Assignment: Static code analysis using SonarQube

SonarQube is the static code analysis tool of choice. It is open source and can be integrated in your Gitlab environment.

Difficulty: ☆☆☆★★

Learning objectives:

- Installing SonarQube
- Applying SonarQube to your project
- Analysing SonarQube results
- Implementing SonarQube in your CI solution

Estimated time required: 100 minutes

Step 1: Download and install SonarQube

Get the SonarQube Community Edition: https://www.sonarqube.org/downloads/ and install it. Check out https://docs.sonarqube.org/latest/setup/get-started-2-minutes/ and choose from the Zip file (unless you already have a bit of Docker experience). When you have SonarQube running (StartSonar.bat) try to access it by going to: http://localhost:9000 If you run into trouble check the next step.

Step 2: Check if it works

There is a high chance of things not working in your first try. Some solutions:

- SonarQube can be run from the console using Administrator mode. This can remove issues.
- The port SQ uses can be used by something else. Open sonar.properties in the conf folder to change the default port.

```
104  # Web context. When set, it must start with forward slash (for exa
105  # The default value is root context (empty value).
106  #sonar.web.context=
107  # TCP port for incoming HTTP connections. Default value is 9000.
108  sonar.web.port=9002
109
```

- You might get an Elasticsearch error. This also is a port issue. Check out sonar.properties in the conf folder and change the port to 0 (let elasticsearch auto choose a port).

```
273  # Elasticsearch port. Defaul
274  # As a security precaution,
275  sonar.search.port=0
276
```

Step 3: check out the online SonarQube environment

Login to SonarQube using the default username/password: admin/admin. Make sure you create a new token to be used in the connection between your back-end and SonarQube.

Step 4: add SonarQube as part of your gradle.properties

Check out https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-gradle/ to see how to add SonarQube to your project. Take special notice of the gradle.properties. Add this file in your

project root if not there. It could look something like:

```
# possible location for your java home folder
org.gradle.java.home=C:\\Program Files\\Java\\jdk-11.0.8

# gradle.properties changes for sonarqube
systemProp.sonar.host.url=http://localhost:9000

#----- Token generated from an account with 'publish analysis' permission
systemProp.sonar.login=50a1981d1af6a63be196963720b452f7f9f0bc6c
```

Also change your gradle.build file to include SonarQube. At least the plugin must be included for SonarQube to work:

```
id 'org.sonarqube' version '3.0'
```

Sometimes you also need to edit the wrapper.conf file in the SonarQube folder to include the correct path to your JVM:

```
1  # Path to JVM executable. By default it must be available in PATH.
2  # Can be an absolute path, for exam[redacted]:
3  #wrapper.java.command=/path/to/my/jdk/bin/java
4  wrapper.java.command=java
5
```

Step 5: analyse the results

Run the SonarQube analysis using your commandline with ./gradlew sonarqube. It should run the SonarQube analysis. When completed, go to http://localhost:9000 (or a different port depending on settings) and check out the results. Check how to improve your code, starting with critical bugs, code smells and vulnerabilities, and work your way down to the minor things. Decide and reason if it is useful to spent time fixing these.

Final step: Implement SonarQube in your CI solution

When it is all working fine add SonarQube as a new stage to your gitlab-ci file.

When completed

Discuss the results with your teacher, and make sure every commit and push SonarQube runs and checks your code, so you can be assured of a better-quality product.