

ASSIGNMENT 4 (FINAL)

REFACTORING & WRAP UP

CSE2115 Software Engineering Methods
Team 90

Made by:
Victor Roest
Tim Numan
Maikel Houbaer
Jesse Nieland
Eduard Filip

What and why?

We evaluated two code metric tools: MetricsReloaded and Sonarqube, we ended up choosing Sonarqube over MetricsReloaded because Sonarqube gave much more verbose and useful information, such as tips on how to solve and graphs to show our progress. All screenshots used in the rest of this document are therefore screenshots from the Sonarqube webgui.

Method level

```
@Override
public boolean touchUp(int screenX, int screenY, int pointer, int button) {
    if (button == Input.Buttons.LEFT) {
        if (cue.isVisible()) {
            Vector3 ballPos =
                stage.getCamera().project(new Vector3(cueBall.getBody().getPosition(), 0));
            Vector2 direction = new Vector2(ballPos.x, ballPos.y)
                .cpy().sub(new Vector2(screenX, Gdx.graphics.getHeight() - screenY));
```

Cast one of the operands of this subtraction operation to a "float". See Rule

2 months ago ▾ L200 🔗

🔒 Bug ▾ ⚡ Minor ▾ 🔵 Open ▾ Not assigned ▾ 5min effort Comment

🔗 cert, cwe, overflow, sans-top25-risky ▾

```
        cue.shoot(cueBall, direction);
```

```
public boolean touchDragged(int screenX, int screenY, int pointer) {
    if (cue.isVisible()) {
        Vector2 position = new Vector2(screenX, Gdx.graphics.getHeight() - screenY);
```

Cast one of the operands of this subtraction operation to a "float". See Rule

last month ▾ L219 🔗

🔒 Bug ▾ ⚡ Minor ▾ 🔵 Open ▾ Not assigned ▾ 5min effort Comment

🔗 cert, cwe, overflow, sans-top25-risky ▾

```
        Vector2 ballPos =
```

2 Times a possible overflow: Fixed by casting to float.

```
private final transient AtomicLong counter = new AtomicLong();
```

```
@RequestMapping("/greeting")
```

Add a "method" parameter to this "@RequestMapping" annotation. See Rule

2 months ago ▾ L28 🔗

🔒 Vulnerability ▾ 🚫 Blocker ▾ 🔵 Open ▾ Not assigned ▾ 5min effort Comment

🔗 cwe, owasp-a6, sans-top25-insecure, s... ▾

```
public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {
    return new Greeting(counter.incrementAndGet(),
        String.format(template, name));
}
```

"RequestMapping" is unnecessarily vague and poses a security risk, changed to "GetMapping" to fix.

```
 */
private Vector2 determineNumberBallPosition(int x, int y) {
    float posX = x * (float) Math.sqrt(3f) * this.RADIUS;

    float posY = x * this.RADIUS - 2 * y * this.RADIUS;
}
```

Use static access with "nl.tudelft.cse.sem.pool.creators.BallPlacer" for "RADIUS". See Rule 8 days ago L86 [🔗](#)
🚫 Code Smell 🚫 Critical 🔓 Open 📌 Not assigned ⏱ 5min effort 💬 Comment 🧐 confusing

Use static access with "nl.tudelft.cse.sem.pool.creators.BallPlacer" for "RADIUS". See Rule 8 days ago L87 [🔗](#)
🚫 Code Smell 🚫 Critical 🔓 Open 📌 Not assigned ⏱ 5min effort 💬 Comment 🧐 confusing

Use static access with "nl.tudelft.cse.sem.pool.creators.BallPlacer" for "RADIUS". See Rule 8 days ago L87 [🔗](#)
🚫 Code Smell 🚫 Critical 🔓 Open 📌 Not assigned ⏱ 5min effort 💬 Comment 🧐 confusing

Changed access to a static as suggested.

```

*/
private List<Integer> createRandomOrder() {
    List<Integer> ballOrder = new ArrayList<Integer>();
}

```

Replace the type specification in this constructor call with the diamond operator ("<>"). See Rule

13 days ago ▾ L67 🔗

🔒 Code Smell ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 1min effort Comment

🐞 clumsy ▾

```

for (int i = 1; i <= 15; i++) {

```

The "Integer" specifier was just unneeded, removed to fix the complaint.

```

User parsedOutput = new ObjectMapper().readValue(output, User.class);

Assert.assertEquals(parsedOutput.getUsername(), testUser.getUsername());
Assert.assertEquals(parsedOutput.getPassword(), "");

```

Swap these 2 arguments so they are in the correct order: expected value, actual value. See Rule

2 months ago ▾ L62 🔗

1

🔒 Code Smell ▾ 🔴 Major ▾ 🔵 Open ▾ Not assigned ▾ 2min effort Comment

🐞 suspicious, tests ▾

In our tests we sometimes specified the order of variables in the asserts in the wrong order: the expected value should be first, followed by the returned value.

Sonarqube didn't flag all of them, as seen in the screenshot above the first assert also has the order of variables in the wrong order, this made us check and fix all asserts (using some regex).

Class level

```

...
@Generated
// In order to be hidden from the test coverage report
// Package GUI is excluded for test coverage as allowed, but still shows up in the report otherwise
public class MatchScreen implements Screen, InputProcessor {

```

```

    public static float worldWidth = 2;

```

Make worldWidth a static final constant or non-public and provide accessors if needed. See Rule

2 months ago ▾ L39 🔗

🔒 Vulnerability ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 10min effort Comment

🐞 cwe ▾

Make this "public static worldWidth" field final See Rule

2 months ago ▾ L39 🔗

🔒 Vulnerability ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 20min effort Comment

🐞 cert, cwe ▾

```

    public transient float worldHeight;

```

Make worldHeight a static final constant or non-public and provide accessors if needed. See Rule

last month ▾ L40 🔗

🔒 Vulnerability ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 10min effort Comment

🐞 cwe ▾

```

    public static transient float musicVolume = 0.5f;

```

Make musicVolume a static final constant or non-public and provide accessors if needed. See Rule

22 hours ago ▾ L41 🔗

🔒 Vulnerability ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 10min effort Comment

🐞 cwe ▾

Make this "public static musicVolume" field final See Rule

22 hours ago ▾ L41 🔗

🔒 Vulnerability ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 20min effort Comment

🐞 cert, cwe ▾

Fixed by making them private, static and final (where applicable) and providing getters and setters where needed, also renamed each static variable to UPPERCASE.



Unnecessary public static variables: fixed by making them private and providing getters and setters where needed.



Static final variable not matching the conventional naming scheme, changed to match.



Unnecessary overrides, fixed by removing them.



The analysis didn't detect this completely correctly, this should actually be a *final* variable, of which the naming scheme does match. Changed the variable to final to fix.

Some graphs showing our improvements



This shows the amount of code smells over time, left is before the refactor and to the right is after.



This shows the amount of vulnerabilities over time, we fixed all the reported ones as can be seen in the graph.



Sonarqube also gave us an “A” in all categories with which we are really satisfied.

Remarks

Sonarqube reported that we added a lot of unnecessary “transient” specifiers, however we chose to remove that rule. This is because otherwise PMD would complain, in fact PMD actually ‘forced’ us to use all those “transient’ specifiers. It was simply not allowed to have variables that didn’t have either static or transient before them. Because PMD is one of the tools we had to use according to the template project/project description, we decided to following is rule in this case and suppressed Sonarqube’s.