



Department of Computer Engineering

Course Name: Software Engineering

Course Number: 10636312

Assignment Report

Instructor: Dr. Haya Samaana

Academic Year: 2024 - 2025

Semester: 2nd

Students

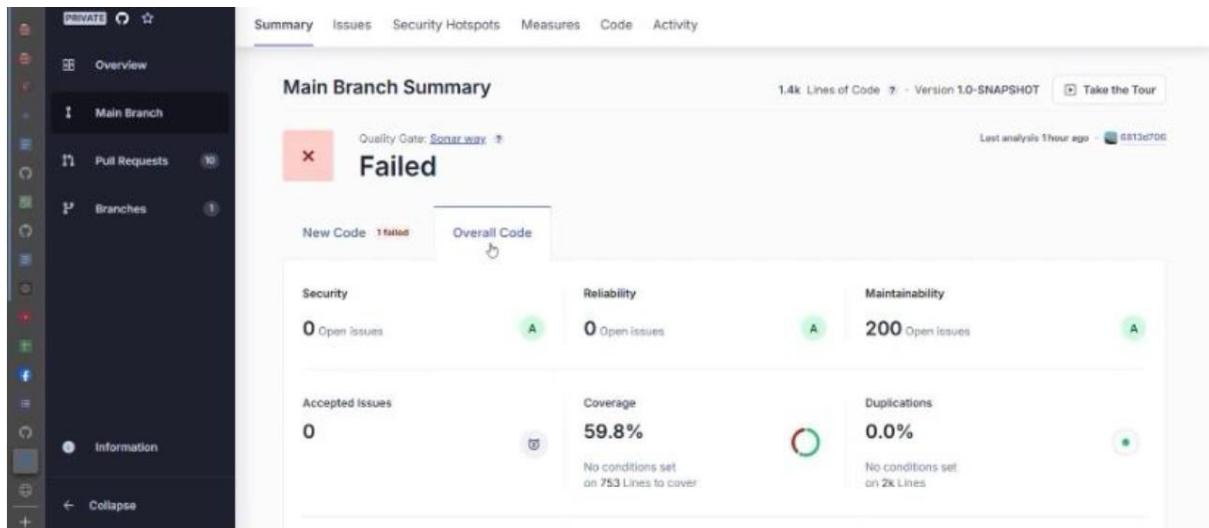
1- Taqwa Odeh

2- Ghayda Saify

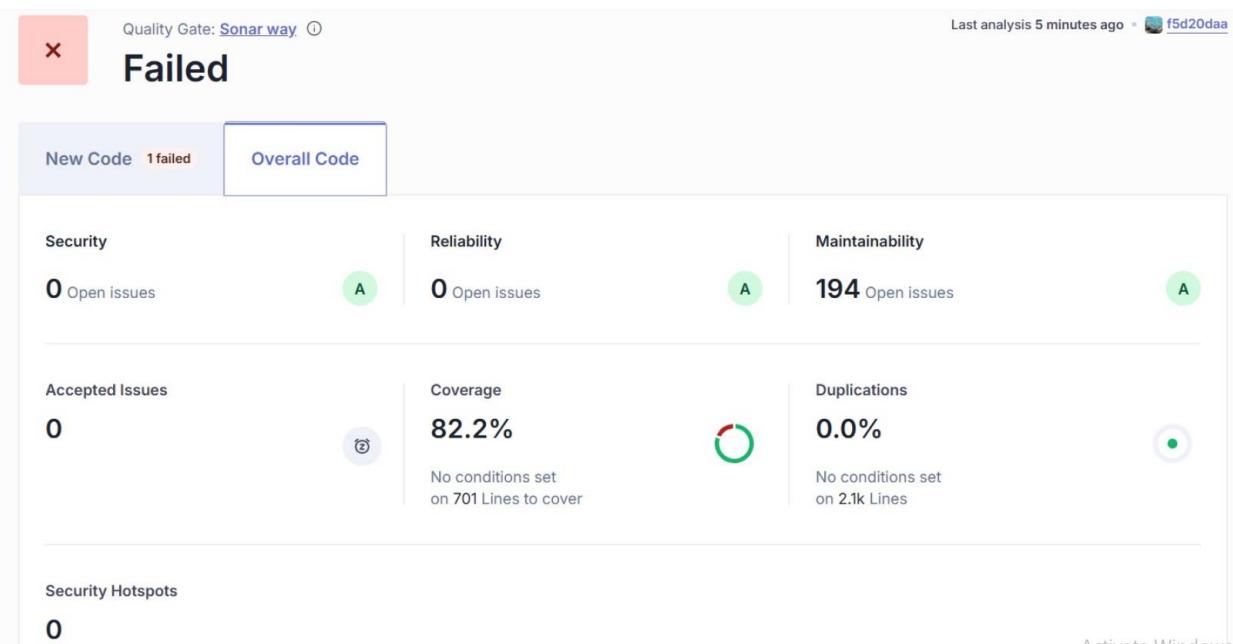
3- Tala Alhendi



- The results before refactoring :



- The enhanced code before doing the 4 bad smells refactoring (Coverage >81%)





Element	Class, %	Method, %	Line, %	Branch, %
org.example.Client	100% (22/22)	82% (183/223)	84% (467/550)	66% (81/121)
org.example.Instructor	100% (1/1)	68% (30/44)	70% (62/88)	33% (4/12)
org.example.User	100% (1/1)	72% (18/25)	76% (35/46)	50% (2/4)
org.example.Program	100% (1/1)	83% (5/6)	92% (12/13)	50% (2/4)
org.example.ProgramService	100% (1/1)	92% (24/26)	90% (58/64)	50% (6/12)
org.example.Admin	100% (2/2)	90% (31/34)	93% (129/138)	75% (43/57)
org.example.InstructorRepository	100% (1/1)	83% (5/6)	73% (14/19)	83% (5/6)
org.example.ClientRepository	100% (1/1)	83% (5/6)	73% (14/19)	83% (5/6)
org.example.Recipe	100% (1/1)	75% (3/4)	83% (5/6)	100% (0/0)
org.example.AdminService	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
org.example.Profile	100% (1/1)	100% (10/10)	100% (10/10)	100% (0/0)
org.example.UserStatus	100% (1/1)	100% (2/2)	100% (6/6)	100% (0/0)
org.example.Article	100% (1/1)	100% (3/3)	100% (5/5)	100% (0/0)
org.example.Content	100% (1/1)	100% (9/9)	100% (16/16)	100% (0/0)
org.example.Role	100% (1/1)	100% (2/2)	100% (2/2)	100% (0/0)
org.example.HealthTip	100% (1/1)	100% (3/3)	100% (5/5)	100% (0/0)
org.example.ProgramManager	100% (1/1)	100% (3/3)	100% (10/10)	100% (0/0)
org.example.SubscriptionService	100% (1/1)	80% (4/5)	80% (4/5)	100% (0/0)
org.example.Schedule	100% (1/1)	33% (3/9)	43% (7/16)	100% (0/0)
org.example.ProgressManager	100% (1/1)	100% (7/7)	100% (21/21)	100% (0/0)
org.example.Complaint	100% (1/1)	75% (3/4)	83% (5/6)	100% (0/0)
org.example.SubscriptionPlan	100% (1/1)	75% (3/4)	83% (5/6)	100% (0/0)

Refactoring 4 bad smells :

- Renaming bad smell:

Summary Issues Security Hotspots Measures Code Activity

1 / 8 issues

src/.../example/Client.java

Rename method "hasReceivedFeedback" to prevent any misunderstanding/clash with method "hasReceivedfeedback". ↗
Methods and field names should not be the same or differ only by capitalization
java:S1845

Software qualities impacted: (Maintainability)

Open Not assigned Code Smell Blocker

Where Why Activity

130 public void receiveFeedback(String feedback) {
131 feedbacks.add(feedback);
132 }
133
134 public boolean hasReceivedMessage(String message) {
135 return messages.contains(message);
136 }
137
138 public boolean hasReceivedFeedback(String feedback) {
 return false; // TODO: Implement this method
}

Rename method "hasReceivedFeedback" to prevent any

Open in IDE



```

112
113     ▲ Ghayda-Saify
114     public void setMessage(String message) { messages.add(message); }
115
116
117
118     no usages ▲ Ghayda-Saify
119     public boolean hasReceivedFeedback(String feedback) {
120         if (feedbacks.contains(feedback))
121         {
122             return true;
123         }
124         else
125             return false;
126     }
127
128     1 usage ▲ Ghayda-Saify
129     public void receiveMessage(String message) { messages.add(message); }
130
131     1 usage ▲ Ghayda-Saify
132     public void receiveFeedback(String feedback) { feedbacks.add(feedback); }
133
134     2 usages ▲ Ghayda-Saify
135     public boolean hasReceivedMessage(String message) { return messages.contains(message); }
136
137     1 usage ▲ Ghayda-Saify
138     public boolean hasReceivedFeedback(String feedback) { return feedbacks.contains(feedback); }
139
140     2 usages ▲ TalaAlhendi
141     public void setWorkoutsCompleted(int doneWorkouts, int allWorkouts) {
142         this.workoutsCompleted = doneWorkouts;
143         this.totalWorkouts = allWorkouts;
144     }

```

```

121     1 usage ▲ Ghayda-Saify
122     public void receiveFeedback(String feedback) { feedbacks.add(feedback); }
123
124     2 usages ▲ Ghayda-Saify
125     public boolean hasReceivedMessage(String message) { return messages.contains(message); }
126
127     1 usage ▲ Ghayda-Saify
128     public boolean hasReceivedFeedback(String feedback) { return feedbacks.contains(feedback); }
129
130     2 usages ▲ TalaAlhendi
131     public void setWorkoutsCompleted(int doneWorkouts, int allWorkouts) {
132         this.workoutsCompleted = doneWorkouts;
133         this.totalWorkouts = allWorkouts;
134     }
135
136     2 usages ▲ TalaAlhendi
137     public void setSessionsAttended(int sessionsAttended, int totalSessions) {
138         this.sessionsAttended = sessionsAttended;
139         this.totalSessions = totalSessions;
140     }
141
142     1 usage ▲ TalaAlhendi
143     public double getCompletionRate() {
144         if (totalWorkouts == 0) return 0;
145         return (double) workoutsCompleted / totalWorkouts * 100;

```



- Duplication bad smell:

Before:

```
List<Map<String, String>> revenueReport = new ArrayList<>();
for (Program program : programs) {
    double revenue = program
    .getClientsEnrolled().size() *Double.parseDouble( program.getPrice());
    revenueReport.add(Map.of(
        "Program Name", program.getTitle(),
        "Revenue", String.valueOf(revenue)
    ));
}
return revenueReport;
}
/**
```

Define a constant instead of duplicating this literal "Program Name" 4 times.

```
        "Revenue", String.valueOf(revenue)
    )));
}
return revenueReport;
}
/**
```

** Retrieves the statuses of all programs as a table, represented as a list of maps.*

** Each map contains the following keys:*

- * - "Program NAME" - the name of the program.*
- * - "Status" - the status of the program ("Upcoming", "Active", or "Completed").*

Activate Windows
Go to Settings to activate Wind

After:

```
188     * - "Active": The program is currently active.
189     * - "Completed": The program has already finished.
190
191     * @return a list of maps where each map represents a program and its corresponding status.
192     */
193
194     public List<Map<String, String>> getProgramStatusesAsTable() {
195         List<Map<String, String>> result = new ArrayList<>();
196         for (Program p : programs) {
197             if (p.getStartDate().after(new Date())){
198                 result.add(Map.of(
199                     "programNameDuplicated", p.getTitle(),
200                     "statusDuplicated", "Upcoming"
201                 ));
202             }
203             else if (p.getEndDate().after(new Date())){
204                 result.add(Map.of(
205                     "programNameDuplicated", p.getTitle(),
206                     "statusDuplicated", "Active"
207                 ));
208             }
209             else if (p.getEndDate().before(new Date())){
210                 result.add(Map.of(
211                     "programNameDuplicated", p.getTitle(),
212                     "statusDuplicated", "Completed"
213                 ));
214             }
215         }
216     }
```



- End the switch case with an unconditional break , return or throw statement

Intentionality | Not clear

End this switch case with an unconditional break, return or throw statement.

Switch cases should end with an unconditional "break" statement [java:S128](#)

Software qualities impacted: [Maintainability](#)

Open Ghayda-Saify Code Smell Blocker

Tags cert ... +

Line affected L363

Effort 10 min

Introduced 4 hours ago

Where Why Activity More info Open in IDE

```

370 g.safw_
371 g.safw_ ●
372 g.safw_
373 g.safw_
374
375 g.safw_
376 g.safw_
377
378 g.safw_
379 g.safw_
380
381
382 g.safw_
383 g.safw_
384
385 g.safw_
386 g.safw_

```

```

    case CLIENT:
        assert client != null;
        if(client.getPassword().equals(password)){
            client.setLoggedIn(true);
            System.out.println("You have signed in successfully.");
            return true;
        }

        case ADMIN:
            if(password.equals("123456")){
                this.loggedIn=true;
                System.out.println("You have signed in successfully.");
                return true;
            }
            else {
                System.out.println("Invalid email or password.");
            }

```

case CLIENT:

End this switch case with an unconditional break, return or throw statement.

```

assert client != null;
if(client.getPassword().equals(password)){
    client.setLoggedIn(true);
    System.out.println("You have signed in successfully.");
    return true;
}

```

```

switch (role){
    case INSTRUCTOR:
        assert instructor != null;
        if(instructor.getPassword().equals(PASSWORD)){
            instructor.setLoggedIn(true);
            System.out.println("You have signed in successfully.");
            return true;
        }
        return false;

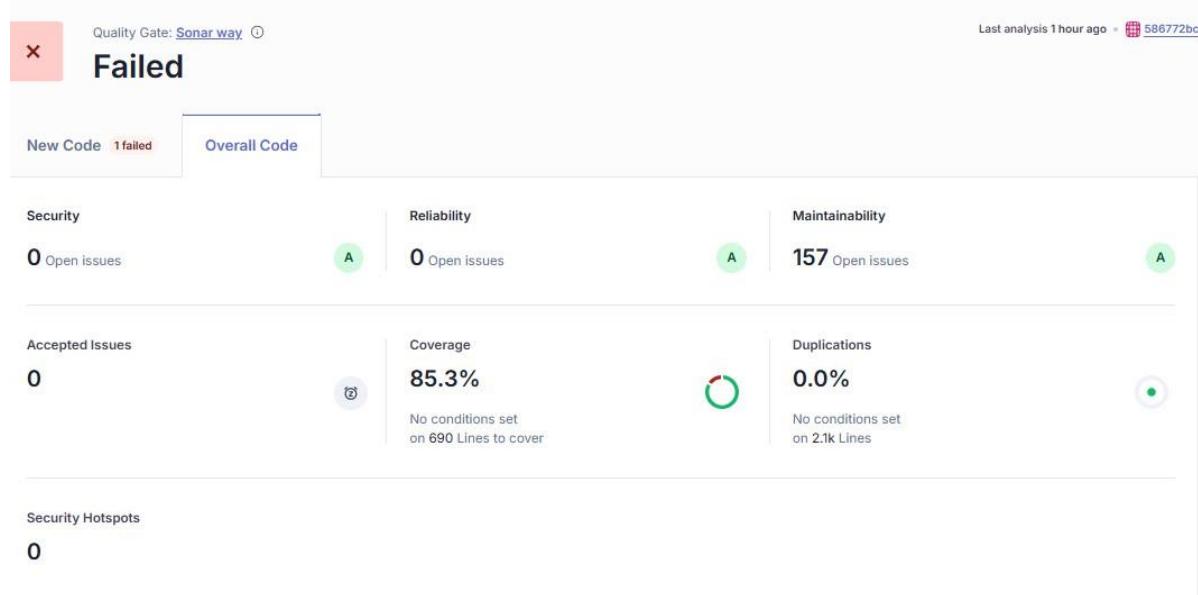
    case CLIENT:
        assert client != null;
        if(client.getPassword().equals(PASSWORD)){
            client.setLoggedIn(true);
            System.out.println("You have signed in successfully.");
            return true;
        }
        return false;
}

```



The result after refactoring :

Element	Class, %	Method, %	Line, %	Branch, %
org.example	100% (23/23)	86% (193/224)	87% (482/551)	66% (81/121)
Client	100% (1/1)	68% (30/44)	70% (62/88)	33% (4/12)
Instructor	100% (1/1)	75% (18/24)	83% (35/42)	50% (2/4)
User	100% (1/1)	83% (5/6)	92% (12/13)	50% (2/4)
Program	100% (1/1)	92% (24/26)	90% (58/64)	50% (6/12)
ProgramService	100% (1/1)	90% (10/11)	85% (42/49)	70% (14/20)
Admin	100% (2/2)	93% (31/33)	92% (131/141)	75% (43/57)
InstructorRepository	100% (1/1)	100% (5/5)	100% (16/16)	83% (5/6)
ClientRepository	100% (1/1)	83% (5/6)	73% (14/19)	83% (5/6)
FeedbackAndReviewsManager	100% (1/1)	100% (5/5)	100% (6/6)	100% (0/0)
Recipe	100% (1/1)	100% (3/3)	100% (5/5)	100% (0/0)
AdminService	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
Profile	100% (1/1)	100% (10/10)	100% (10/10)	100% (0/0)
UserStatus	100% (1/1)	100% (2/2)	100% (6/6)	100% (0/0)
Article	100% (1/1)	100% (3/3)	100% (5/5)	100% (0/0)
Content	100% (1/1)	100% (9/9)	100% (16/16)	100% (0/0)
Role	100% (1/1)	100% (2/2)	100% (2/2)	100% (0/0)
HealthTip	100% (1/1)	100% (3/3)	100% (5/5)	100% (0/0)
ProgramManager	100% (1/1)	100% (3/3)	100% (10/10)	100% (0/0)
SubscriptionService	100% (1/1)	80% (4/5)	80% (4/5)	100% (0/0)
Schedule	100% (1/1)	88% (8/9)	75% (12/16)	100% (0/0)
ProgressManager	100% (1/1)	100% (7/7)	100% (21/21)	100% (0/0)
Complaint	100% (1/1)	75% (3/4)	83% (5/6)	100% (0/0)
SubscriptionPlan	100% (1/1)	75% (3/4)	83% (5/6)	100% (0/0)





All test Cases True :

Element	Class, %	Method, %	Line, %
org.example.AdminService	100% (1/1)	69% (30/43)	72% (62/86)
org.example.Client	100% (1/1)	88% (8/9)	75% (12/16)
org.example.Schedule	100% (1/1)	83% (5/6)	77% (17/22)
org.example.ClientRepository	100% (1/1)	75% (3/4)	83% (5/6)
org.example.Complaint	100% (1/1)	75% (3/4)	83% (5/6)
org.example.SubscriptionPlan	100% (1/1)	75% (3/4)	83% (5/6)
org.example.SubscriptionService	100% (1/1)	80% (4/5)	83% (5/6)
org.example.Instructor	100% (1/1)	75% (18/24)	83% (35/42)
org.example.ProgramService	100% (1/1)	90% (10/11)	84% (45/53)
org.example.Program	100% (1/1)	92% (24/26)	90% (58/64)
org.example.Admin	100% (2/2)	91% (31/34)	91% (133/145)
org.example.User	100% (1/1)	83% (5/6)	92% (12/13)
org.example.Article	100% (1/1)	100% (3/3)	100% (5/5)
org.example.HealthTip	100% (1/1)	100% (3/3)	100% (5/5)
org.example.Recipe	100% (1/1)	100% (3/3)	100% (5/5)
org.example.UserStatus	100% (1/1)	100% (2/2)	100% (6/6)
org.example.FeedbackAndReviewsManager	100% (1/1)	100% (5/5)	100% (10/10)
org.example.Profile	100% (1/1)	100% (10/10)	100% (11/11)
org.example.ProgramManager	100% (1/1)	100% (3/3)	100% (14/14)
org.example.Content	100% (1/1)	100% (9/9)	100% (16/16)
org.example.InstructorRepository	100% (1/1)	100% (5/5)	100% (19/19)
org.example.ProgressManager	100% (1/1)	100% (7/7)	100% (25/25)



The generated UML diagram:



[The UML diagram pic for clearer version, open it with draw.io for better clarity.](#)

Thank You Dr. Haya 😊

Feature Completed ✓
Phase 2 😊 :

