



An-Najah National University
Faculty of Engineering
Computer Engineering Department

Preparing By
Sama' Ghazi Yasin
&
Tala Mohammad Yaseen
Supervisor
Dr.Haya Samaana

Sakancom project

In our project we made 10 features.

Tala Yaseen achievements:

1. Everything has to do with the admin (Features, Testing and source code).
2. 70% of the Tenant feature.

Sama' Yasin achievements:

1. Everything has to do with the owner (Features, Testing and The source code).
2. log in feature.
3. 30% of the Tenant feature.

Our system includes 3 menus:

❖ **Admin's menu**

1. View requests for advertisement
2. Accept a housing advertisement
3. Reject a housing advertisement
4. View reservations
5. Add and advertise a housing unit
6. Modify housing data
7. Exit.

❖ **Owner's menu**

1. Add a private residence
2. View listed residences
3. display the number of tenants and floors in specific residence
4. The names of the tenants and their contact information and the number of bathrooms and bedrooms, and balcony availability should be displayed
5. Exit.

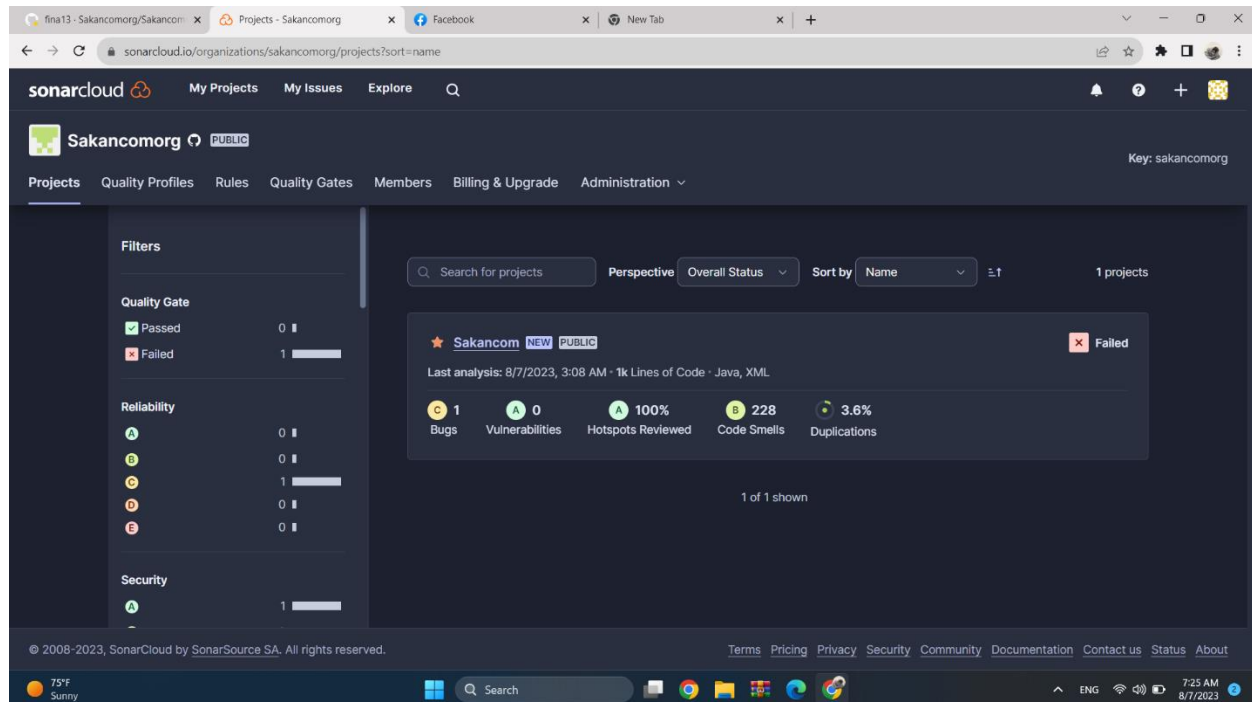
❖ **Tenant's menu**

1. View available housing.
2. View housing details and pictures.
3. Book accommodation.
4. View information about housemates.
5. Advertise used furniture for sale.
6. View tenant control panel.
7. Exit.

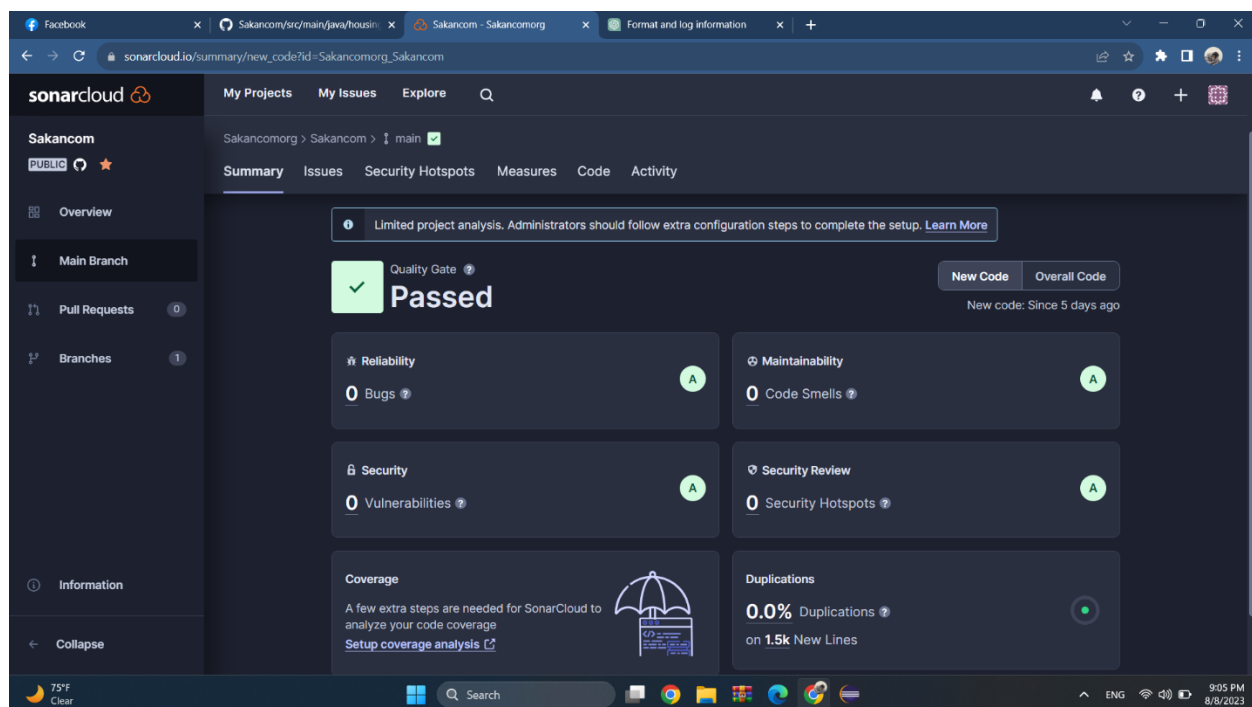
We have created an account on the GitHub (is a platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code), and we add repository to it to push our project, in order to ensure the quality of the project, and we have connected it with the sonar cloud to check the bad smells in our project, so that we can fix it, and it checks maintainability, reliability, and security for it thus improving the overall quality for project.

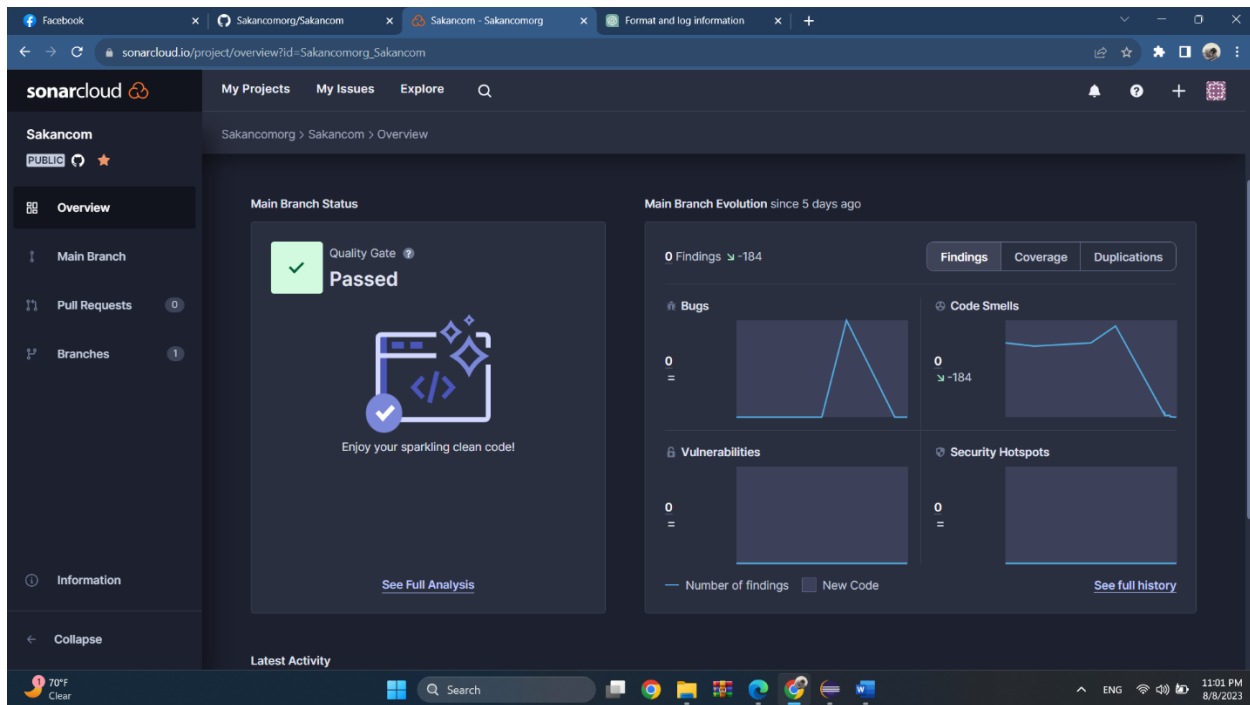
In the first analysis:

Main Branch Evolution:



Then we refactoring our project to decrease the number of code smells and Bugs:

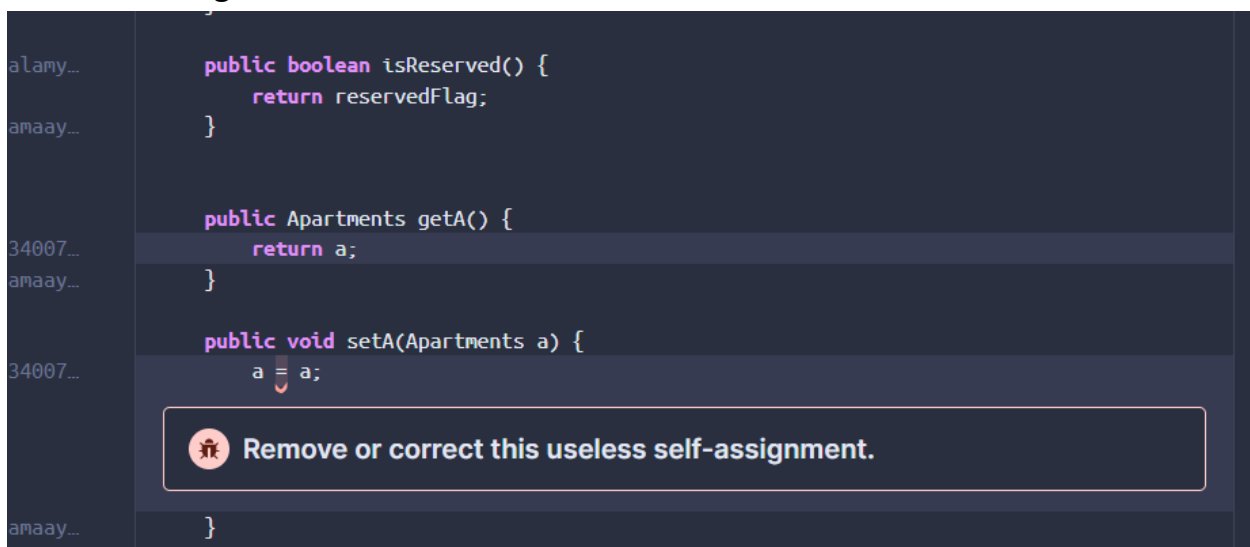




AcceptanceTest (م ١١:٥٣:١٦ ٢٠٢٣/٠٨/٠٨)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
<ul style="list-style-type: none"> Sakancom <ul style="list-style-type: none"> src/main/java <ul style="list-style-type: none"> src/test/java 	<ul style="list-style-type: none"> 88,7 % 86,7 % 98,9 % 	<ul style="list-style-type: none"> ٢,٤٣٧ ١,٧١٠ ٧٢٧ 	<ul style="list-style-type: none"> ٣١٤ ٢٧٣ ٤١ 	<ul style="list-style-type: none"> ٢,٧٥١ ١,٩٨٣ ٧٦٨

We have a bug:



We solve this bug like this:

```

        return a;
    }

    public void setA(Apartments a) {
        this.a = a;
    }

```

Some examples for bad smells we refactored

1. The parameters in constructor more than 7.

Before:

```

public HousingUnit(String location, String photo, double rent, int numberOfTenants, int numberOfFloors,
String availableServices, boolean reservedFlag, Apartments A, boolean studentHouse, int id) {
    this.location = location;
    this.photo = photo;
    this.rent = rent;
    this.numberOfTenants = numberOfTenants;
    this.numberOfFloors = numberOfFloors;
    this.availableServices = availableServices;
    this.reservedFlag = reservedFlag;
    this.A = A;
    this.studentHouse = studentHouse;
    this.id = id;
}

```

After:

```

public HousingUnit(LocationInfo locationInfo, Apartments apartmentInfo,
boolean reservedFlag, boolean studentHouse, int numoffloors, String availableServices, int id) {
    this.locationInfo = locationInfo;
    this.apartmentInfo = apartmentInfo;
    this.reservedFlag = reservedFlag;
    this.studentHouse = studentHouse;
    this.numoffloors=numoffloors;
    this.availableServices=availableServices;
    this.id = id;
}

```

2. Delete the comments because it gives code smells

Before:

```

public void initarrayvalues() {
    /* userinfo.add(new User("admin", "samaa", "samaa@gmail.com", "4545", 1));
    userinfo.add(new User("tenant", "malak", "m@gmail.com", "1234", 1));
    userinfo.add(new User("admin", "tala", "tala@gmail.com", "7676", 1)); */
    userinfo.add(new User("admin", "samaa", "samaa@gmail.com", "4545", 1));
    userinfo.add(new User("tenant", "malak", "m@gmail.com", "1234", 1));
    userinfo.add(new User("owner", "nancy", "N@gmail.com", "4444", 1));
    userinfo.add(new User("admin", "tala", "tala@gmail.com", "7676", 1));
}

```

After:

```

56
57     }
58     public void initarrayvalues() {
59         userinfo.add(new User("admin", "samaa", "samaa@gmail.com", "4545", 1));
60         userinfo.add(new User("tenant", "malak", "m@gmail.com", "1234", 1));
61         userinfo.add(new User("owner", "nancy", "N@gmail.com", "4444", 1));
62         userinfo.add(new User("admin", "tala", "tala@gmail.com", "7676", 1));
63     }
64
65     public static ArrayList<HousingUnit> ReservationsList() {

```

3. System.Out.println(" ");

Before:

```

System.out.println("===== Housing Owner Menu =====");
System.out.println("1. Add a private residence");
System.out.println("2. View listed residences");
System.out.println("3. display the number of tenants and floors in specific residence ");
System.out.println("4. The names of the tenants and their contact information and the number of floors");
System.out.println("5. Exit");
System.out.print("Enter your choice: ");

```

After:

```

logger.info("===== Housing Owner Menu =====");
logger.info("1. Add a private residence");
logger.info("2. View listed residences");
logger.info("3. Display the number of tenants and floors in specific residence");
logger.info("4. Display tenant information and apartment details");
logger.info("5. Exit");
logger.info("Enter your choice: ");

```

4. Convert Array list to list

Before:

```

public ArrayList<String> getTenantNames() {
    return tenantNames;
}

public void setTenantNames(ArrayList<String> tenantNames) {
    this.tenantNames = tenantNames;
}

public ArrayList<String> getMeansOfCommunication() {
    return meansOfCommunication;
}

public void setMeansOfCommunication(ArrayList<String> meansOfCommunication) {
    this.meansOfCommunication = meansOfCommunication;
}
private int floor;

```

After:

```

public List<String> getTenantNames() {
    return tenantNames;
}

public void setTenantNames(List<String> tenantNames) {
    this.tenantNames = tenantNames;
}

public List<String> getMeansOfCommunication() {
    return meansOfCommunication;
}

public void setMeansOfCommunication(List<String> meansOfCommunication) {
    this.meansOfCommunication = meansOfCommunication;
}

```

5. Conver public variable to private with setter and getter

Before:

```

1 package housing.unit;
2
3 public class User {
4     public String name;
5     public String email;
6     public String pa;
7     String role;
8     public int s;
9
10    User(String role,String name ,String email , String pa,int s){
11        this.name=name;
12        this.email=email;
13        this.pa=pa;
14        this.s=s;
15        this.role=role;
16    }
17
18 }
19 |

```

After:


```
User.java x LoginSteps.java LoginTest.java
1 package housing.unit;
2
3 public class User {
4     private String name;
5     private String email;
6     private String pa;
7     String role;
8     private int s;
9
10    User(String role,String name ,String email , String pa,int s){
11        this.name=name;
12        this.email=email;
13        this.pa=pa;
14        this.s=s;
15        this.role=role;
16    }
17
18    public String getName() {
19        return name;
20    }
21
22    public void setName(String name) {
23        this.name = name;
24    }
25
26    public String getEmail() {
27        return email;
28    }
29
30    public void setEmail(String email) {
31        this.email = email;
32    }
33
34    public String getPa() {
35        return pa;
36    }
37 }
```

6. Add private constructor to hide implicit one

Before:

```
import java.util.ArrayList;

public class HouseExist {

    Add a private constructor to hide the implicit public one.

    public static boolean check(HousingUnit enteredhouse, ArrayList<HousingUnit>
HousingUnitList) {
    // TODO Auto-generated method stub
    for (HousingUnit house : HousingUnitList) {
        if(house.equals(enteredhouse))
            return true;
    }

    return false;
}
```

After:

```

1 package housing.unit;
2
3 import java.util.List;
4
5 public class HouseExist {
6     private HouseExist() {
7     }
8
9     public static boolean check(HousingUnit enteredhouse, List<HousingUnit> housingunitlist) {
10         for (HousingUnit house : housingunitlist) {
11             if (house.equals(enteredhouse)) {
12                 return true;
13             }
14         }
15         return false;
16     }
17 }
18
19 }
20
21 |

```

7. built-in formatting to construct the argument for the logger.info

Before:

```

meansOfCommunication = apartment.getMeansOfCommunication();

for (int i = 0; i < tenantNames.size(); i++) {
    logger.info("Tenant Name: " + tenantNames.get(i));
}

```

After:

```

for (int i = 0; i < tenantNames.size(); i++) {
    String tenantInfo = String.format("Tenant Name: %s, Means of Communication: %s",
                                      tenantNames.get(i), meansOfCommunication.get(i));
    logger.info(tenantInfo);
}

```

8. Before:

Sakancomorg > Sakancom > main

Summary Issues Security Hotspots Measures Code Activity

1 / 1 issues

src/.../unit/ModifyHouseUnit.java

Change this condition so that it does not always evaluate to "false"

🐞 Bug 1 execution flow

1 Implies 'housingUnitToModify' is not null.

2 Expression is always false.

Navigate locations Alt + ⬆ ⬇ ⬇

1 of 1 shown

Where is the issue? Why is this an issue? Activity More info

```

13 - private static HousingUnit modifyHouseUnitToModify;
14 - private static int index;
15 - public static void modify() {
16 -     logger.info("Enter the ID of the housing unit to modify: ");
17 -     housingUnitToModify = HousingUnit.FindHouse(MyData.getHousingUnitList());
18 -     if (1 housingUnitToModify == null) return;
19 -     index = MyData.housingunitlist.indexOf(housingUnitToModify);
20 -     Scanner scanner = new Scanner(System.in);
21 -     if (2 housingUnitToModify == null) {
22 -         logger.info("No housing unit found with the given ID.");
23 -         return;
24 -     }
25 -     logger.info("Enter the new location: ");
26 -     String newLocation = scanner.next();
27 -     housingUnitToModify.setLocation(newLocation);
28 -
29 -     logger.info("Enter the new photo: ");
30 -     String newPhoto = scanner.next();

```

🐞 Change this condition so that it does not always evaluate to "false"

Show 46 more lines

After:

```

logger.info("Enter the ID of the housing unit to modify: ");
housingUnitToModify = HousingUnit.FindHouse(MyData.getHousingUnitList());
if (housingUnitToModify == null) {
    logger.info("No housing unit found with the given ID.");
    return;
}

index = MyData.housingunitlist.indexOf(housingUnitToModify);
Scanner scanner = new Scanner(System.in);

```

9. Before:

12 / 13 issues

"TenantSystem.class".

🐞 Code Smell

src/.../HousingUnitsTesting.java

Use assertEquals instead.

🐞 Code Smell 1 location

Use assertNotNull instead.

🐞 Code Smell 1 location

🔴 A null-check is performed here, which is better expressed with assertNotNull.

Use assertNotNull instead.

🐞 Code Smell 1 location

Use assertNotNull instead.

🔗

JUnit assertTrue/assertFalse should be simplified to the corresponding dedicated assertion [java:S5785](#) junit tests

🐞 Code Smell Open Major talamyaseen L77 • 2min effort • 1 day ago

Where is the issue? Why is this an issue? Activity

```

76 - // Write code here that turns the phrase above into concrete actions
77 - assertTrue(• ModifiesHouseBefore!=null);
78 -
79 - }
80 -
81 - @When("the Admin makes changes to the housing data")
82 - public void the_admin_makes_changes_to_the_housing_data() {
83 -     // Write code here that turns the phrase above into concrete actions
84 -     assertTrue(ModifiesHouseAfter!=null);
85 -
86 - }

```

🐞 Use assertNotNull instead.

Show 500 more lines

After:

```
76  
77     ModifyHouseUnit.modifyAdmin();  
78     this.ModifiesHouseBefore= ModifyHouseUnit.getHousingUnitToModify();  
79     assertNotNull(ModifiesHouseBefore);  
80  
81 }  
82  
83
```

We remove duplicate in code :

Before:

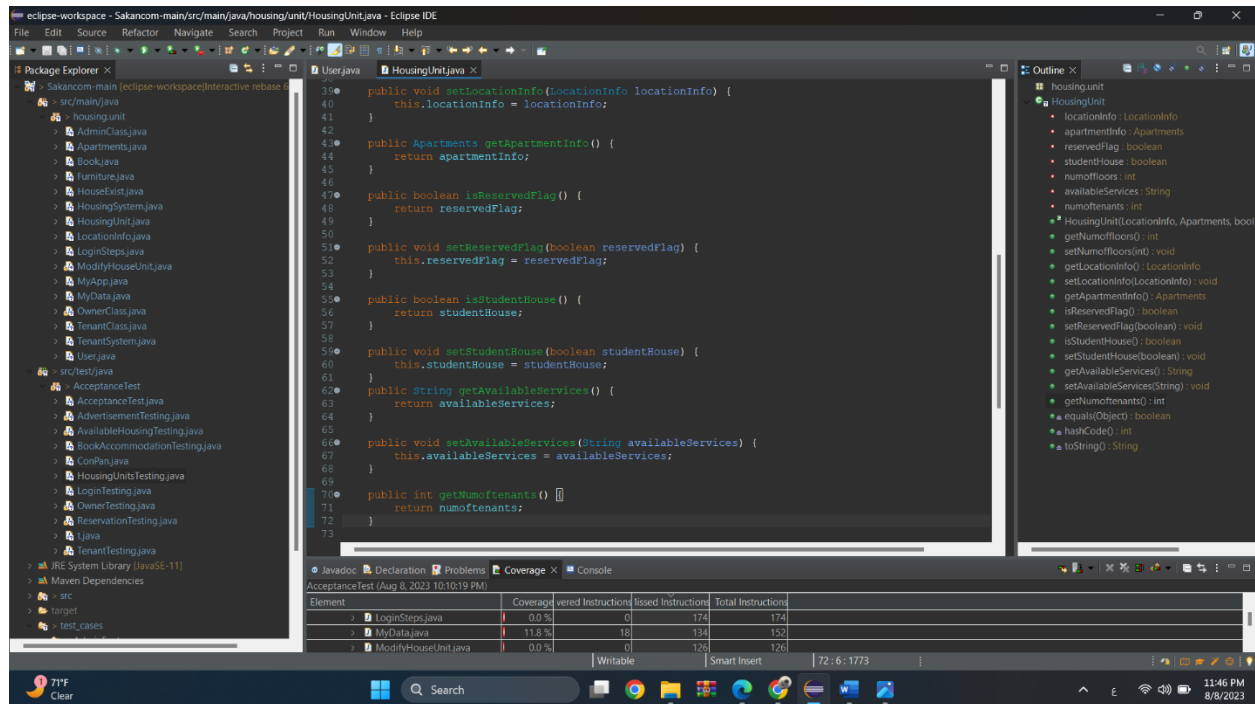
The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with packages like `src/main/java` and `src/test/java`.
- Outline:** Shows the class hierarchy for `HousingUnit`, including fields like `locationInfo`, `apartmentInfo`, `reservedFlag`, `studentHouse`, `numoffloors`, `availableServices`, `numof tenants`, and methods like `getAvailableServices()`, `setAvailableServices()`, `getNumof tenants()`, `setNumof tenants()`, `getLocationInfo()`, `setLocationInfo()`, `getApartmentInfo()`, `setApartmentInfo()`, `isReservedFlag()`, `setReservedFlag()`, `isStudentHouse()`, `setStudentHouse()`, `equals()`, `hashCode()`, and `toString()`.
- Code Editor:** Displays the `HousingUnit.java` file with the following code:

```
56  
57 public boolean isReservedFlag() {  
58     return reservedFlag;  
59 }  
60  
61 public void setReservedFlag(boolean reservedFlag) {  
62     this.reservedFlag = reservedFlag;  
63 }  
64  
65 public boolean isStudentHouse() {  
66     return studentHouse;  
67 }  
68  
69 public void setStudentHouse(boolean studentHouse) {  
70     this.studentHouse = studentHouse;  
71 }  
72  
73  
74 @Override  
75 public boolean equals(Object o) {  
76     if (this == o) return true;  
77     if (o == null || getClass() != o.getClass()) return false;  
78     HousingUnit that = (HousingUnit) o;  
79     return locationInfo.getId() == that.locationInfo.getId();  
80 }  
81  
82 @Override  
83 public int hashCode() {  
84     return Objects.hash(locationInfo.getId());  
85 }  
86  
87 @Override  
88 public String toString() {  
89     return "ID: " + locationInfo.getId() +  
90         ", Location: " + locationInfo.getLocation() +  
91         ", Reser: " + locationInfo.getReser() +
```
- Coverage:** Shows a table with columns: Element, Coverage, Covered Instructions, Missed Instructions, Total Instructions.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> LoginSteps.java	0.0 %	0	174	174
> MyData.java	11.8 %	18	134	152
> ModifyHouseUnit.java	0.0 %	0	126	126
- Console:** Shows the output of the test run.

After:



Then we build the project in Jenkins

The console output:

```
[INFO] argLine set to -
javaagent:C:\Windows\system32\config\systemprofile\.m2\repository\org\jacoco\org.jacoco.agent\0.8.7\org.jacoco.agent-0.8.7-
runtime.jar=destfile=C:\ProgramData\Jenkins\.jenkins\workspace\Sakancom\target\jacoco.exec
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ Sakancom ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\ProgramData\Jenkins\.jenkins\workspace\Sakancom\src\main\resources
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ Sakancom ---
[INFO] Changes detected - recompiling the module! :source
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 16 source files with javac [debug target 11] to target\classes
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.964 s
[INFO] Finished at: 2023-08-08T21:07:06+03:00
[INFO] -----
```