

# Software Engineering

## Online Library

Dr.Haya Samaana

Students :

Sarah Hinno 11926569

Marah Direeni 11923808

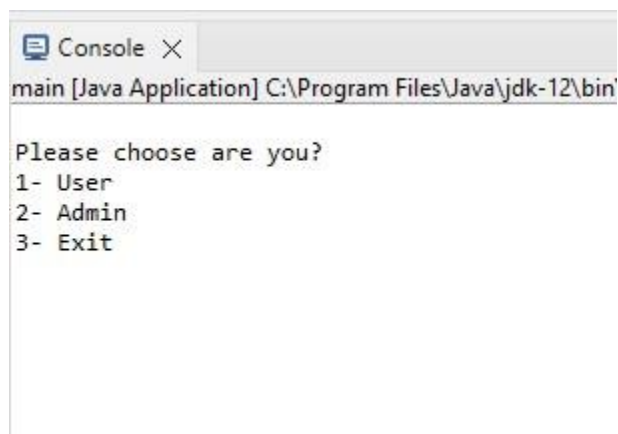
Yafa Nada 11924868

## ➤ Introduction :-

In our homework we build an online library that contain a two type of users the first is admin that can do many tasks like register user ,search and add book , and the second is the user that can borrow book and return it .

Here we have a screenshot that describe how our code work :-

Here we have the main that the user and the admin deal with :-



```
main [Java Application] C:\Program Files\Java\jdk-12\bin'
Please choose are you?
1- User
2- Admin
3- Exit
```

Console X  
main [Java Application] C:\Program Files\Java\jdk-12\bin\jav

```
Please choose are you?
1- User
2- Admin
3- Exit
2
Please Enter the password:
adminadmin
successfull log in
1- Register user
2- Search book
3- Add book
4- Unregister user
5- print all registered user
6- print all books in the library
7- Back to main menue
5
```

Console X  
main [Java Application] C:\Program Files\Java\jdk-12\bin\javaw.exe

```
Please choose are you?
1- User
2- Admin
3- Exit
1
Enter your ID
1234567
please choose a number:
1- Borrow book
2- Return book
3- Search book
4- print all books i have borrowed
5- Back to main menue
```

```
Please choose are you?
1- User
2- Admin
3- Exit
2
Please Enter the password:
adminadmin
successfull log in
1- Register user
2- Search book
3- Add book
4- Unregister user
5- print all registered user
6- print all books in the library
7- Back to main menue
7
Please choose are you?
1- User
2- Admin
3- Exit
1
Enter your ID
1234567
please choose a number:
1- Borrow book
2- Return book
3- Search book
4- print all books i have borrowed
5- Back to main menue
```

## ➤ Coverage :-

Then we make a coverage for our code before we build the main and the result we have got :-

The screenshot shows the Eclipse IDE interface with the following components:

- Top Bar:** eclipse-workspace - softwareProject/src/test/java/softwareProject/userRegisterSteps.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse IDE toolbar with icons for file operations, editing, and running.
- Left Panel:**
  - Project Explorer:** Shows the project structure with 'softwareProject' expanded.
  - JUnit:** Shows test results for 'softwareProject.AcceptanceTest [Runner: JUnit 4] (0.1s)'. It indicates 'Runs: 22/22', 'Errors: 0', and 'Failures: 0'.
  - Failure Trace:** Currently empty.
- Right Panel:**
  - Coverage Tab:** Displays a table of coverage data for the project and its sub-elements.
  - Java Editor:** Shows the source code for 'userRegisterSteps.java'.

Element	Coverage	Covered Instruction...	Missed Instructions	Total Instructions
softwareProject	96.3%	1,111	71	1,182
src/main/java	88.9%	881	00	881
softwareProject	88.9%	881	00	881
user.java	81.0%	192	40	232
admin.java	95.7%	189	0	189
book.java	100.0%	10	0	10
myLibrary.java	98.7%	160	2	162
src/test/java	99.0%	070	7	077
softwareProject	99.0%	070	7	077
AcceptanceTest.java	100.0%	3	0	3
addSteps.java	97.0%	70	2	72
searchSteps.java	99.1%	113	1	114
adminLoginSteps.java	100.0%	39	0	39
adminLogoutSteps.java	100.0%	18	0	18
borrowSteps.java	100.0%	177	0	177
returnSteps.java	100.0%	88	0	88
userRegisterSteps.java	100.0%	116	0	116

```
public void there_is_a_user_with_id_name_email_address_postal_code_and_city(String id, String na
u=new user(id,name,email,address,postal,city);
}
@When("admin tries to register a user")
public void admin_tries_to_register_a_user() {
    s=a.register(u);
    after=myLibrary.registeredUsers.size();
}
@Then("an error message {string} should display")
public void an_error_message_should_display(String string) {
    assertEquals(string,s);
}
```

eclipse-workspace - softwareProject/src/main/java/softwareProject/admin.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project Explorer JUnit X

Finished after 0.171 seconds

Runs: 22/22 Errors: 0 Failures: 0

softwareProject.AcceptanceTest [Runner: JUnit 4] (0.171 s)

Failure Trace

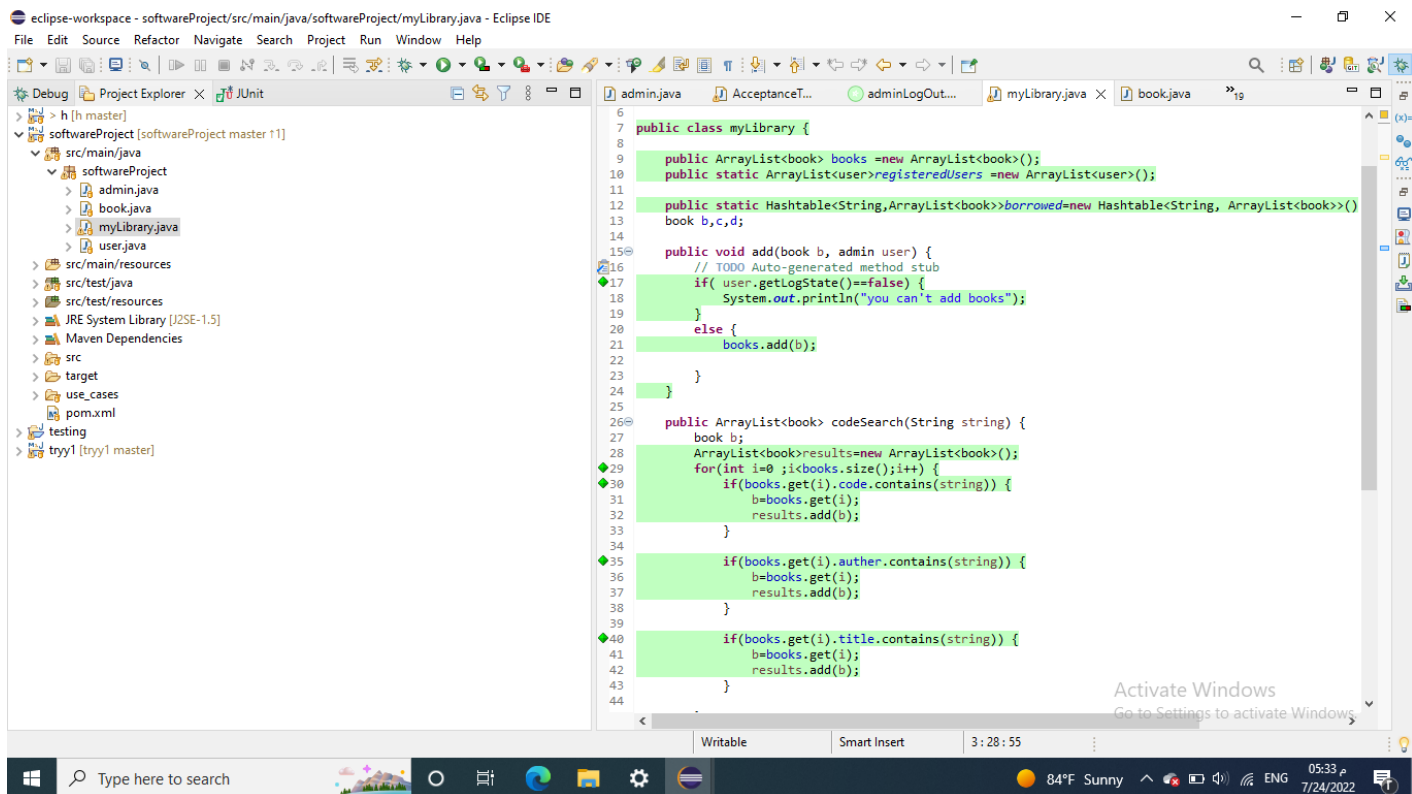
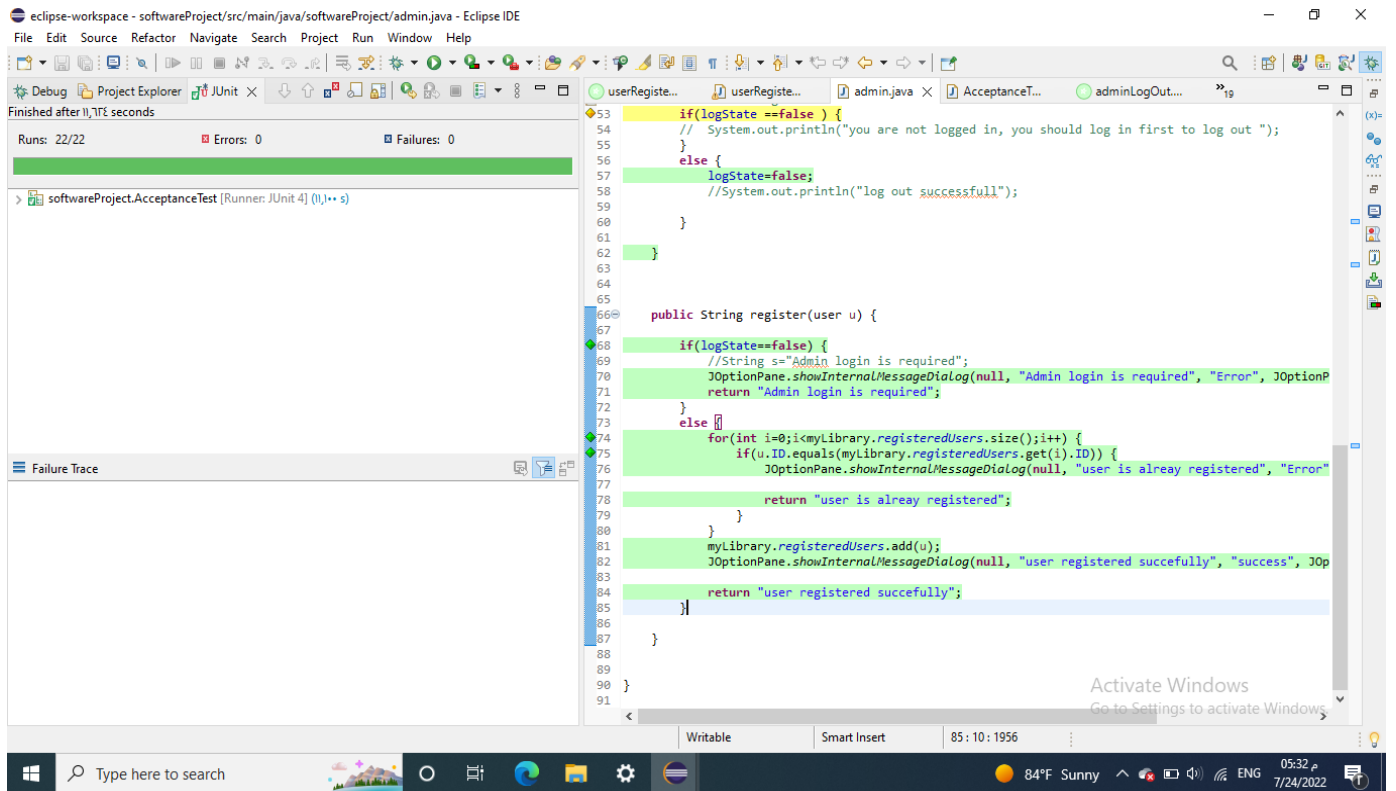
```
1 package softwareProject;
2
3 import javax.swing.JOptionPane;
4
5 public class admin {
6
7     boolean logState;
8     String password;
9
10
11     public admin() {
12         password="adminadmin";
13         logState=false;
14     }
15
16     public void setlogState(boolean b) {
17         // TODO Auto-generated method stub
18         logState=b;
19     }
20
21
22     public boolean login(String pass) {
23         // TODO Auto-generated method stub
24         if(logState){
25             System.out.println("you are already logged in");
26             return false;
27         }
28         else {
29             if(this.password.equals(pass)) {
30                 System.out.println("successfull log in");
31                 logState=true;
32                 return true;
33             }
34         }
35         else {
36             System.out.println("wrong password");
37             return false;
38         }
39     }
40 }
```

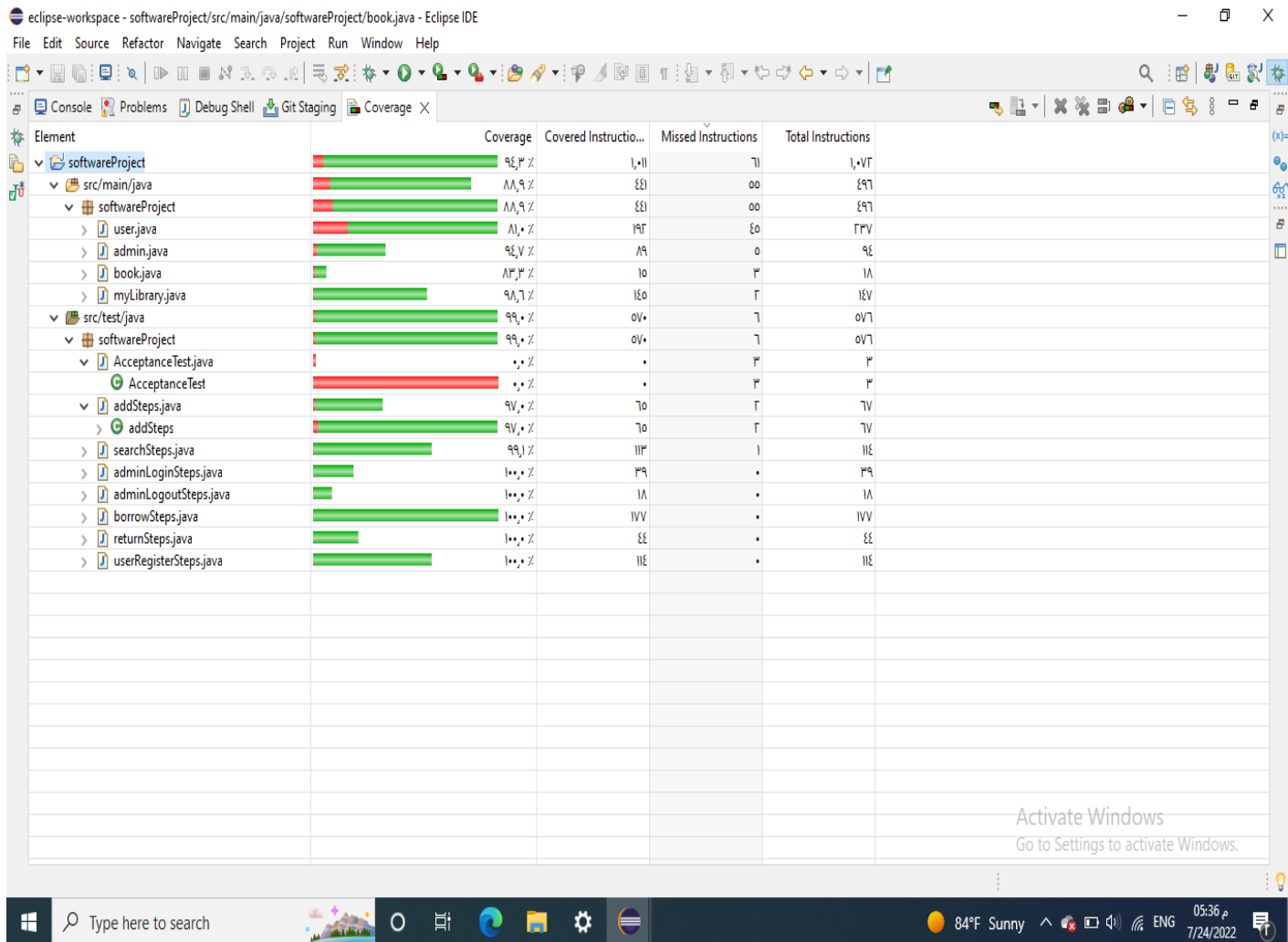
Writab... Smart Insert 85:10:1956

Activate Windows  
Go to Settings to activate Windows

Type here to search

84°F Sunny 05:32 PM 7/24/2022





As we see the percentage of coverage is good .

The picture bellow is the percentage of coverage is less than the coverage above because of the main that we can't implement a test cases for it .and it is the way to make the user enabled with the design ,it's like a GUI .

But for test files the coverage is high which mean that the senarios we wrote cover high percent of production code.

eclipse-workspace - softwareProject/src/test/java/softwareProject/AcceptanceTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Console Coverage X

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
softwareProject	77.8%	1,062	307	1,369
src/main/java	87.7%	790	119	909
softwareProject	87.7%	790	119	909
main.java	100%	0	0	0
user.java	100%	0	0	0
admin.java	100%	0	0	0
MockDateHolder.java	100%	0	0	0
myLibrary.java	100%	0	0	0
book.java	100%	0	0	0
DateServer.java	100%	0	0	0
src/test/java	99.1%	199	2	201
softwareProject	99.1%	199	2	201
AcceptanceTest.java	100%	0	0	0
addSteps.java	100%	0	0	0
borrowSteps.java	100%	0	0	0
lateBookSteps.java	100%	0	0	0
searchSteps.java	100%	0	0	0
adminLoginSteps.java	100%	0	0	0
adminLogoutSteps.java	100%	0	0	0
returnSteps.java	100%	0	0	0
unregisterSteps.java	100%	0	0	0
userRegisterSteps.java	100%	0	0	0

## ➤ SonarQube:-

We use the SonarQube that is a Static analysis ,then we record the resulted .

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Elib master Last analysis of this Branch had 1 warning July 13, 2022 at 8:08 PM Version 0.0.1-SNAPSHOT

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

The Project Analysis has failed. More details available on the [Background Tasks](#) page.

To benefit from more of SonarQube's features, [set up analysis in your favorite CI](#).

QUALITY GATE STATUS **Passed**  
All conditions passed.

MEASURES

New Code Overall Code

0 Bugs Reliability **A**

0 Vulnerabilities Security **A**

1 Security Hotspots **0.0%** Reviewed Security Review **E**

6h 57min Debt 63 Code Smells Maintainability **A**



sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministration

Search for projects...

A

Elib

master

Last analysis of this Branch had 1 warning

July 13, 2022 at 8:08 PM

Version 0.0.1-SNAPSHOT

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

The Project Analysis has failed. More details available on the [Background Tasks](#) page.

0VulnerabilitiesSecurityA

1Security Hotspots0.0%ReviewedSecurity ReviewE

6h 57minDebt63Code SmellsMaintainabilityA

0.0%Coverage on 126 Lines to coverUnit Tests-0.0%Duplications on 289 LinesDuplicated Blocks0

ACTIVITY

Issues

July 13, 2022 at 8:08 PM

0.0.1-SNAPSHOT

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministration

Search for projects...

A

Elib

master

Last analysis of this Branch had 1 warning

July 13, 2022 at 8:08 PM

Version 0.0.1-SNAPSHOT

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

The Project Analysis has failed. More details available on the [Background Tasks](#) page.

My IssuesAll

Bulk Change

0 issues0 effort

Filters

Type

- Bug0
- Vulnerability0
- Code Smell0

Severity

- Blocker0Minor0
- Critical0Info0
- Major0

Scope

Resolution

Status

Security Category

Creation Date

Language

Rule

Tag

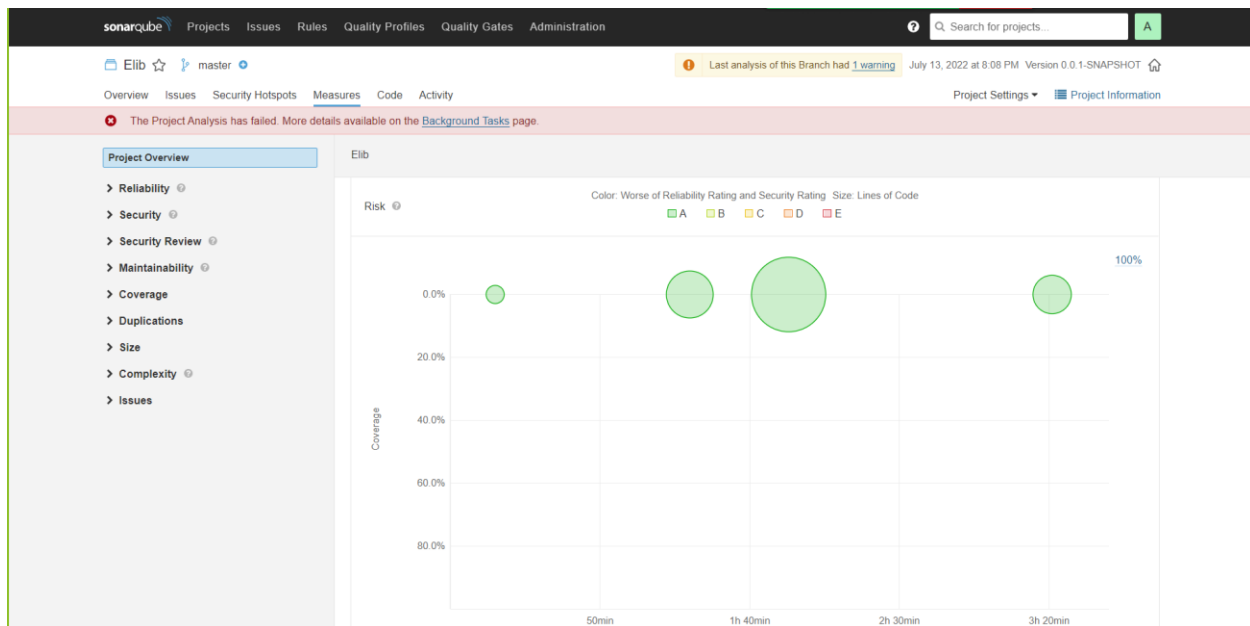
No Issues. Hooray!

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA

Community Edition - Version 9.5 (build 56709) - LGPL v3 - Community - Documentation - Plugins - Web API



Search for files...							
	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
Elib							
src	221	0	0	63	1	0.0%	0.0%
pom.xml	68	0	0	0	0	—	0.0%
2 of 2 shown							

## ➤ Generalization :-

Then we build a new class (abstract class) called (item) ,and we make an inheritance relation between book class and item class to make the book class inherit the item class. and we create a journal class also inherit the item class .

```

public class book extends item {
    String title;
    String author;
    String code;
    boolean borrowed;
    LocalDate borrowingDate;
    public book() {

    }
    public book(String t,String a,String c) {
        title=t;
        author=a;
        code=c;
        borrowed =false;
        borrowingDate=LocalDate.now();
    }
}

public class journal extends item{
    String journalTitle;
    LocalDate publishDate;
}

```

## ➤ Refactor:-

In the result of the sonarqube we have a bad smell code ,do we refactor the code by delete all the comment statement ,and delete a print statements .

And we did a move and extract for the function borrow:

```

public user() {
    System.out.println();
}

public user(String id,String name,String e,String add,String postal,String c) {}

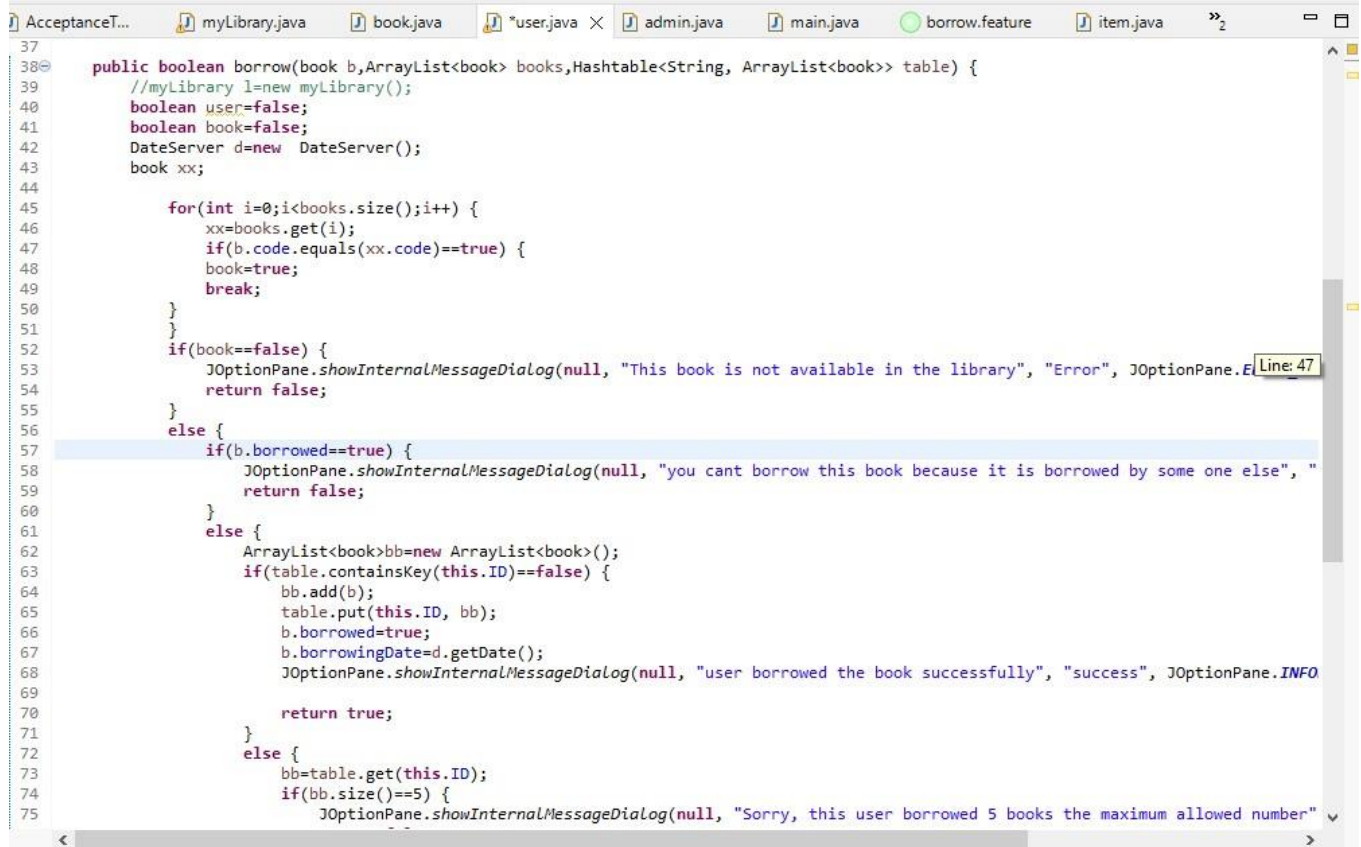
public boolean borrow(book b,myLibrary l) {
    //myLibrary l=new myLibrary();
    boolean user=false;
    boolean book=false;
    book xx;
    for (int i=0;i<myLibrary.registeredUsers.size();i++) {
        if (myLibrary.registeredUsers.get(i).ID.equals(this.ID)) {
            user=true;
            break;
        }
    }
    if(user==false) {
        JOptionPane.showInternalMessageDialog(null, "This user is not registered", "Error", JOptionPane.ERROR_MESSAGE);
        return false;
    }
    else {
        for(int i=0;i<l.books.size();i++) {
            xx=l.books.get(i);
            if(b.code.equals(xx.code)==true) {
                book=true;
                break;
            }
        }
        if(book==false) {
            JOptionPane.showInternalMessageDialog(null, "This book is not available in the library", "Error", JOptionPane.ERROR_ME
            return false;
        }
        else {
            if(b.borrowed==true) {
                JOptionPane.showInternalMessageDialog(null, "you cant borrow this book because it is borrowed by some one else", "

```

As we see the function borrow was in class user and take reference of book and reference of myLibrary and in its implementation it checks the user if he is registered in the library and check the book if it is available in the library and check if this book is borrowed and so on to make sure that borrowing process is successful.

```
AcceptanceT... myLibrary.java x book.java user.java admin.java main.java borrow.feature item.java »2
85     }
86
87
88     public boolean checkUser(user u) {
89
90         for (int i=0;i<registeredUsers.size();i++) {
91             if (registeredUsers.get(i).ID.equals(u.ID)) {
92
93                 return true;
94             }
95         }
96
97         JOptionPane.showMessageDialog(null, "This user is not registered before in the library", "Error", JOptionPane.ERROR_MESSAGE);
98         return false;
99     }
100 }
101
102     public boolean borrow(book b,user u) {
103
104         boolean f=checkUser(u);
105         if(f) {
106             f=lateBooks(u, 21);
107             if(f==false) {
108                 if(u.countFine(getFine())!=0) {
109                     JOptionPane.showMessageDialog(null, "Cant borrow book,you have fines", "Error", JOptionPane.ERROR_MESSAGE);
110                     return false;
111                 }
112                 f=u.borrow(b, books,borrowed);
113                 return f;
114             }
115             else return false;
116         }
117
118         else {
119             return f;
120         }
121     }
122
123
124
```

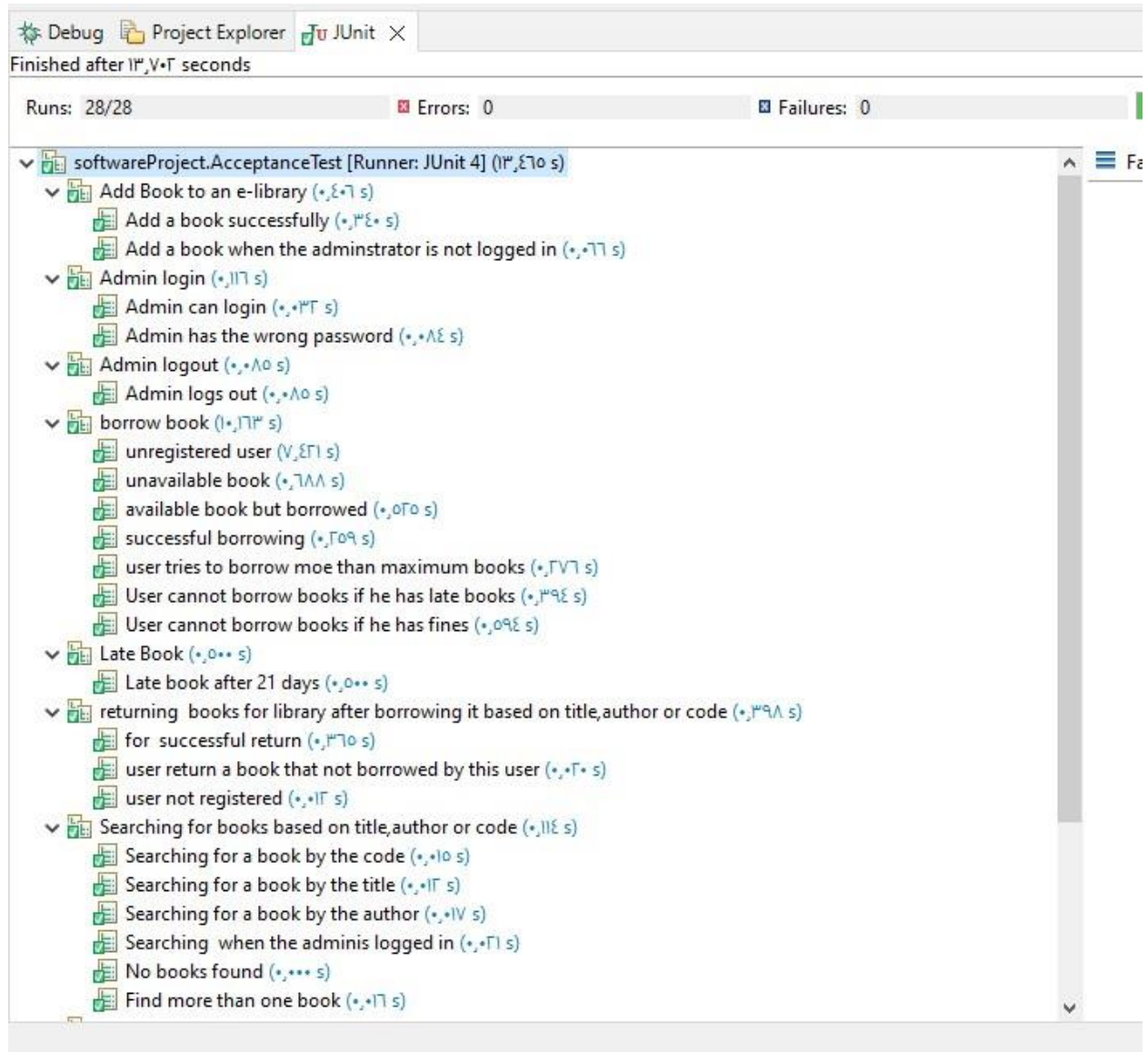
What we did as refactoring is we moved the declaration of borrow function to the library class and we inside borrow we call the user borrow function. Also we removed the part of checking the user if he is registered and extract it to another function checkUser function un class myLibrary.



```
37
38 public boolean borrow(book b, ArrayList<book> books, Hashtable<String, ArrayList<book>> table) {
39     //myLibrary l=new myLibrary();
40     boolean user=false;
41     boolean book=false;
42     DateServer d=new DateServer();
43     book xx;
44
45     for(int i=0;i<books.size();i++) {
46         xx=books.get(i);
47         if(b.code.equals(xx.code)==true) {
48             book=true;
49             break;
50         }
51     }
52     if(book==false) {
53         JOptionPane.showMessageDialog(null, "This book is not available in the library", "Error", JOptionPane.ERROR_MESSAGE);
54         return false;
55     }
56     else {
57         if(b.borrowed==true) {
58             JOptionPane.showMessageDialog(null, "you cant borrow this book because it is borrowed by some one else", "Error", "
59         }
60     }
61     else {
62         ArrayList<book>bb=new ArrayList<book>();
63         if(table.containsKey(this.ID)==false) {
64             bb.add(b);
65             table.put(this.ID, bb);
66             b.borrowed=true;
67             b.borrowingDate=d.getDate();
68             JOptionPane.showMessageDialog(null, "user borrowed the book successfully", "success", JOptionPane.INFORMATION_MESSAGE);
69         }
70         return true;
71     }
72     else {
73         bb=table.get(this.ID);
74         if(bb.size()==5) {
75             JOptionPane.showMessageDialog(null, "Sorry, this user borrowed 5 books the maximum allowed number"
```

And here is how function borrow become in class user. Now it takes an array list of books available in the library and takes hash table of borrowing books as parameters to the function.

In the end these photos shows the result for all features and Scenarios





Finished after 13,707 seconds

Runs: 28/28

Errors: 0

Failures: 0

- ✓ Admin logs out (0,000 s)
- ✓ borrow book (1,173 s)
  - ✓ unregistered user (0,001 s)
  - ✓ unavailable book (0,000 s)
  - ✓ available book but borrowed (0,000 s)
  - ✓ successful borrowing (0,000 s)
  - ✓ user tries to borrow more than maximum books (0,000 s)
  - ✓ User cannot borrow books if he has late books (0,000 s)
  - ✓ User cannot borrow books if he has fines (0,000 s)
- ✓ Late Book (0,000 s)
  - ✓ Late book after 21 days (0,000 s)
- ✓ returning books for library after borrowing it based on title,author or code (0,000 s)
  - ✓ for successful return (0,000 s)
  - ✓ user return a book that not borrowed by this user (0,000 s)
  - ✓ user not registered (0,000 s)
- ✓ Searching for books based on title,author or code (0,000 s)
  - ✓ Searching for a book by the code (0,000 s)
  - ✓ Searching for a book by the title (0,000 s)
  - ✓ Searching for a book by the author (0,000 s)
  - ✓ Searching when the admin is logged in (0,000 s)
  - ✓ No books found (0,000 s)
  - ✓ Find more than one book (0,000 s)
- ✓ Unregister user (0,000 s)
  - ✓ successful unregister (0,000 s)
  - ✓ user has borrowed books (0,000 s)
  - ✓ user has fine unpaid (0,000 s)
- ✓ Register user (0,000 s)
  - ✓ Admin is not logged in (0,000 s)
  - ✓ user is already registered (0,000 s)
  - ✓ user not registered before (0,000 s)

And finally here is our sonar report for final project:



## Passed

All conditions passed.

### New Code

### Overall Code

2 Bugs

Reliability E

0 Vulnerabilities

Security A

3 Security Hotspots

0.0% Reviewed

Security Review E

2d 6h Debt

154 Code Smells

Maintainability B

0.0%  
Coverage on 336 Lines to cover

-  
Unit Tests

4.5%  
Duplications on 605 Lines

2  
Duplicated Blocks