Software Engineering

Online Library

Dr.Haya Samaana

Students :

Sarah Hinno 11926569

Marah Direeni 11923808

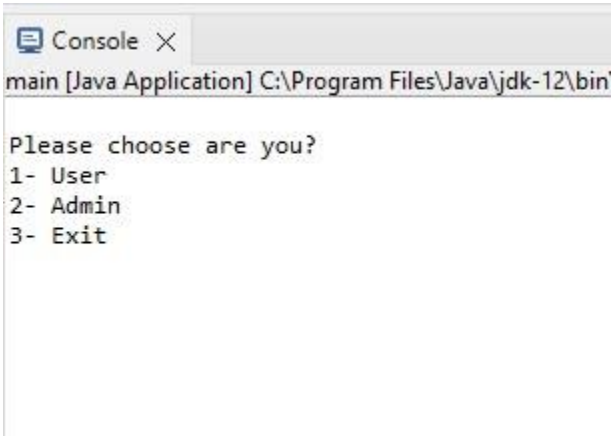
Yafa Nada 11924868

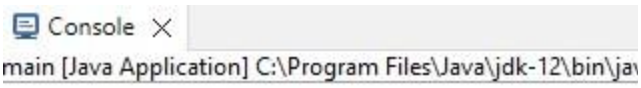
➢Introduction :-

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 :-

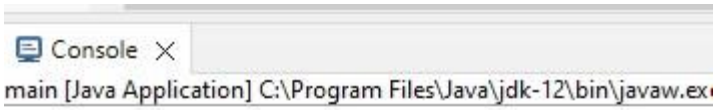
Here we have the main that the user and the admin deal with :-






```
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
S
```

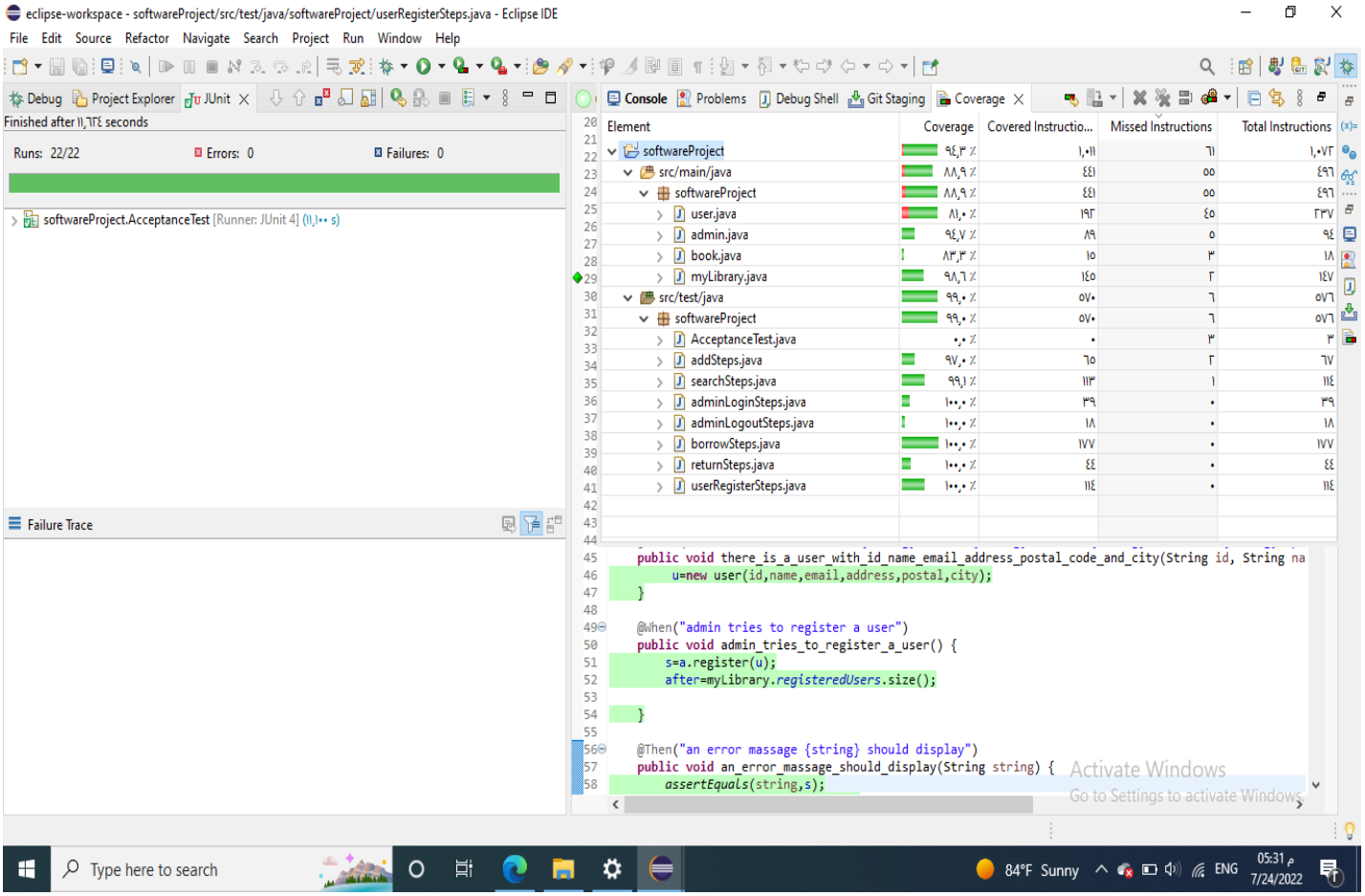
```
Please choose are you?
1- User
2- Admin
3- Exit
1
Enter your ID
1234567
please choose a number:
1- Borrow book
2- Return book
3- Search book
4- print all books i have borrowed
5- Back to main menue
```

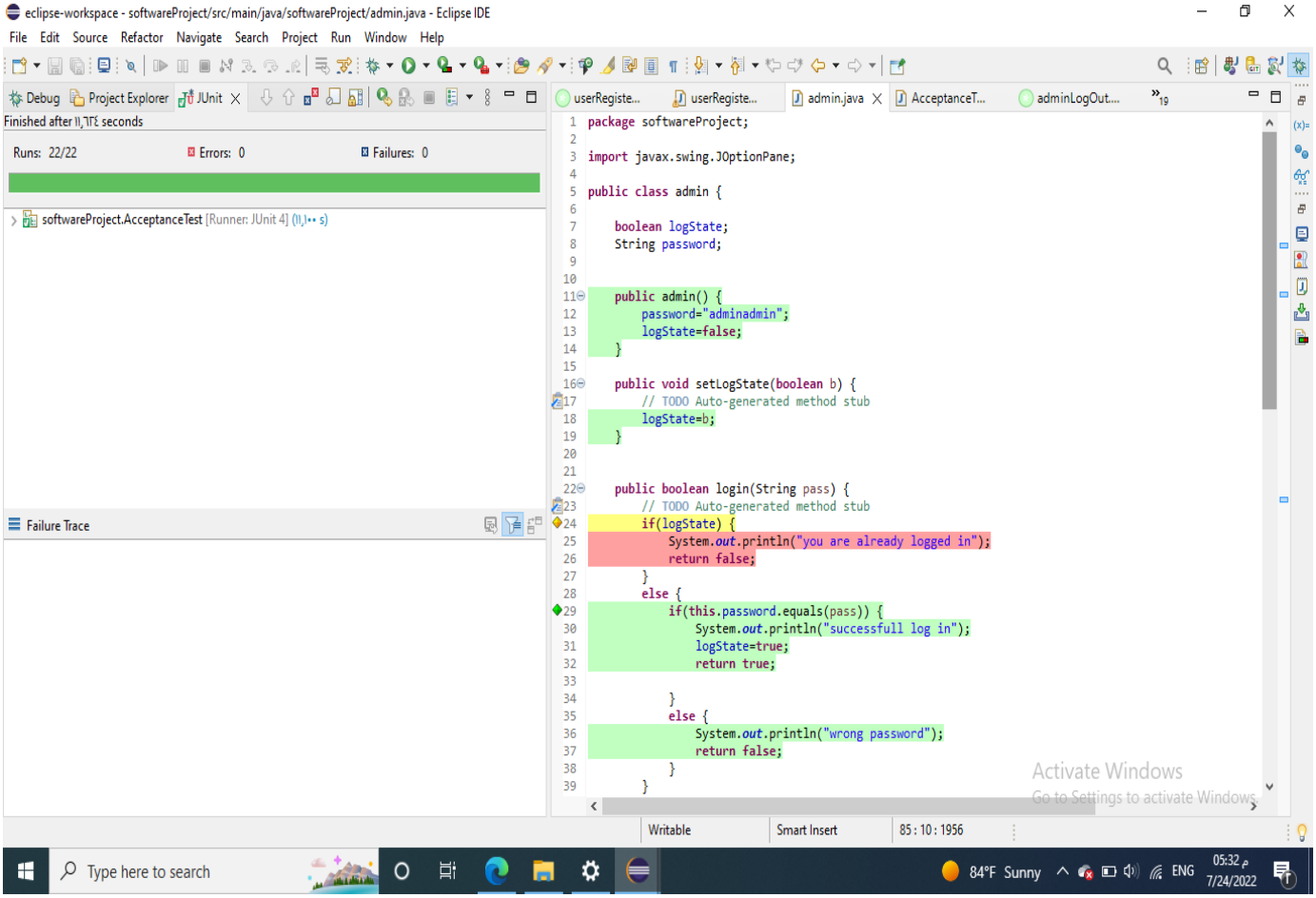


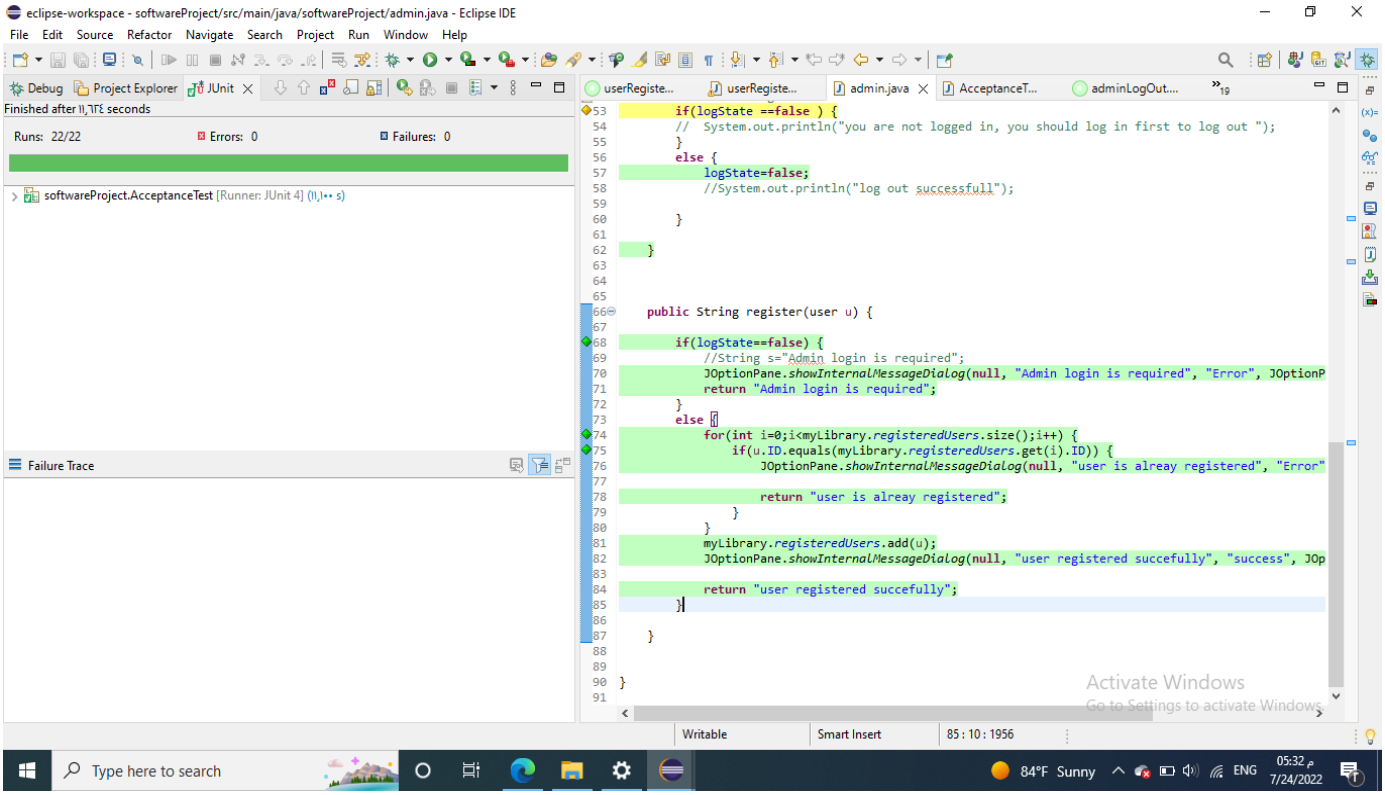

```
Please choose are you?
1- User
2- Admin
3- Exit
2
Please Enter the password:
adminadmin
successfull log in
1- Register user
2- Search book
3- Add book
4- Unregister user
5- print all registered user
6- print all books in the library
7- Back to main menue
7
Please choose are you?
1- User
2- Admin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

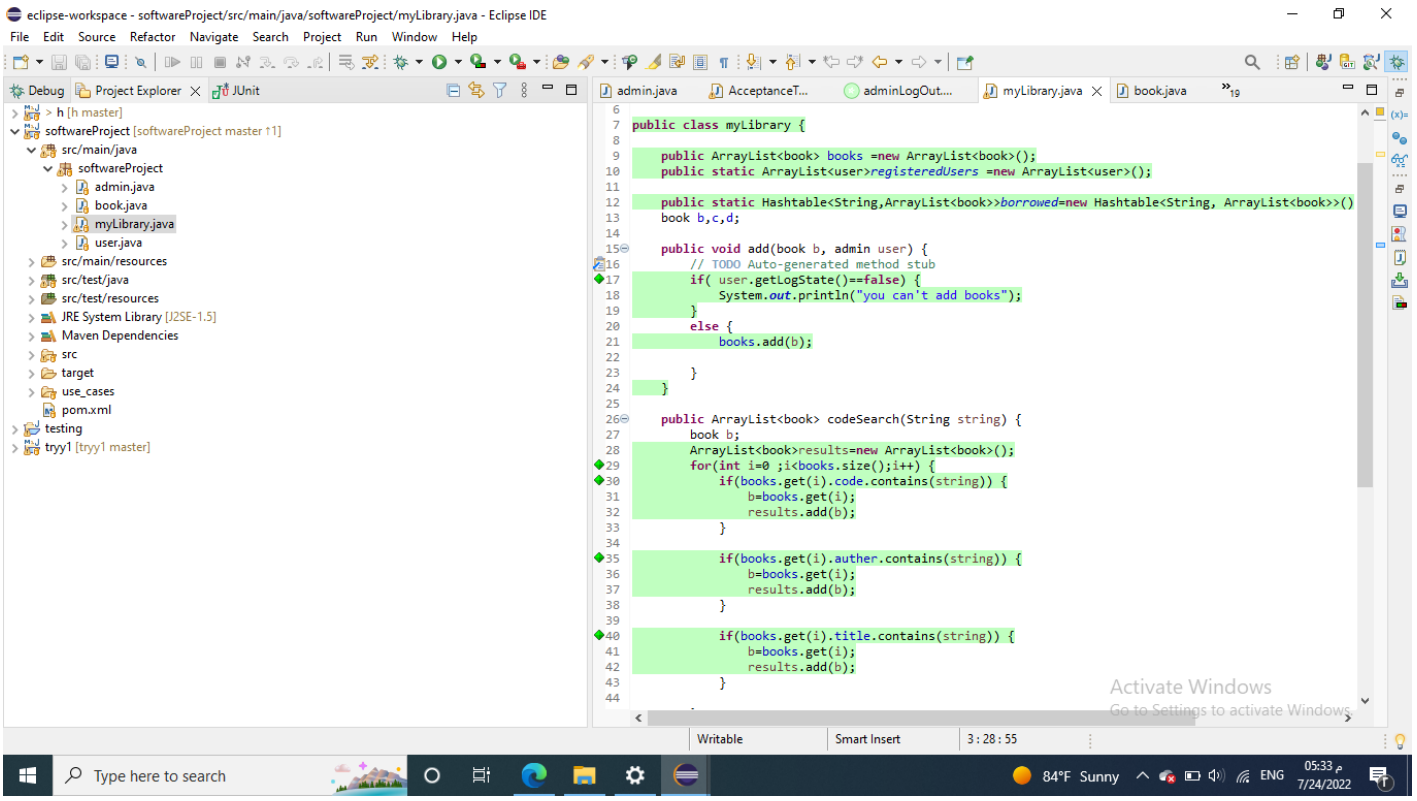
➢Coverage :-

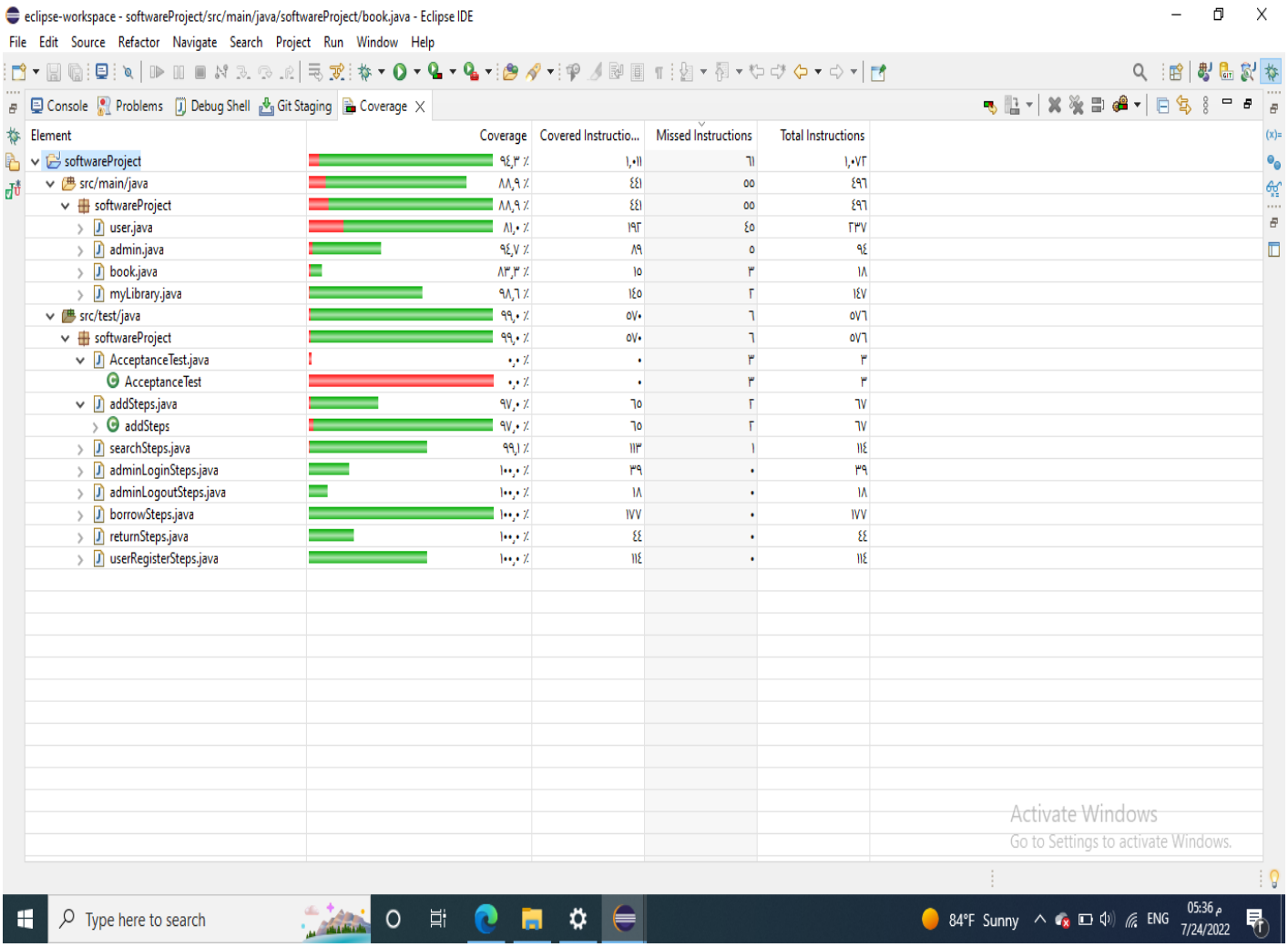
Then we make a coverage for our code before we build the main and the result we have got :-




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
Finished after ١١,٦٤٤ seconds
Runs: 22/22
Errors: 0
Failures: 0
softwareProject.AcceptanceTest [Runner: JUnit 4] (١١,١•• s)
Failure Trace
userRegiste... userRegiste... admin.java AcceptanceT... adminLogOut....
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 }
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
85 : 10 : 1956
Type here to search
84°F Sunny
ENG
05:32 م
7/24/2022


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
Finished after 11.765 seconds
Runs: 22/22 Errors: 0 Failures: 0
softwareProject.AcceptanceTest [Runner: JUnit 4] (11.1** s)
Failure Trace
userRegiste... userRegiste... admin.java AcceptanceT... adminLogOut...
53 if(logState ==false) {
54 // System.out.println("you are not logged in, you should log in first to log out ");
55 }
56 else {
57 logState=false;
58 //System.out.println("log out successfull");
59
60 }
61
62 }
63
64
65
66 public String register(user u) {
67
68 if(logState==false) {
69 //String s="Admin login is required";
70 JOptionPane.showInternalMessageDialog(null, "Admin login is required", "Error", JOptionP
71 return "Admin login is required";
72 }
73 else {
74 for(int i=0;i<myLibrary.registeredUsers.size();i++) {
75 if(u.ID.equals(myLibrary.registeredUsers.get(i).ID)) {
76 JOptionPane.showInternalMessageDialog(null, "user is alreay registered", "Error"
77
78 return "user is alreay registered";
79 }
80 }
81 myLibrary.registeredUsers.add(u);
82 JOptionPane.showInternalMessageDialog(null, "user registered succefully", "success", JOp
83
84 return "user registered succefully";
85 }
86
87 }
88
89
90 }
91
Activate Windows
Go to Settings to activate Windows.
Writable Smart Insert 85 : 10 : 1956
Type here to search
84°F Sunny
ENG
05:32 م
7/24/2022

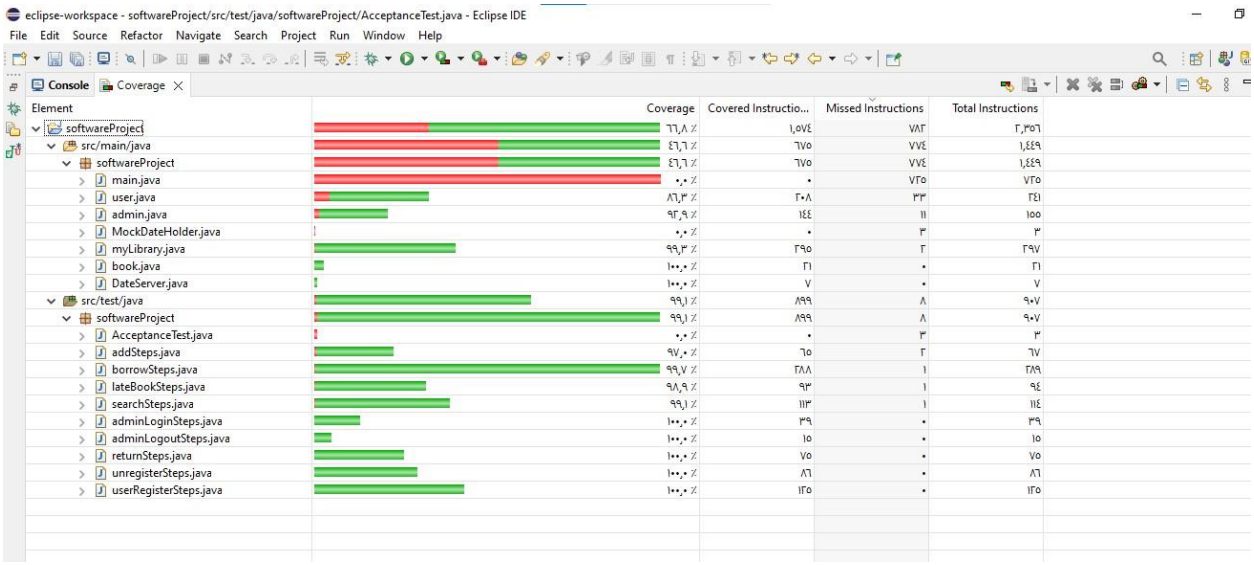

eclipse-workspace - softwareProject/src/main/java/softwareProject/myLibrary.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Project Explorer JUnit
h [h master]
softwareProject [softwareProject master ↑1]
src/main/java
softwareProject
admin.java
book.java
myLibrary.java
user.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
use_cases
pom.xml
testing
tryy1 [tryy1 master]
admin.java AcceptanceT... adminLogOut... myLibrary.java book.java
6
7 public class myLibrary {
8
9 public ArrayList<book> books =new ArrayList<book>();
10 public static ArrayList<user>registeredUsers =new ArrayList<user>();
11
12 public static Hashtable<String,ArrayList<book>>borrowed=new Hashtable<String, ArrayList<book>>()
13 book b,c,d;
14
15 public void add(book b, admin user) {
16 // TODO Auto-generated method stub
17 if(user.getLogState()==false) {
18 System.out.println("you can't add books");
19 }
20 else {
21 books.add(b);
22
23 }
24 }
25
26 public ArrayList<book> codeSearch(String string) {
27 book b;
28 ArrayList<book>results=new ArrayList<book>();
29 for(int i=0 ;i<books.size();i++) {
30 if(books.get(i).code.contains(string)) {
31 b=books.get(i);
32 results.add(b);
33 }
34
35 if(books.get(i).auther.contains(string)) {
36 b=books.get(i);
37 results.add(b);
38 }
39
40 if(books.get(i).title.contains(string)) {
41 b=books.get(i);
42 results.add(b);
43 }
44
Activate Windows
Go to Settings to activate Windows.
Writable Smart Insert 3 : 28 : 55
Type here to search
84°F Sunny
ENG
05:33 م
7/24/2022



As we see the percentage of coverage is good .

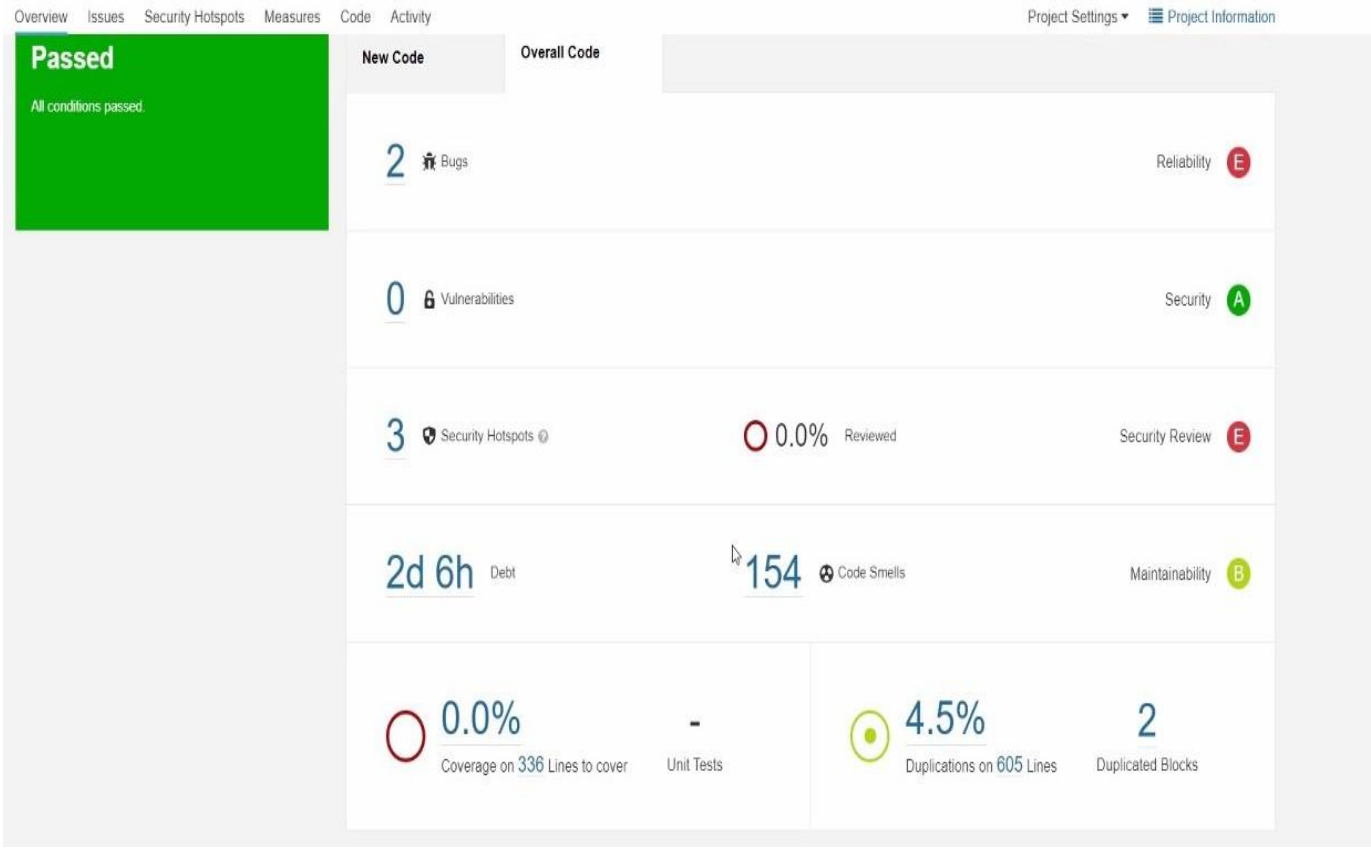
The picture bellow is the percentage of coverage is less than the coverage above because of the main that we can't implement a test cases for it .and it is the way to make the user enabled with the design ,it's like a GUI .

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



➢SonarQube:-

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



➢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 auther;
    String code;
    boolean borrowed;
    LocalDate borrowingDate;
    public book() {

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

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

}
```

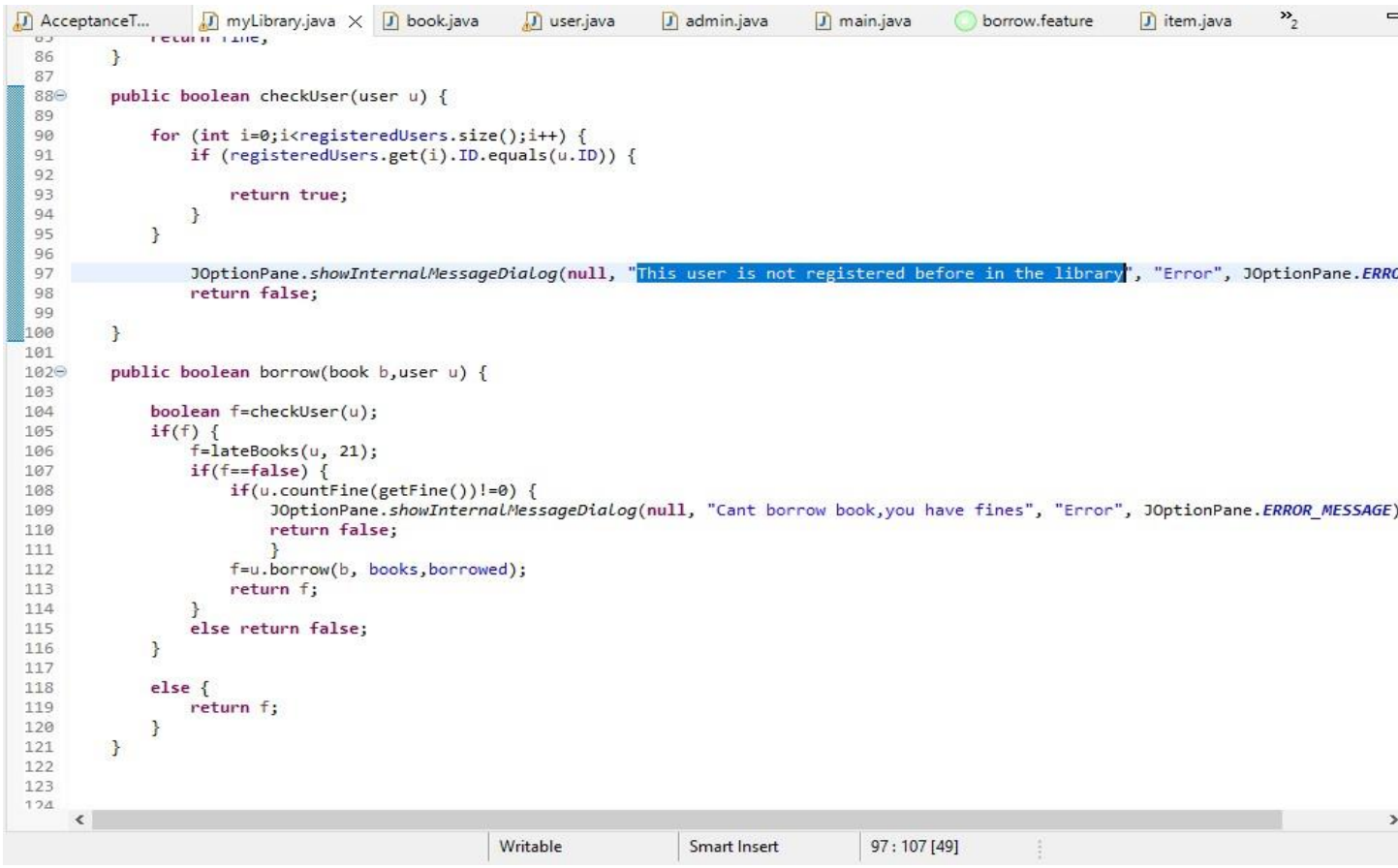
➢Refactor:-

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 .

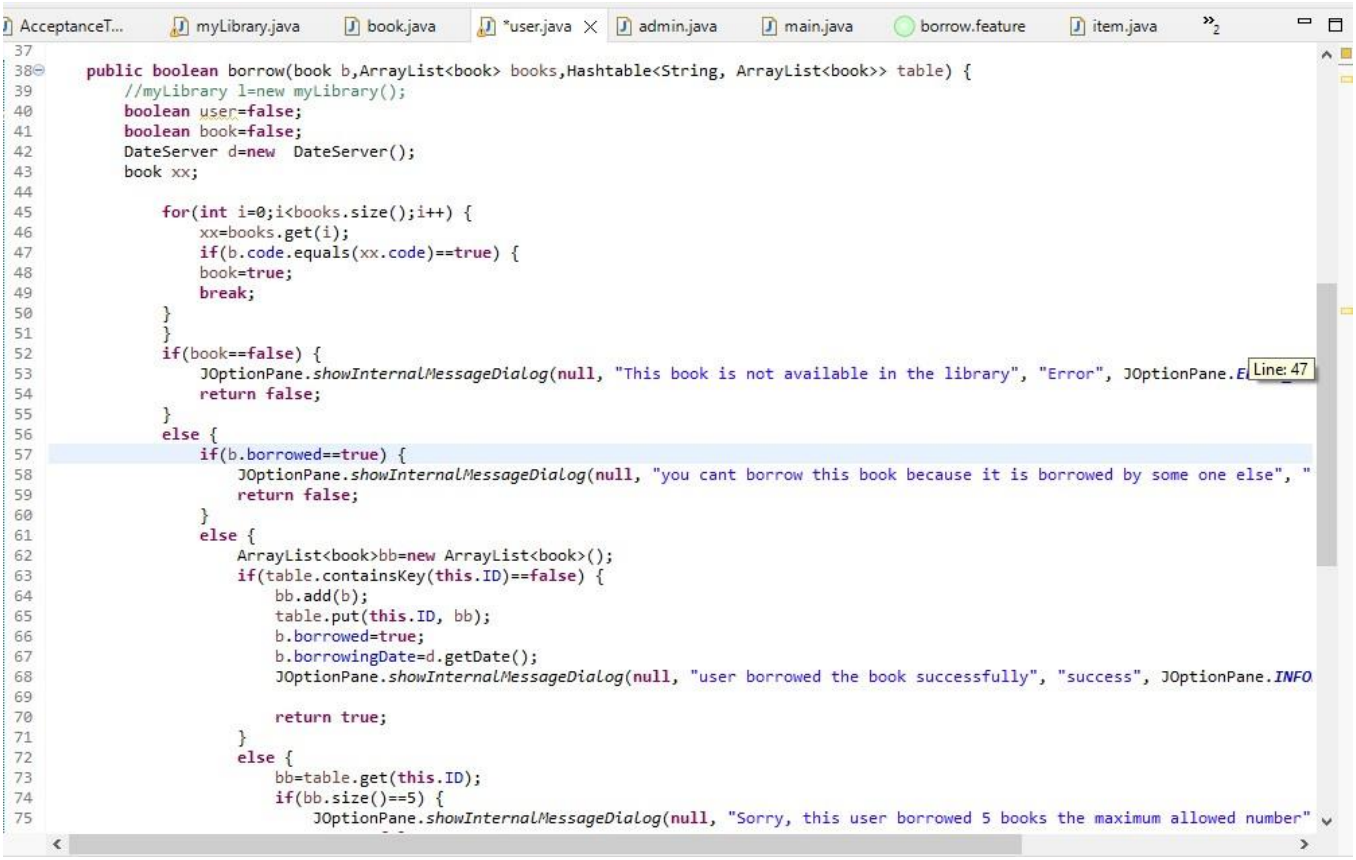
And we did a move and extract for the function borrow:

```
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.



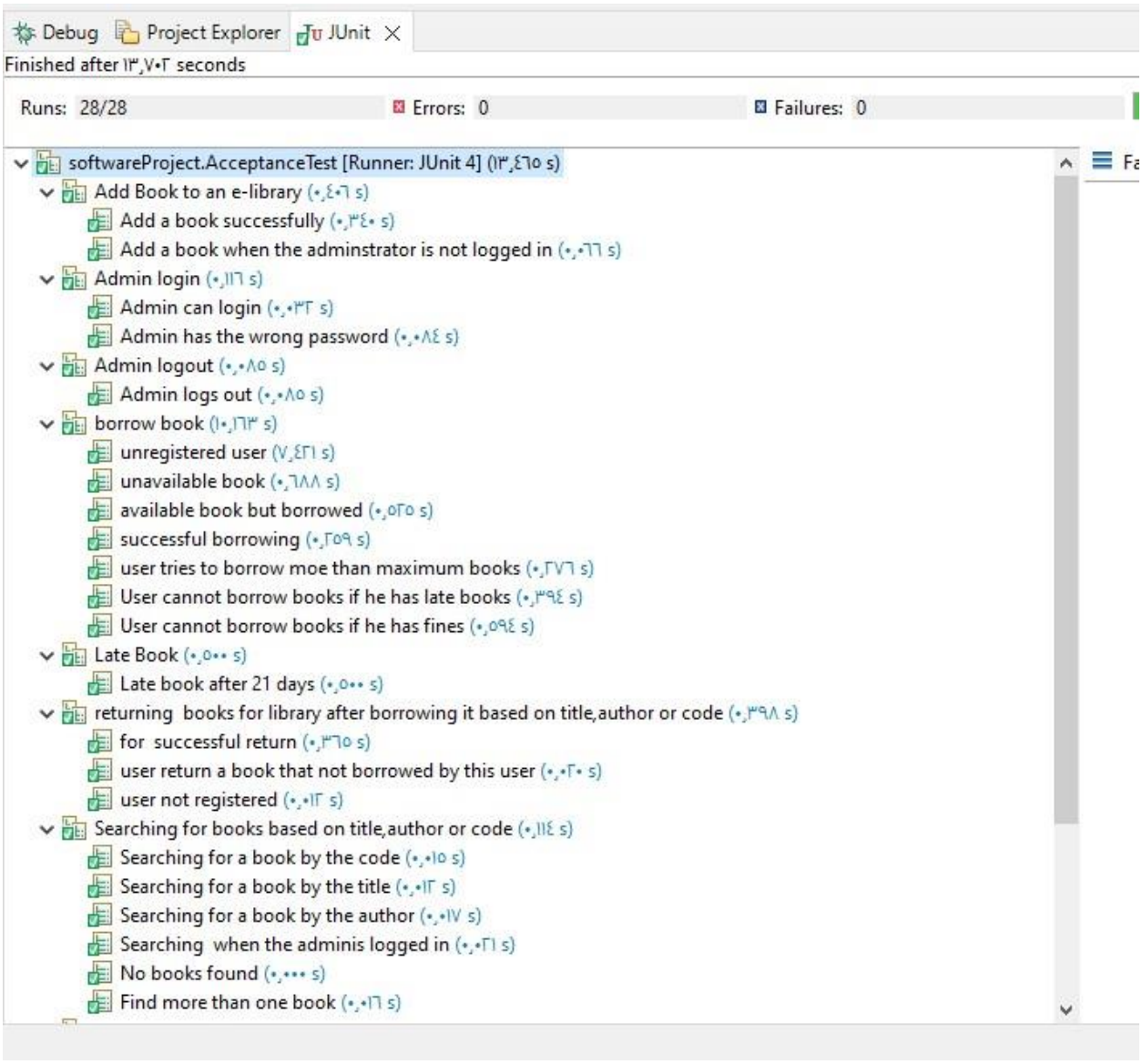
What we did as refactoring is we moved the declaration of borrow function to the library class and we inside borrow we call the user borrow function. Also we removed the part of checking the user if he is registered and extract it to another function checkUser function un class myLibrary.

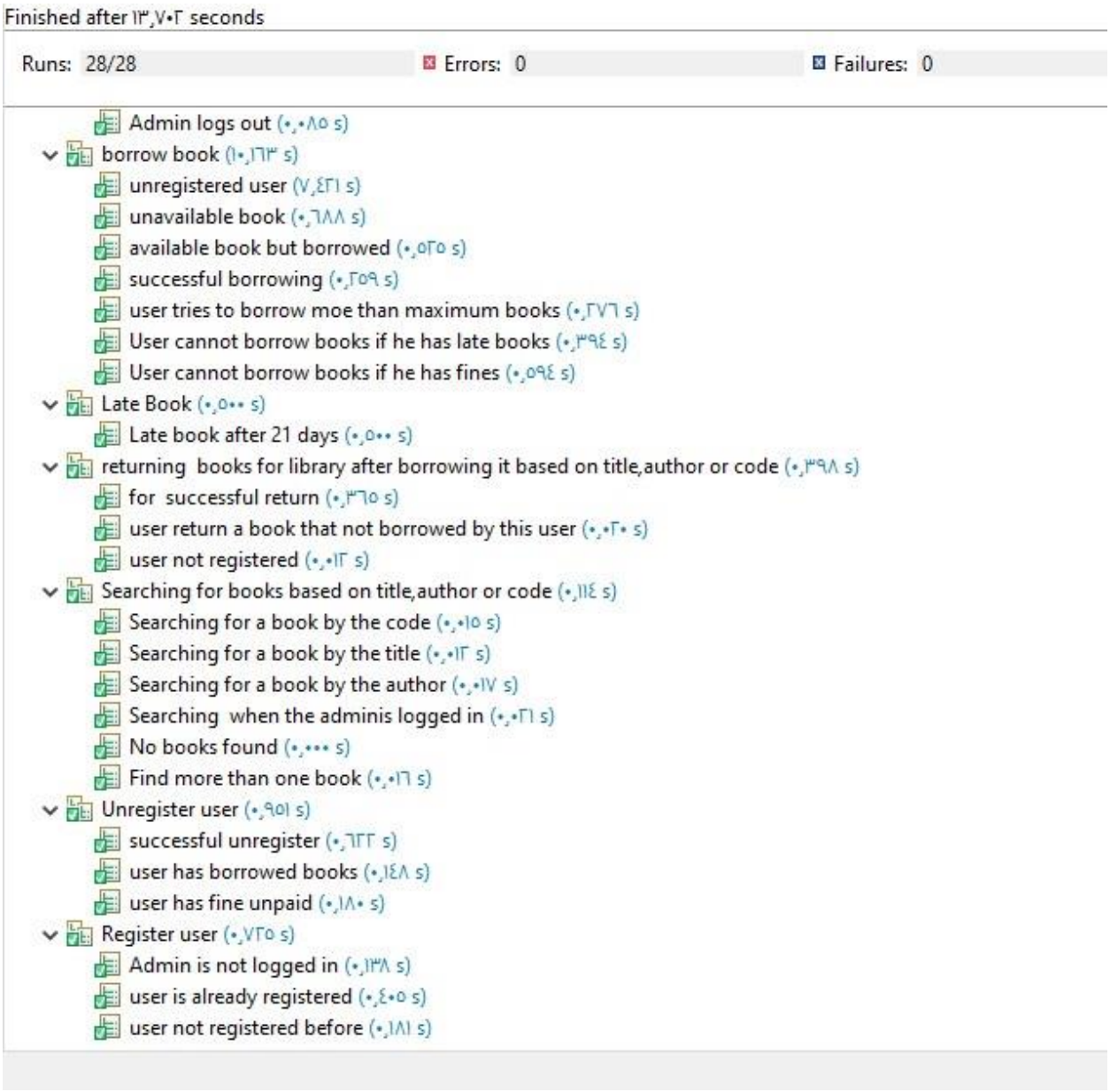



```java
public boolean borrow(book b,ArrayList<book> books,Hashtable<String, ArrayList<book>> table) {
    //myLibrary l=new myLibrary();
    boolean user=false;
    boolean book=false;
    DateServer d=new  DateServer();
    book xx;

        for(int i=0;i<books.size();i++) {
            xx=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.E
            return false;
        }
        else {
            if(b.borrowed==true) {
                JOptionPane.showInternalMessageDialog(null, "you cant borrow this book because it is borrowed by some one else", "
                return false;
            }
            else {
                ArrayList<book>bb=new ArrayList<book>();
                if(table.containsKey(this.ID)==false) {
                    bb.add(b);
                    table.put(this.ID, bb);
                    b.borrowed=true;
                    b.borrowingDate=d.getDate();
                    JOptionPane.showInternalMessageDialog(null, "user borrowed the book successfully", "success", JOptionPane.INFO

                    return true;
                }
                else {
                    bb=table.get(this.ID);
                    if(bb.size()==5) {
                        JOptionPane.showInternalMessageDialog(null, "Sorry, this user borrowed 5 books the maximum allowed number"
```

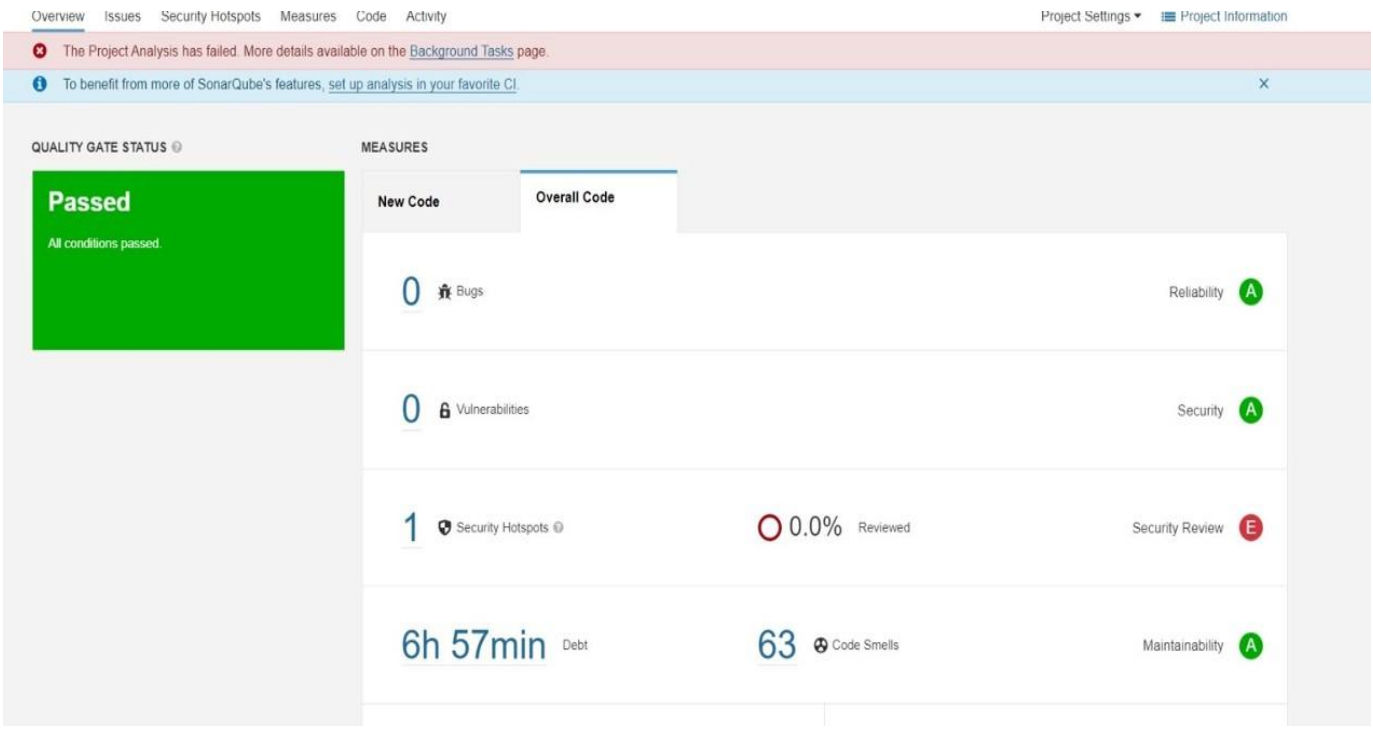
And here is how function borrow become in class user. Now it takes an array list of books available in the library and takes hash table of borrowing books as parameters to the function.

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





And finally here is our sonar report for final project:


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