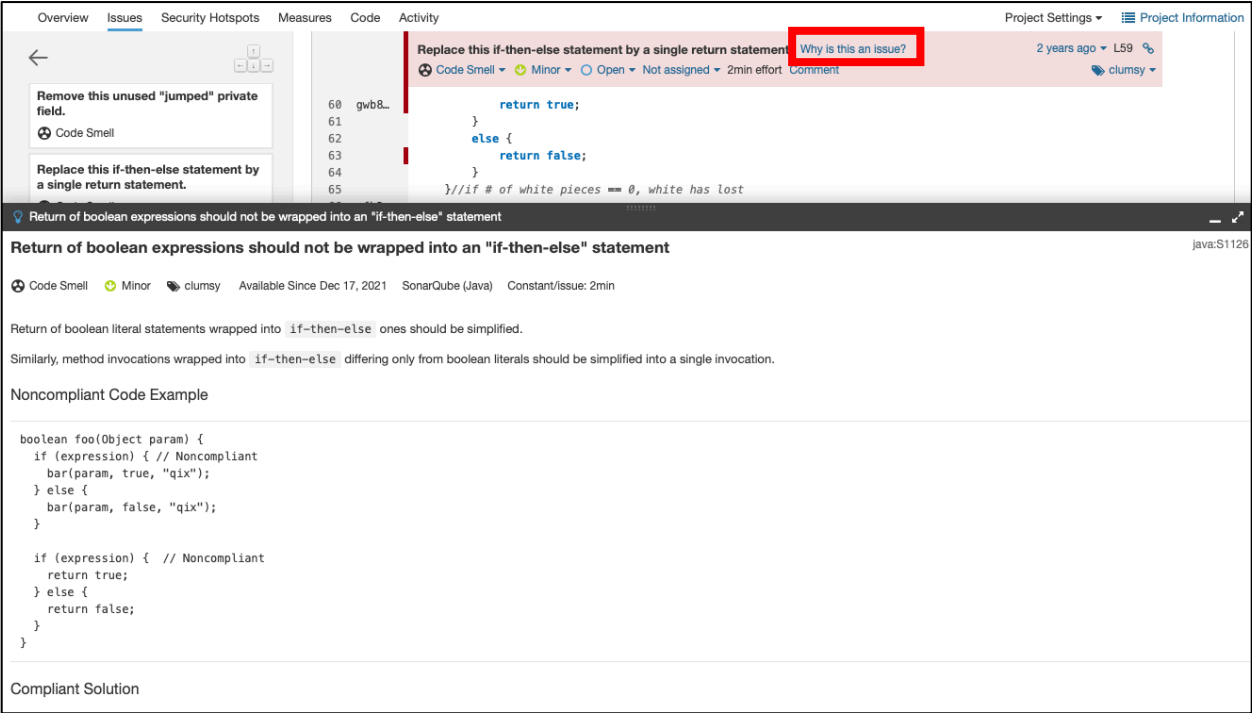
44 }  
 45 gwb8... **else if** (!this.redMove && !this.redJump) {  
 46     return true;  
 47 }  
 48     else {  
 49         return false;  
 50     }  
 51     //return (this.redpieces == 0);  
 52  
 53 } //if # of red pieces == 0, red has lost  
 54  
 55 public boolean whiteLost() {  
 56     if(this.whitepieces == 0) {  
 57         return true;  
 58     }  
 59 gwb8... **else if** (!this.whiteMove && !this.whiteJump) {  
 60     return true;  
 61 }  
 62 }

Replace this if-then-else statement by a single return statement. Why is this an issue? 2 years ago L45  
 Code Smell Minor Open Not assigned 2min effort Comment clumsy

This block of commented-out lines of code should be removed. Why is this an issue? 2 years ago L51  
 Code Smell Major Open Not assigned 5min effort Comment unused

Replace this if-then-else statement by a single return statement. Why is this an issue? 2 years ago L59  
 Code Smell Minor Open Not assigned 2min effort Comment clumsy

If a particular issue is not clear, click the “Why is this an issue?”, which will provide a description of the issue including code examples of a non-compliant issue and a compliant solution.



### Project Sprint 4 Deliverable

Identify 3-4 areas within your code that have been flagged by SonarQube and provide your analysis and recommendations. Include any relevant screenshot(s) with each area. This will be part of your final design documentation in your Sprint 4 submission. Be sure to include at least one from both reports (Java and TypeScript).

### Stopping SonarQube

Refer to the setup document to stop SonarQube.