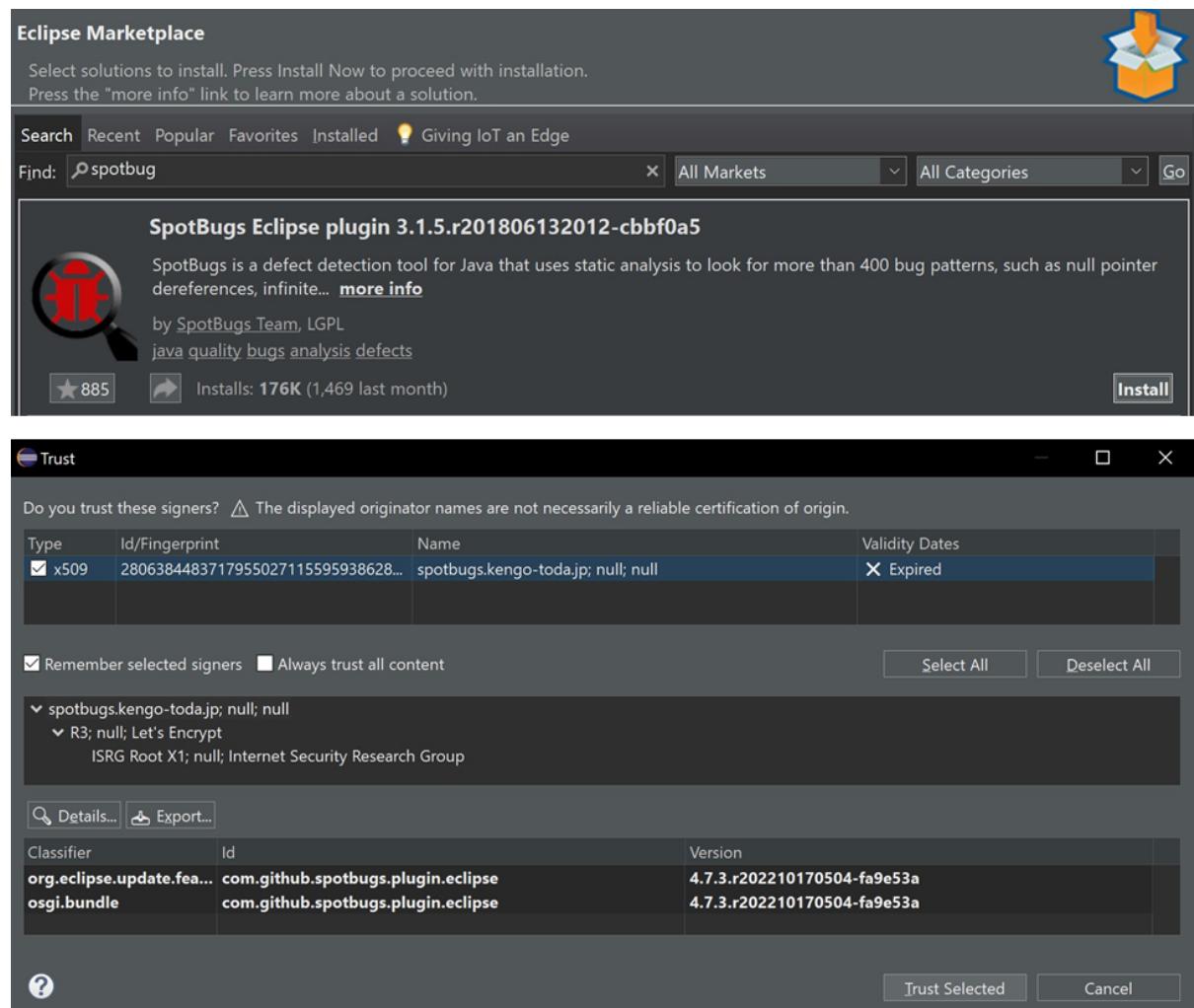


Car Rental System Analysis

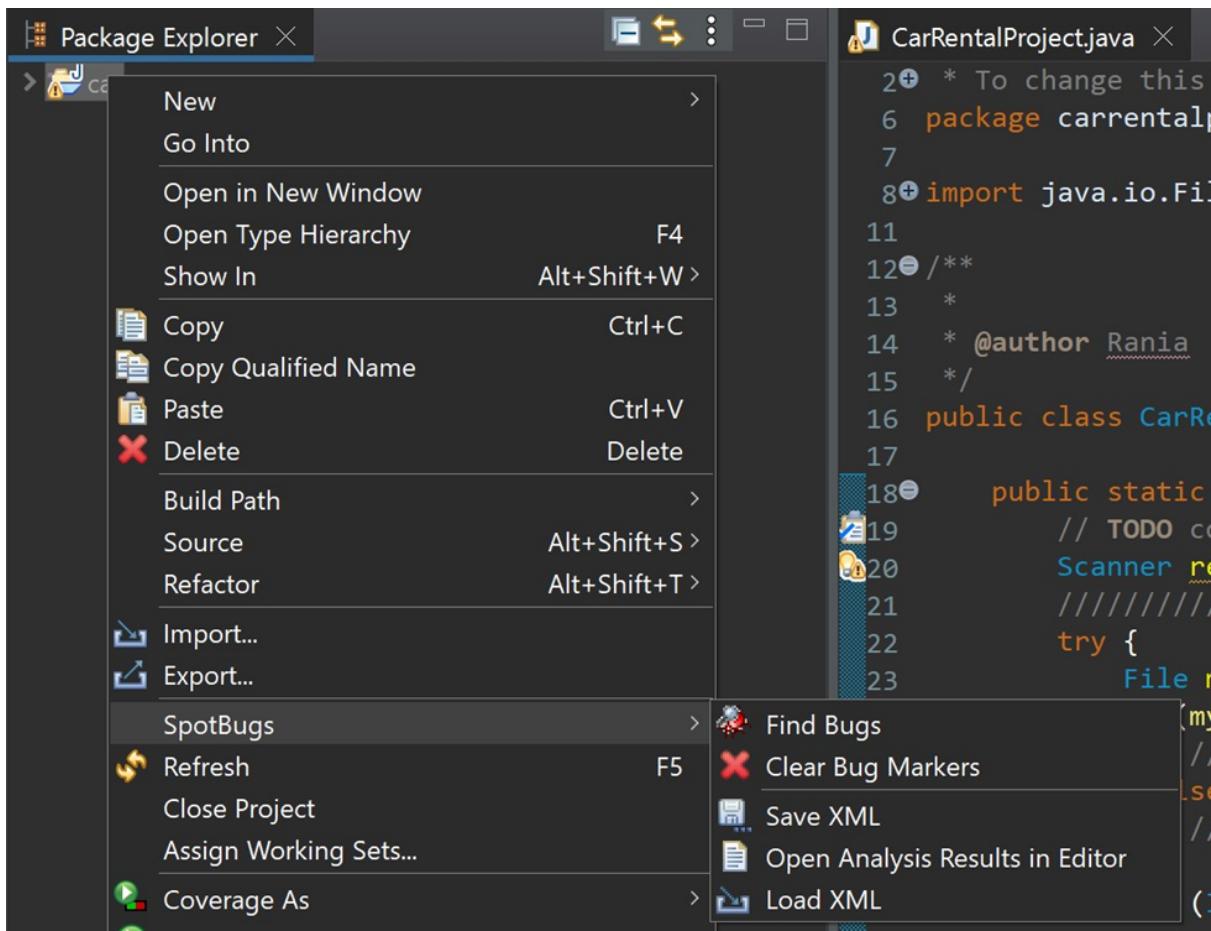
Static Analysis

To perform static analysis on the car rental system, I will utilize the SpotBugs tool. SpotBugs is a popular static code analysis tool that helps identify potential bugs, security vulnerabilities, and coding errors in the source code.

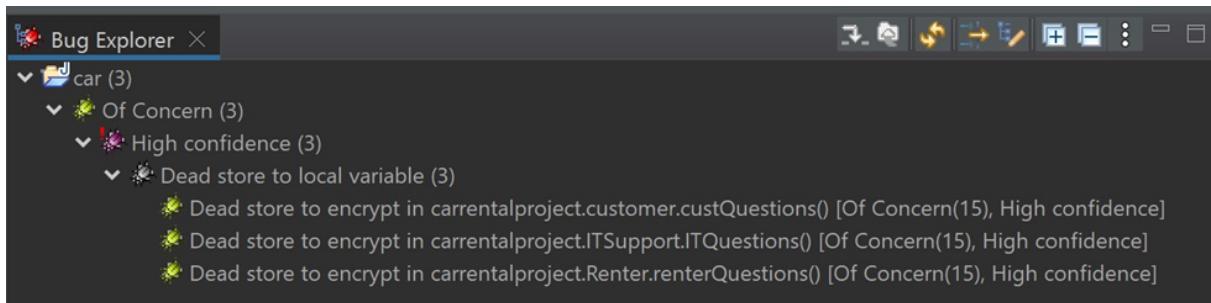
The first step is to install SpotBugs from the Eclipse Marketplace or any other suitable method.



Once installed, I will configure SpotBugs to analyze the program code of the car rental system.



The bugs of concern found using SpotBugs tool



Case 1: in customers class

CarRentalProject.java

```

35● public void custQuestions() throws IOException, Int
36     Scanner scan = new Scanner(System.in);
37     //-----
38     System.out.print("\nEnter your name: ");
39     Cname = scan.next(); //validation
40     while (!Cname.matches("[a-zA-Z]+[a-zA-Z]+\\s[")
41         System.out.println("Enter correct name: ");
42         Cname = scan.next();
43     }
44     //-----
45     System.out.print("Enter your age: ");
46     Cage = scan.nextInt();
47     if (Cage < 18) {
48         System.out.println("\nYou can't create an a
49         System.exit(0);
50     }
51     //-----
52     System.out.print("Enter your ID num: ");
53     Cid = scan.nextInt();
54     System.out.print("Enter your key: ");
55     Ckey = scan.nextInt();
56     AES encrypt = new AES();
57     String check = "";
58     check = encrypt.getPass(Cid, Ckey);
59     byte[] bytes = Files.readAllBytes(Paths.get("cu
60     String s = new String(bytes);
61     // Check if the name is contained
62     if (s.indexOf(check) != -1) {
63         //System.out.println("true");
64     } else {
65         System.out.println("wrong!");
66         System.exit(0);
67     }
68     //-----
69     if (s.indexOf(Cid) != -1) {
70         System.out.println("");
71     } else {

```

Bug Info

customer.java: 56

Navigation

Dead store to encrypt in carrentalproject.customer.custQuestions()
Local variable named encrypt

Bug: Dead store to encrypt in carrentalproject.customer.custQuestions()

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because SpotBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

Rank: Of Concern (15), **confidence:** High
Pattern: DLS_DEAD_LOCAL_STORE
Type: DLS, **Category:** STYLE (Dodgy code)

XML output:

```

<BugInstance type="DLS_DEAD_LOCAL_STORE" priority="1" rank="15" abbrev="<Class classname="carrentalproject.customer">
    <SourceLine classname="carrentalproject.customer" sourcefile="custom
</Class>
<Method classname="carrentalproject.customer" name="custQuestions" sig=
    <SourceLine classname="carrentalproject.customer" start="36" end="29"
</Method>
<LocalVariable name="encrypt" register="2" pc="135" role="LOCAL_VARIAB
<SourceLine classname="carrentalproject.customer" start="56" end="56"
<SourceLine classname="carrentalproject.customer" start="56" end="56"
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.DEA
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.LOCA
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.METH
</BugInstance>

```

Case 2: in renters class

CarRentalProject.java

```

14● public void renterQuestions () throws IOException,
15     Scanner scan = new Scanner (System.in);
16     char answer;
17     //-----
18     System.out.print("\nEnter your name: ");
19     String Rname= scan.next();
20     while (!Rname.matches("[a-zA-Z]+[a-zA-Z]+\\s[")
21         System.out.println(" Enter correct name: ");
22         Rname= scan.next();
23     }
24     //-----
25     System.out.print("Enter your age: ");
26     int Rage = scan.nextInt();
27     if (Rage<18) {
28         System.out.println("\nYou can't create an a
29         System.exit(0);
30     }
31     //-----
32     System.out.print("Enter your ID num: ");
33     String Rid = scan.nextInt();
34     System.out.print("Enter your key: ");
35     String Rkey =scan.nextInt();
36     AES encrypt = new AES();
37     String check;
38     check = encrypt.getPass(Rid,Rkey);
39     byte[] bytes = Files.readAllBytes(Paths.get("re
40     String s = new String(bytes);
41     // Check if the name is contained
42     if(s.indexOf(check) != -1){
43         System.out.println("");
44     } else {
45         System.out.println("wrong!");
46         System.exit(0);
47     }
48     //-----
49     if(s.indexOf(Rid) != -1){

```

Bug Info

Renter.java: 37

Navigation

Dead store to encrypt in carrentalproject.Renter.renterQuestions()
Local variable named encrypt

Bug: Dead store to encrypt in carrentalproject.Renter.renterQuestions()

This instruction assigns a value to a local variable, but the value is not read or used in any subsequent instruction. Often, this indicates an error, because the value computed is never used.

Note that Sun's javac compiler often generates dead stores for final local variables. Because SpotBugs is a bytecode-based tool, there is no easy way to eliminate these false positives.

Rank: Of Concern (15), **confidence:** High
Pattern: DLS_DEAD_LOCAL_STORE
Type: DLS, **Category:** STYLE (Dodgy code)

XML output:

```

<BugInstance type="DLS_DEAD_LOCAL_STORE" priority="1" rank="15" abbrev="<Class classname="carrentalproject.Renter">
    <SourceLine classname="carrentalproject.Renter" sourcefile="Renter.j
</Class>
<Method classname="carrentalproject.Renter" name="renterQuestions" sig=
    <SourceLine classname="carrentalproject.Renter" start="16" end="247"
</Method>
<LocalVariable name="encrypt" register="7" pc="119" role="LOCAL_VARIAB
<SourceLine classname="carrentalproject.Renter" start="37" end="37" st
<SourceLine classname="carrentalproject.Renter" start="37" end="37" st
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.DEA
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.LOCA
<Property name="edu.umd.cs.findbugs.detect.DeadLocalStoreProperty.METH
</BugInstance>

```

Case 3: in ITsupport class

The screenshot shows the Eclipse IDE interface with three open files: customer.java, Renter.java, and ITSupport.java. The ITSupport.java file is the active tab, displaying Java code related to car rental system logic. A right-hand panel displays a 'Bug Info' window from the SpotBugs tool. The bug is identified at line 42 of ITSupport.java, where a local variable 'encrypt' is assigned but never used. The message states: 'Dead store to encrypt in carrentalproject.ITSupport.ITQuestions()'. It notes that Sun's javac compiler often generates dead stores for final local variables. The bug is categorized as 'High' concern, pattern 'DLS_DEAD_LOCAL_STORE', and type 'DLS_STYLE'.

Findings

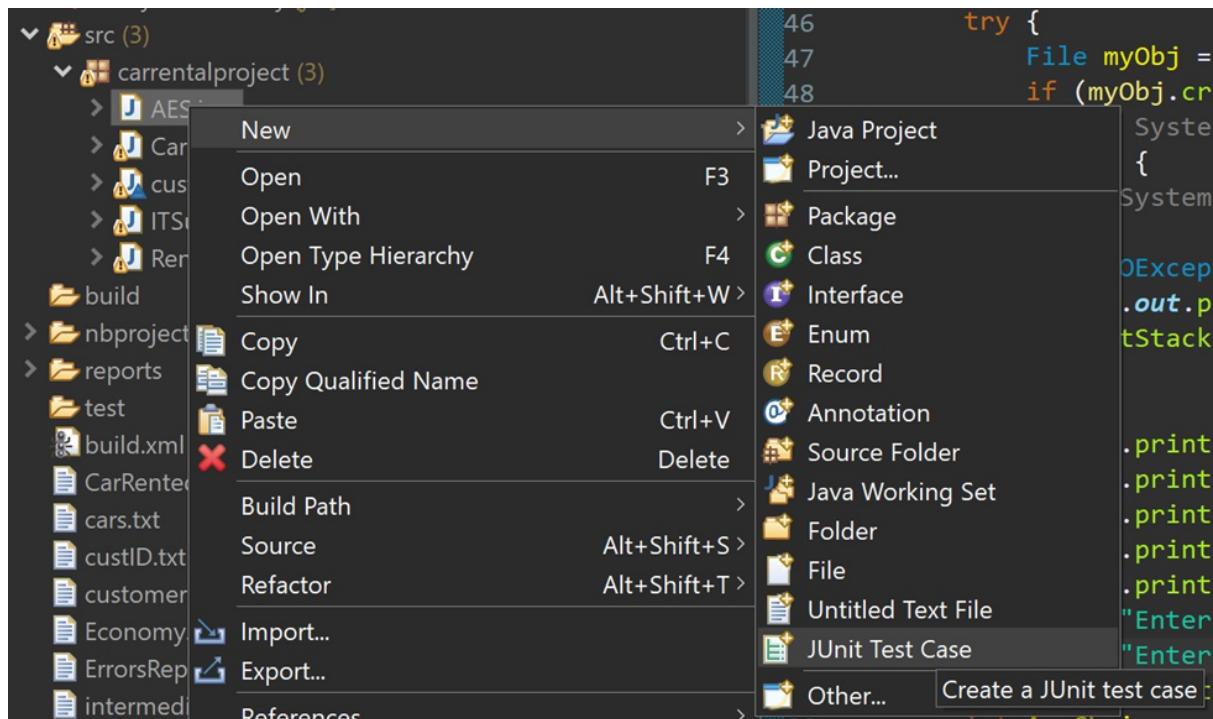
The following findings were discovered through the analysis:

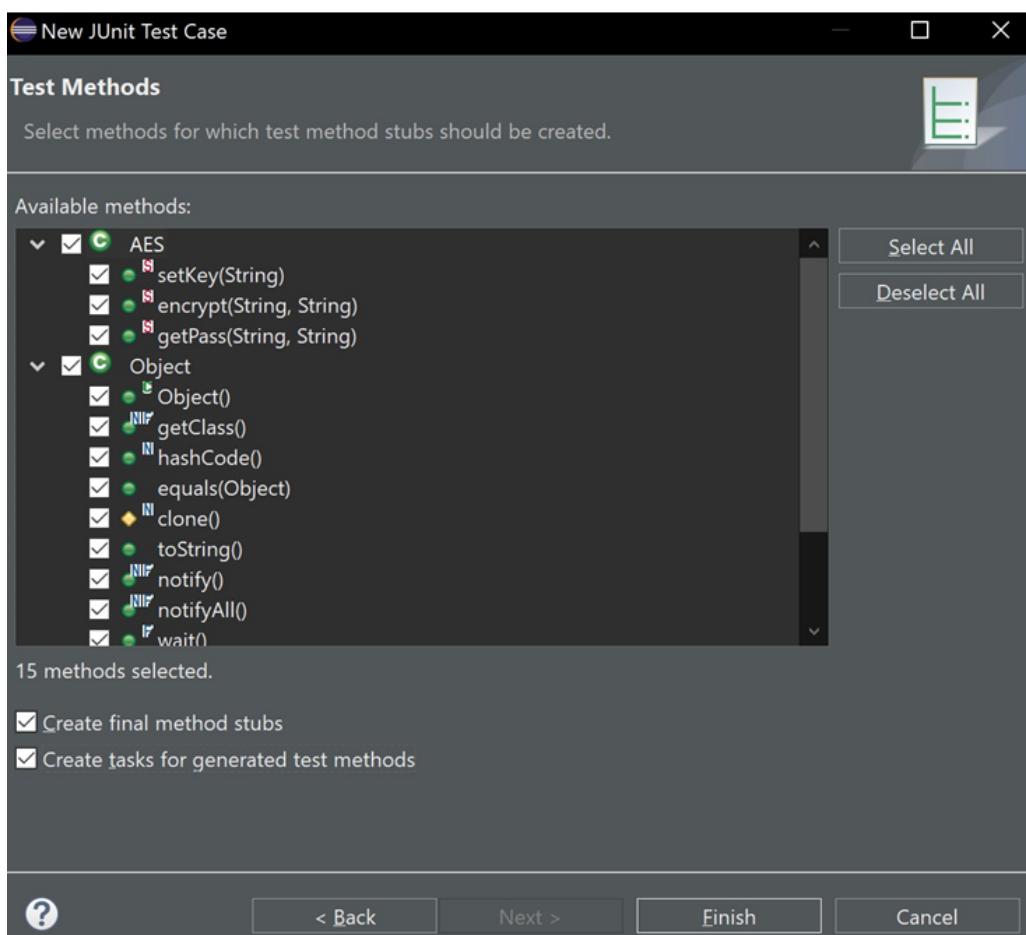
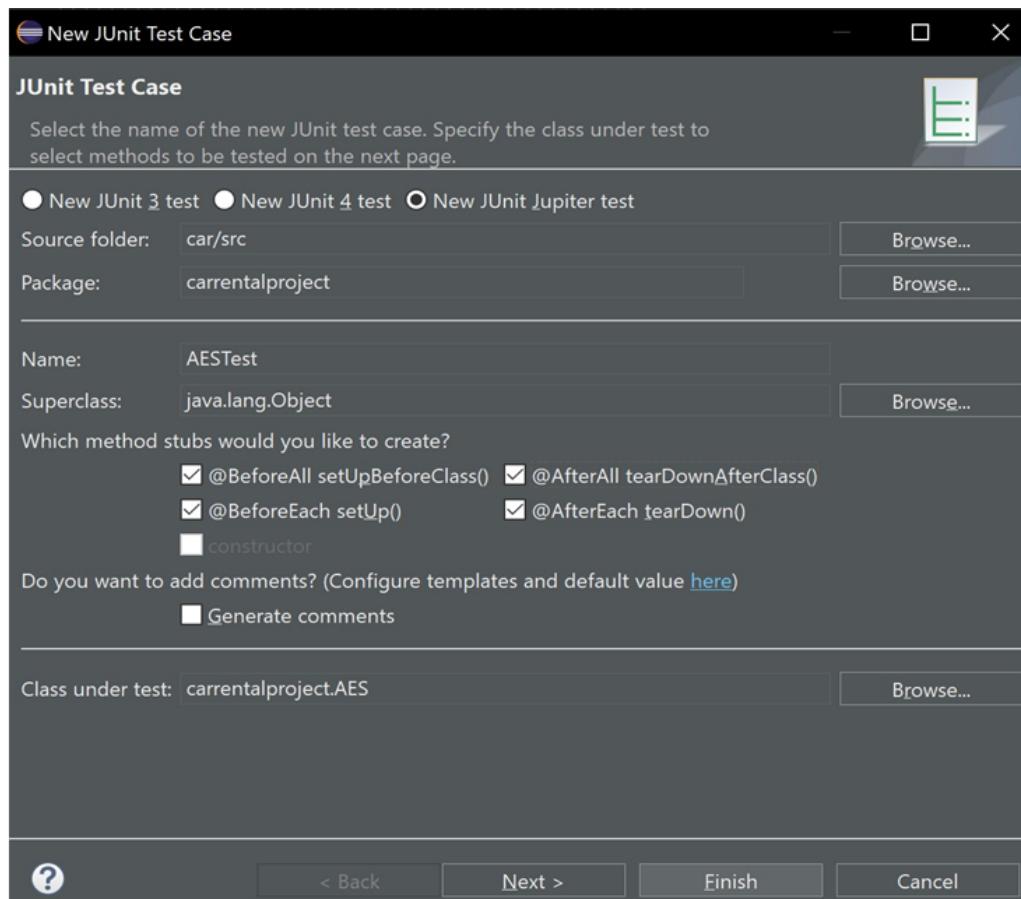
1. In the 3 classes: SpotBugs identified a bug related to a dead store to encrypt.
This bug indicates that the encryption operation performed in the code is redundant and does not contribute to the system's functionality. It is recommended to either remove the encryption operation if unnecessary or modify the code to utilize the encrypted value appropriately.

Dynamic Analysis

Dynamic testing was performed on the car rental system using the JUnit tool in Eclipse. The focus of the testing was on the AES class, which handles encryption and decryption operations.

Case 1: AES class



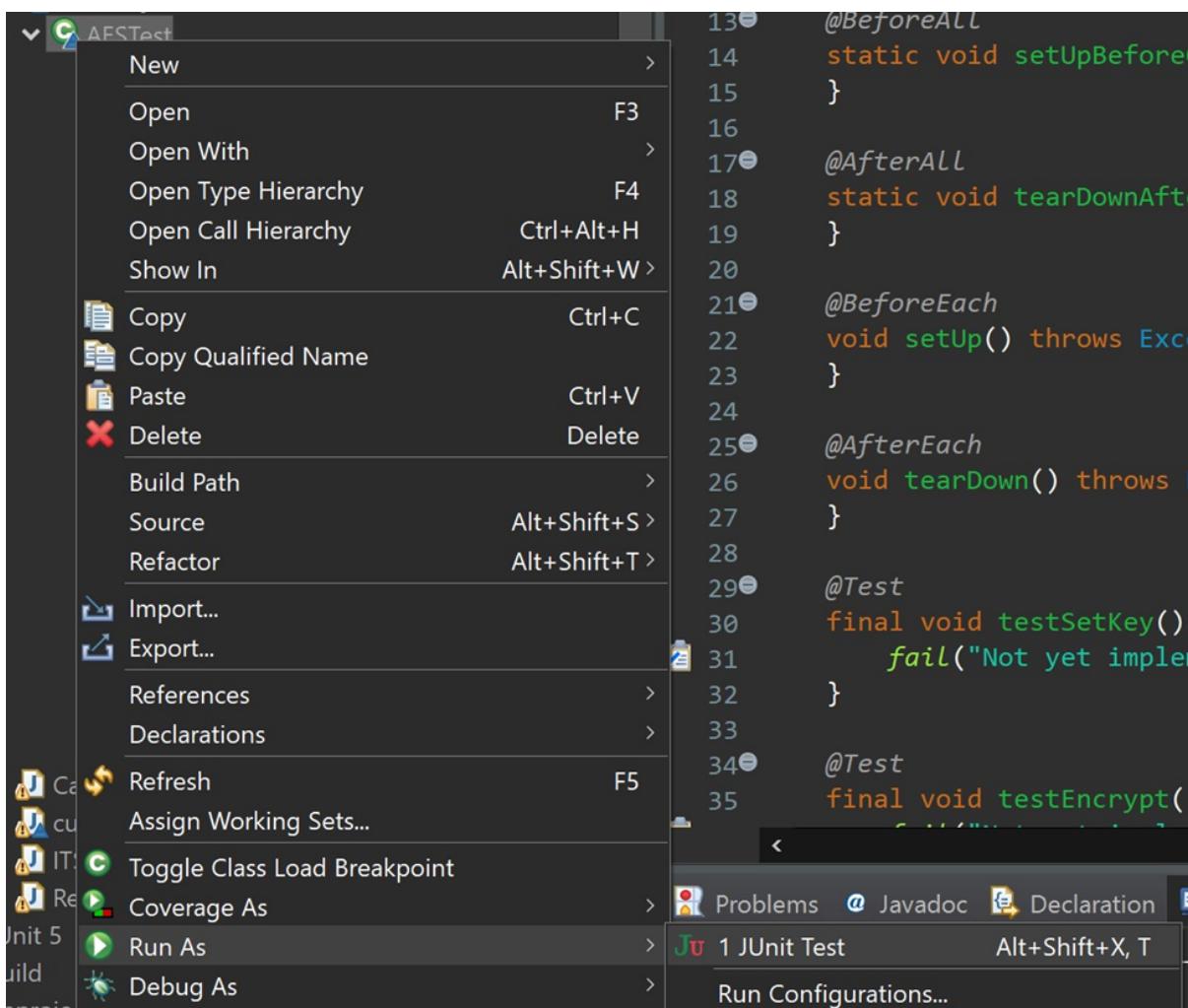


The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a project structure for a 'car' project. It includes a 'src' folder containing 'currentalproject' (which contains 'AES.java' and 'AESTest.java'), 'JUnit 5', 'build', 'nbproject', 'reports', 'test', and several text files like 'CarRented.txt', 'cars.txt', etc. On the right, the code editor shows the 'AESTest.java' file:

```

1 package currentalproject;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class AESTest {
6
7     @BeforeAll
8     static void setUpBeforeClass() throws Exception {
9     }
10
11     @AfterAll
12     static void tearDownAfterClass() throws Exception {
13     }
14
15     @BeforeEach
16     void setUp() throws Exception {
17     }
18
19     @AfterEach
20     void tearDown() throws Exception {
21     }
22
23     @Test
24     final void testSetKey() {
25         fail("Not yet implemented"); // TODO
26     }
27
28     @Test
29     final void testEncrypt() {
30
31
32
33
34
35

```



```

1 package carrentalproject;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6 class AESTest {
7
8     @BeforeAll
9     static void setUpBeforeClass() throws Exception {
10
11     }
12
13     @AfterAll
14     static void tearDownAfterClass() throws Exception {
15
16     }
17
18     @BeforeEach
19     void setUp() throws Exception {
20
21     }
22
23     @AfterEach
24     void tearDown() throws Exception {
25
26     }
27
28     @Test
29     final void testSetKey() {
30
31         fail("Not yet implemented"); // TODO
32
33     }
34
35     @Test
36     final void testEncrypt() {
37
38     }
39
40 }

```

Failure Trace

- org.opentest4j.AssertionFailedError: Not yet implemented
 - at carrentalproject.AESTest.testEncrypt(AESTest.java:36)
 - at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
 - at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

```

1 package carrentalproject;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6 class AESTest {
7
8     @BeforeAll
9     static void setUpBeforeClass() throws Exception {
10
11     }
12
13     @AfterAll
14     static void tearDownAfterClass() throws Exception {
15
16     }
17
18     @BeforeEach
19     void setUp() throws Exception {
20
21     }
22
23     @AfterEach
24     void tearDown() throws Exception {
25
26     }
27
28     @Test
29     final void testSetKey() {
30
31         fail("Not yet implemented"); // TODO
32
33     }
34
35     @Test
36     final void testEncrypt() {
37
38     }
39
40 }

```

Failure Trace

- org.opentest4j.AssertionFailedError: Not yet implemented
 - at carrentalproject.AESTest.testEncrypt(AESTest.java:36)
 - at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
 - at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

Findings

During dynamic testing of the car rental system, the AES class, responsible for encryption and decryption, exhibited failures. These failures indicate potential implementation issues, such as incorrect algorithms or inadequate input validation. Further investigation is required to identify and address the root causes of the failures, ensuring the AES class functions correctly and provides reliable encryption functionality.

Black Box Testing

Case 1:

```
***** Welcome to the car rental *****  
Please choose the the account you want to sign in..  
Enter [1] for customer  
Enter [2] for renter  
Enter [3] for IT support  
Enter Your Choice :
```

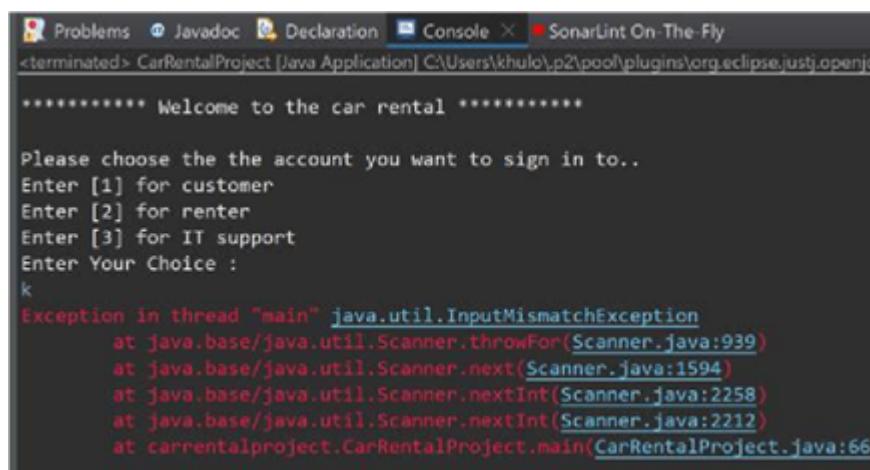
Entering a character as a choice

I choose this case because a character as input shouldn't be accepted as a choice for the menu displayed according to the specification

Expected Results:

It should display a message that states "you should only enter numbers"

Actual Results:



The screenshot shows the Eclipse IDE's Console view. The title bar indicates the project is 'CarRentalProject' and the type is 'Java Application'. The console output is as follows:

```
***** Welcome to the car rental *****  
Please choose the the account you want to sign in to..  
Enter [1] for customer  
Enter [2] for renter  
Enter [3] for IT support  
Enter Your Choice :  
K  
Exception in thread "main" java.util.InputMismatchException  
at java.base/java.util.Scanner.throwFor(Scanner.java:939)  
at java.base/java.util.Scanner.next(Scanner.java:1594)  
at java.base/java.util.Scanner.nextInt(Scanner.java:2258)  
at java.base/java.util.Scanner.nextInt(Scanner.java:2212)  
at carrentalproject.CarRentalProject.main(CarRentalProject.java:66)
```

The character input wasn't handled by the code programmers

Case 2:

```
Enter your name: Khulod  
Enter your age: 21  
Enter your ID num:
```

Entering a negative number as an input

I choose this case because a negative number as input shouldn't be accepted as an ID because there is no valid ID that has characters according to the specification

Expected Results:

It should display a message that states "you should only enter positive numbers"

Actual Results:

```
Enter your ID num: -10  
Enter your key:
```

It accepted the negative input

Conclusion

In summary, I performed static analysis using SpotBugs and identified a bug related to redundant encryption. For dynamic testing, I used JUnit to evaluate the system's behavior, uncovering failures in the AES class. Additionally, two black box test cases focused on input validation. These efforts aim to enhance the system's quality, security, and robustness.