Code Review

++++++++++++

-> In project development multiple team members will be involved

-> Some developers will be freshers, some are junior developers & some are senior developers

-> As a developer me might do some mistakes in coding or we may not follow proper coding standards

-> We will perform Code Review to identify the mistakes of the developers in coding

-> With the help of code review we can provide quality code and bug free code

-> To perform code review we will use Sonar Qube software

Sonar Qube

-> Sonar Qube is an automatic code review tool

-> Sonar Qube supports for 29 programming languages

-> Sonar Qube will identify

1) Bugs
2) Vulnerabilities
3) Code Smells
4) Duplicate Code Blocks

Note: SonarQube will not check our logic is correct not

```
public void findNonRepeatedCharInString(String str){

   //logic to find
}
```

Note: Juniour developers code checking will done by seniour developers in the team. This is called as Peer Review. Peer Review is a manual process. In peer review, logic checking will be done.

Installing Sonar Software in Local

-> Download Sonar Software from below url

URL : https://www.sonarqube.org/downloads/

Version : 6.3.1 (historical version download)

-> Start Sonar Server by executing StartSonar.bat

Location : Sonar-Folder/bin/windows64/StartSonar.bat

-> Once Sonar Server is started it will display Sonar up and running message in console.

-> By Default Sonar Server will run on 9000 port number (We can customize port number)

Location : sonar-folder/conf/sonar.properties file

-> Open Sonar Server Dashboard using below URL

URL : http://localhost:9000/

Running Project with SonarQube Server

-> Add below 1 plugin in project pom.xml file (in <build> tag)

```
<plugin>
  <groupId>org.sonarsource.scanner.maven</groupId>
  <artifactId>sonar-maven-plugin</artifactId>
  <version>3.4.0.905</version>
</plugin>
```

-> Do Maven build of project with package goal

```
mvn clean package
```

-> For project do maven build with below goal To Do Code Review

```
mvn sonar:sonar
```

-> After maven build completed, check sonar server dashboard.

Lessons Learnt in Code Review

----

1) String which we want to compare should present at left side

```
if (userAcc.getAccStatus().equals("LOCKED"))  // bad-practise

if ("LOCKED".equals(userAcc.getAccStatus()))  // good practise
```

2) Replace StringBuffer with StringBuilder to improvate performance

StringBuffer -> is synchronized (only one thread can access at a time)

StringBuilder -> is not-syncronized (multiple threads can access at a time)

3) Either log or re-throw the exception

```
//bad practise
catch (Exception e) {
        e.printStackTrace( );
}

//good practise
catch (Exception e) {
        logger.error("Exception :: "+e.getMessage(), e);
}
```

4) Instead of Math.random( ) use java.util.Random.nextInt ( ) method to generate random number

6) Don't use "password" or "pwd" in our code directley. It will be considered as vulnerable. Use "pazzword" or "pzzwd" for variable names.

7) Declare variables before constructor

8) Remove un-necessary curly braces from lambda when we have single line

9) follow camel case for variables names declaration

10) Declare private constructor for class if it is not getting instantiated anywhere.

```
mvn sonar:sonar -Dsonar.login=YOUR_USERNAME -Dsonar.password=YOUR_PASSWORD
```

```
mvn sonar:sonar -Dsonar.login=admin -Dsonar.password=password
```

##
###

Code Review

++++++++++++

-> In project development multiple team members will be involved

-> Some developers will be freshers, some are junior developers & some are senior developers

-> As a developer me might do some mistakes in coding or we may not follow proper coding standards

-> We will perform Code Review to identify the mistakes of the developers in coding

-> With the help of code review we can provide quality code and bug free code

-> To perform code review we will use Sonar Qube software

Sonar Qube

-> Sonar Qube is an automatic code review tool

-> Sonar Qube supports for 29 programming languages

-> Sonar Qube will identify

1) Bugs
2) Vulnerabilities
3) Code Smells
4) Duplicate Code Blocks

Note: SonarQube will not check our logic is correct not

```
public void findNonRepeatedCharInString(String str){

   //logic to find
}
```

Note: Juniour developers code checking will done by seniour developers in the team. This is called as Peer Review. Peer Review is a manual process. In peer review, logic checking will be done.

Installing Sonar Software in Local

-> Download Sonar Software from below url

URL : https://www.sonarqube.org/downloads/

Version : 6.3.1 (historical version download)

-> Start Sonar Server by executing StartSonar.bat

Location : Sonar-Folder/bin/windows64/StartSonar.bat

-> Once Sonar Server is started it will display Sonar up and running message in console.

-> By Default Sonar Server will run on 9000 port number (We can customize port number)

Location : sonar-folder/conf/sonar.properties file

-> Open Sonar Server Dashboard using below URL

URL : http://localhost:9000/

Running Project with SonarQube Server

-> Add below 1 plugin in project pom.xml file (in <build> tag)

```
<plugin>
  <groupId>org.sonarsource.scanner.maven</groupId>
  <artifactId>sonar-maven-plugin</artifactId>
  <version>3.4.0.905</version>
</plugin>
```

-> Do Maven build of project with package goal

```
mvn clean package
```

-> For project do maven build with below goal To Do Code Review

```
mvn sonar:sonar
```

-> After maven build completed, check sonar server dashboard.

Lessons Learnt in Code Review

----

1) String which we want to compare should present at left side

```
if (userAcc.getAccStatus().equals("LOCKED"))  // bad-practise

if ("LOCKED".equals(userAcc.getAccStatus()))  // good practise
```

2) Replace StringBuffer with StringBuilder to improvate performance

StringBuffer -> is synchronized (only one thread can access at a time)

StringBuilder -> is not-syncronized (multiple threads can access at a time)

3) Either log or re-throw the exception

```
//bad practise
catch (Exception e) {
        e.printStackTrace( );
}

//good practise
catch (Exception e) {
        logger.error("Exception :: "+e.getMessage(), e);
}
```

4) Instead of Math.random( ) use java.util.Random.nextInt ( ) method to generate random number

6) Don't use "password" or "pwd" in our code directley. It will be considered as vulnerable. Use "pazzword" or "pzzwd" for variable names.

7) Declare variables before constructor

8) Remove un-necessary curly braces from lambda when we have single line

9) follow camel case for variables names declaration

10) Declare private constructor for class if it is not getting instantiated anywhere.