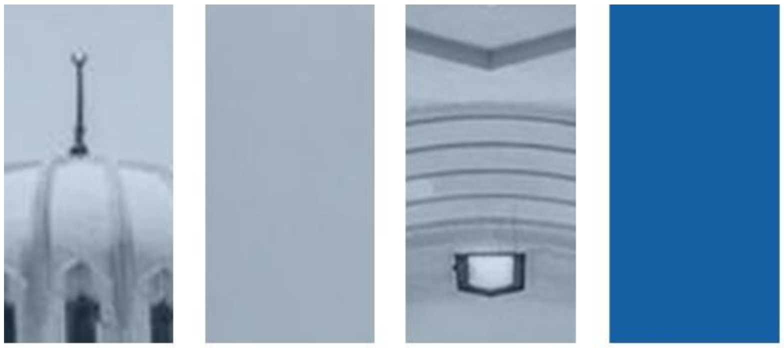


# CSE 338 Software Testing Validation and Verification Project Spring 2022

Examination committee

Dr. Islam El Maddah

Eng. Omar Ahmed

## *Team members*

|  |  |
|---|---|
| Nachaat Fahmy Nachaat | 19p2460 |
| Boules Emad Boules | 19p9291 |
| Shady Ossama Wadea | 19p2602 |
| Kirollos Georges Boutros | 19p9058 |
| Ahmed mekheimer | 1809799 |
| omar bahaa eldein ahmed | 18p7910 performance testing |

https://github.com/kiro1973?tab=repositories

## 1 Unit Testing

The application is build in object oriented methodology so that Programs that test classes are known as object-oriented unit tests. Each test case analyses a specific feature of the behavior of the class under test through a predetermined sequence of method invocations with fixed inputs. Unit tests are becoming an increasingly significant part of the software development process.

A unit in object-oriented programming is typically a whole interface, such as a class, although it can also be a single method.

Why is unit testing so difficult in OOP?

Why is unit testing in OOP more difficult than it is in functional programming? Internal state that is not easily accessed by tests may be maintained by objects. The quality of functional language unit testing frameworks has improved. Because OOP encourages code reuse, your tests will have to evaluate more use cases.

Unit testing entails putting your application's smaller components, such as classes and methods, to the test. You test your code to verify to yourself, and possibly to your users, clients, or customers, that it works.

## Testing the customer class

First of all we instantiated an object from the class Customer in our Test class

And gave values to its attributes

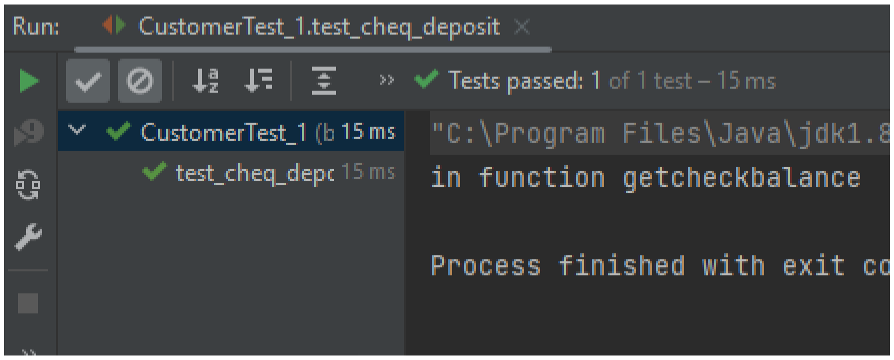
```
public class CustomerTest_1 extends TestCase {

    String customer_name="bolbol";
    String Password="bolbol";

    int customer_balance ;
    public   Customer customer= new Customer(customer_name,Password,100);
    Level customer_level = new Gold();
```

- first function that we used to test is cheqDeposit

```
public void cheqDeposit(int Amount) {
        //EFFECTS: Adds input to the checkBalance of Customer.
        //MODIFIES: checkBalance of Customer.
if (Amount>0)
            Customer.this.checkBalance += Amount;

                                            }
```

- its Test
    - ➢ successful test case

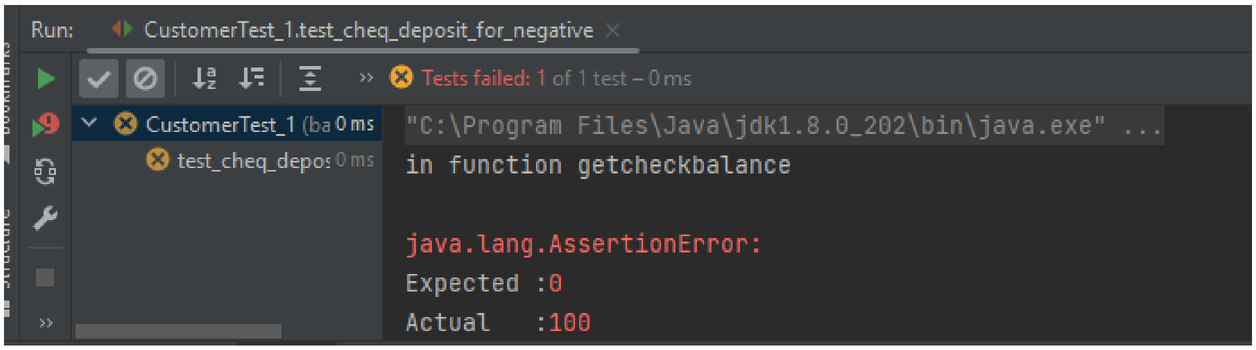
```
@Test
public void test_cheq_deposit()
{
    int amount=200;
    customer.cheqDeposit(100);
    Assert.assertEquals(amount,customer.getCheckBalance());
}
```


Run:
CustomerTest_1.test_cheq_deposit
Tests passed: 1 of 1 test – 15 ms
CustomerTest_1 (b 15 ms
test_cheq_depc 15 ms

```
"C:\Program Files\Java\jdk1.8
in function getcheckbalance

Process finished with exit co
```

- failed test case

```
public void test_cheq_deposit_for_negative ()
{
    //int amount=200;
    customer.cheqDeposit(-100);

    Assert.assertEquals(0,customer.getCheckBalance());
}
```



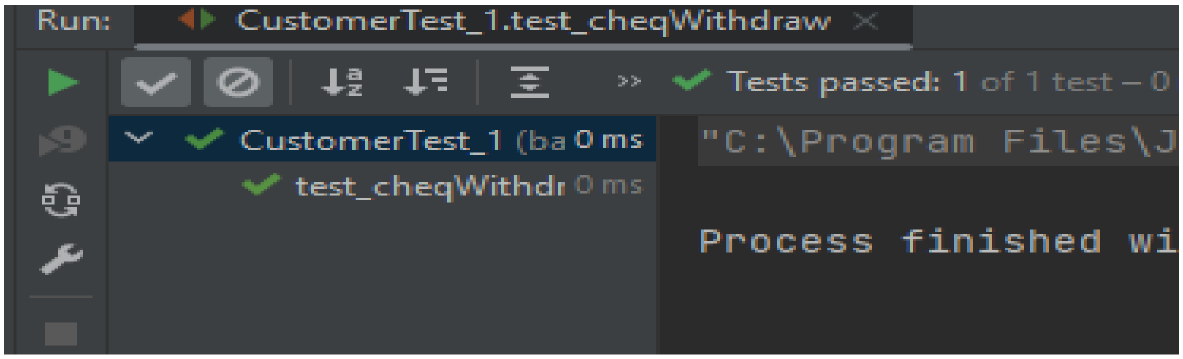
Tests failed: 1 of 1 test – 0 ms

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
in function getcheckbalance

java.lang.AssertionError:
Expected :0
Actual   :100
```

Tests for checking_withdraw

Tests that the inserted amount can be withdrawn from the balance , so a user cannot withdraw an amount greater than his balance

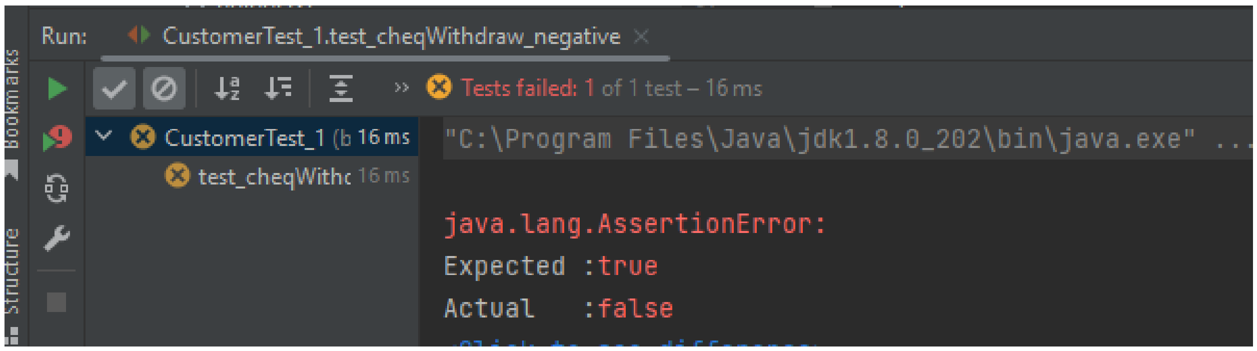
```
public void test_cheqWithdraw()
{
    int withraw_amount=50;
    Assert.assertTrue(customer.cheqWithdraw(withraw_amount));
}
```



```
Run:  CustomerTest_1.test_cheqWithdraw
Tests passed: 1 of 1 test – 0
CustomerTest_1 (ba 0 ms
    test_cheqWithdr 0 ms

"C:\Program Files\J

Process finished wi
```

- Failed test case

```
public void test_cheqWithdraw_negative()
{
    int withraw_amount=200;

Assert.assertEquals(true,customer.cheqWithdraw(withraw_amou
nt));
}
```



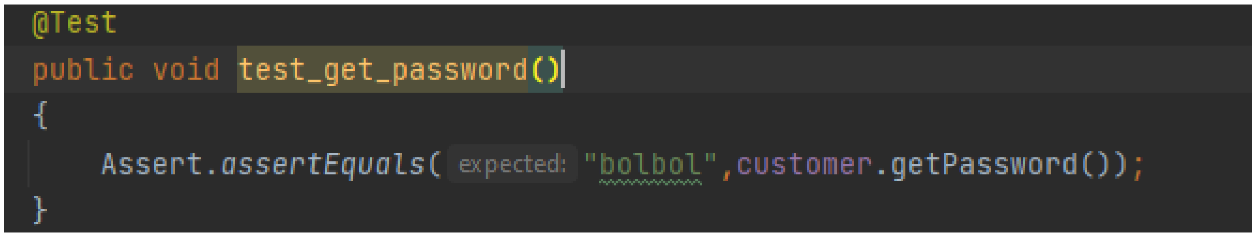
```
Run:  CustomerTest_1.test_cheqWithdraw_negative
Tests failed: 1 of 1 test – 16 ms
CustomerTest_1 (b 16 ms
    test_cheqWithc 16 ms

"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

java.lang.AssertionError:
Expected :true
Actual   :false
```

Testing the getter function for the user name

```
@Test
public void test_get_name()
{
    Assert.assertEquals("bolbol",customer.getName());
}
```

Tests the function against right input i.e. an already existing client in the system

Testing the getter function for the password



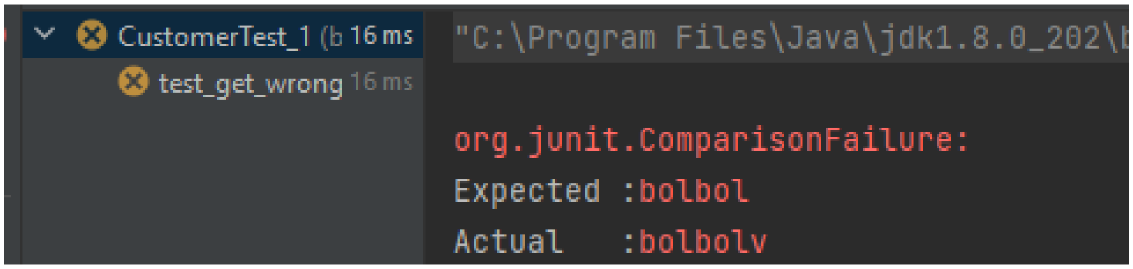
```
@Test
public void test_get_password()
{
    Assert.assertEquals( expected: "bolbol",customer.getPassword());
}
```

Tests the function against right input i.e. an already existing client in the system

```
@Test
public void test_get_wrong_password()
{
    Assert.assertEquals("bolbol",customer.getPassword());
}
```

.

Tests the function against wrong input



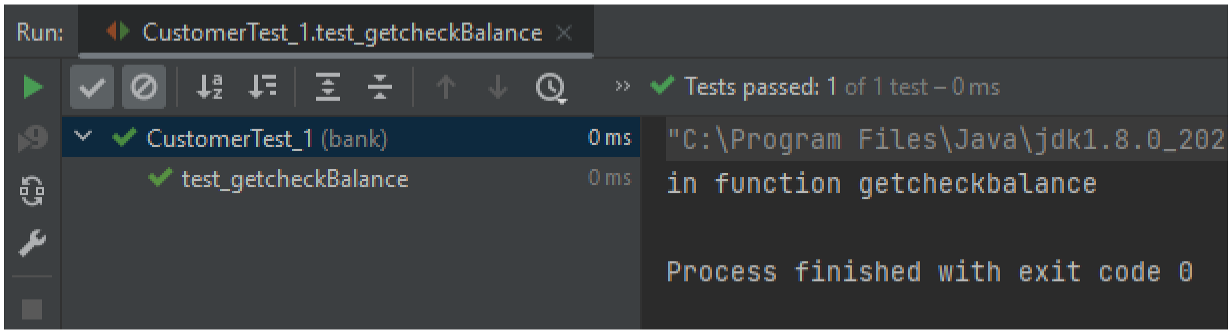
```
CustomerTest_1 (b 16 ms
    test_get_wrong 16 ms
```

```
"C:\Program Files\Java\jdk1.8.0_202\

org.junit.ComparisonFailure:
Expected :bolbol
Actual   :bolbolv
```

## Testing the function that gets the balance of the customer

```
public void test_getcheckBalance()
{
    Assert.assertEquals(100,customer.getCheckBalance());
}
```

- ➢ we already announced the balance of this customer to 100 so we could have a successful test



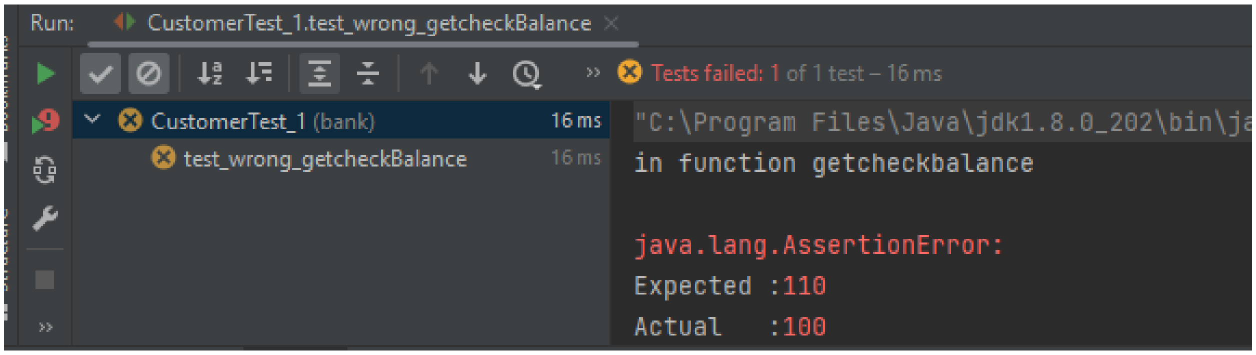
Run: CustomerTest_1.test_getcheckBalance

```
Tests passed: 1 of 1 test – 0 ms

CustomerTest_1 (bank)          0 ms
    test_getcheckBalance       0 ms

"C:\Program Files\Java\jdk1.8.0_202
in function getcheckbalance

Process finished with exit code 0
```

- ➢ failed test case

```
➢ public void test_wrong_getcheckBalance()
  {

  Assert.assertEquals(110,customer.getCheckBalance());
  }
```

here we made a failed test case because the balance should be 100 not 110

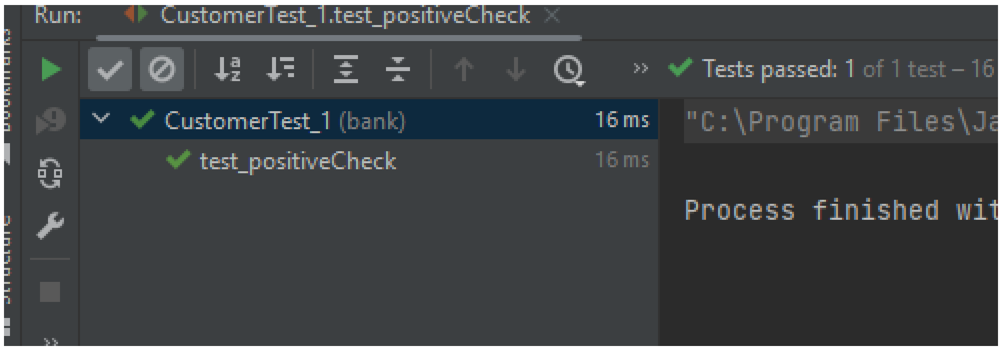


Run: CustomerTest_1.test_wrong_getcheckBalance

```
Tests failed: 1 of 1 test – 16 ms

CustomerTest_1 (bank)              16 ms
    test_wrong_getcheckBalance     16 ms

"C:\Program Files\Java\jdk1.8.0_202\bin\ja
in function getcheckbalance

java.lang.AssertionError:
Expected :110
Actual   :100
```

Testing the function that checks that the balance is positive

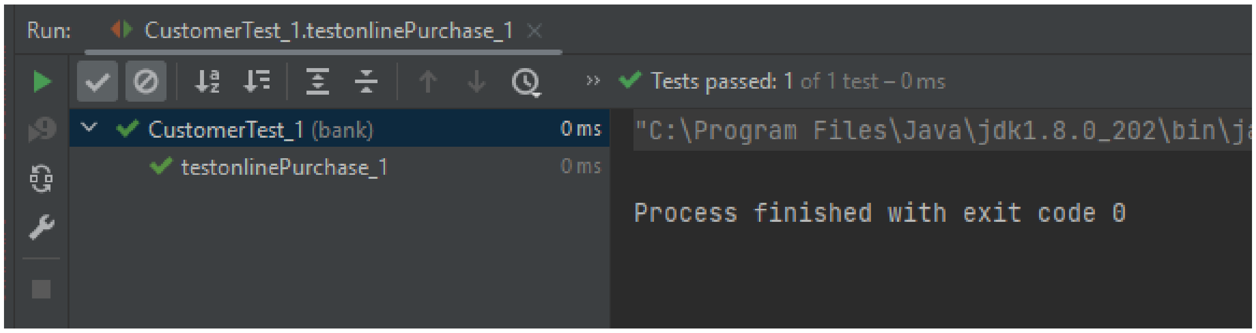
```
@Test
public void test_positiveCheck()
{
    Assert.assertTrue(customer.positiveCheck(0));
}
```



```
Run:   CustomerTest_1.test_positiveCheck
Tests passed: 1 of 1 test – 16
CustomerTest_1 (bank)    16 ms
  test_positiveCheck     16 ms
"C:\Program Files\Ja

Process finished wit
```

## Testing the function that performs the online purchase

```
public void testonlinePurchase_1()
{
    customer.setTheLevel(new Gold());

Assert.assertEquals(true,customer.onlinePurchase(60,customer));

}
```



Run: CustomerTest_1.testonlinePurchase_1

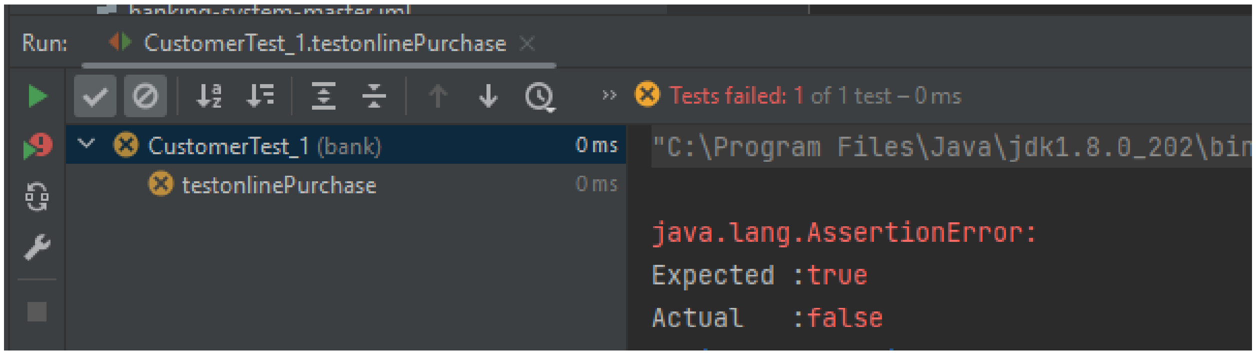
Tests passed: 1 of 1 test – 0 ms

CustomerTest_1 (bank) 0 ms
testonlinePurchase_1 0 ms

```
"C:\Program Files\Java\jdk1.8.0_202\bin\ja

Process finished with exit code 0
```

➢ Failed test case if he wants to make an online purchase if the amount that he wants to purchase with is less than 50

```
➢ @Test
  public void testonlinePurchase()
  {
      customer.setTheLevel(new Gold());

  Assert.assertFalse(customer.onlinePurchase(30,customer));

  }
```



Run: CustomerTest_1.testonlinePurchase

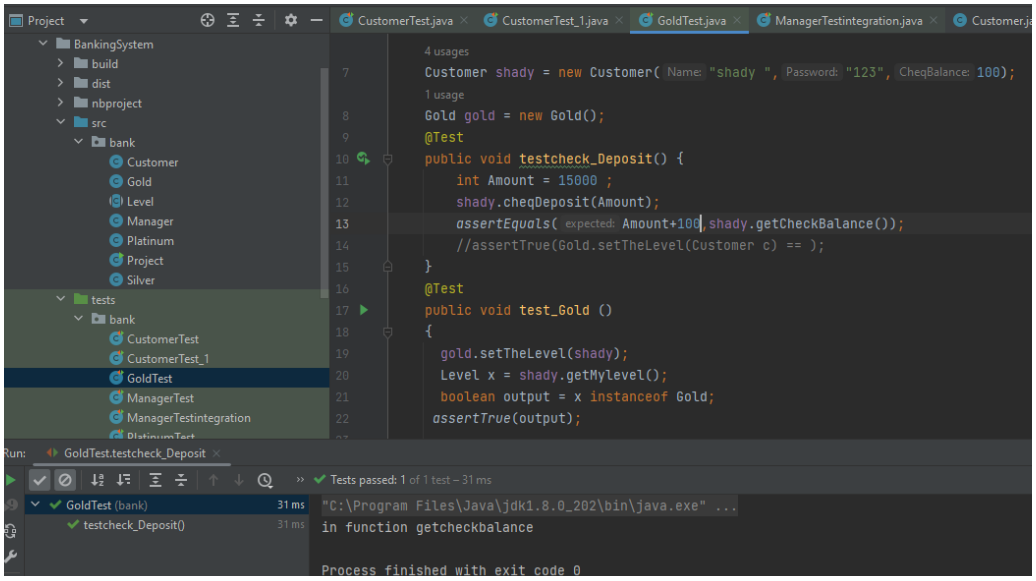
Tests failed: 1 of 1 test – 0 ms

CustomerTest_1 (bank) 0 ms
testonlinePurchase 0 ms

```
"C:\Program Files\Java\jdk1.8.0_202\bin

java.lang.AssertionError:
Expected :true
Actual   :false
```

**This is a function that test to check the amount of money after this deposit is equal to the balance after the deposite**

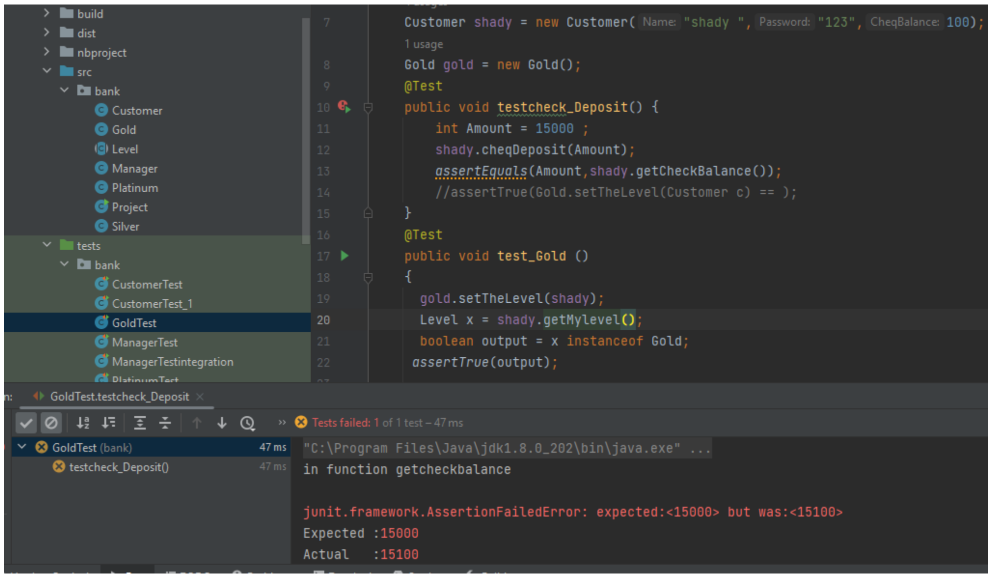
Which is in this example is equal to the sumof amount entered and the old balance



```java
Customer shady = new Customer("shady ", "123", 100);
Gold gold = new Gold();
@Test
public void testcheck_Deposit() {
    int Amount = 15000 ;
    shady.cheqDeposit(Amount);
    assertEquals( Amount+100,shady.getCheckBalance());
    //assertTrue(Gold.setTheLevel(Customer c) == );
}
@Test
public void test_Gold ()
{
  gold.setTheLevel(shady);
  Level x = shady.getMylevel();
  boolean output = x instanceof Gold;
 assertTrue(output);
```

```
Tests passed: 1 of 1 test – 31 ms
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
in function getcheckbalance

Process finished with exit code 0
```

this is the same test but with a uncorrect expected output



```
Customer shady = new Customer( Name: "shady ", Password: "123", CheqBalance: 100);
1 usage
Gold gold = new Gold();
@Test
public void testcheck_Deposit() {
    int Amount = 15000 ;
    shady.cheqDeposit(Amount);
    assertEquals(Amount,shady.getCheckBalance());
    //assertTrue(Gold.setTheLevel(Customer c) == );
}
@Test
public void test_Gold ()
{
  gold.setTheLevel(shady);
  Level x = shady.getMylevel();
  boolean output = x instanceof Gold;
 assertTrue(output);
```

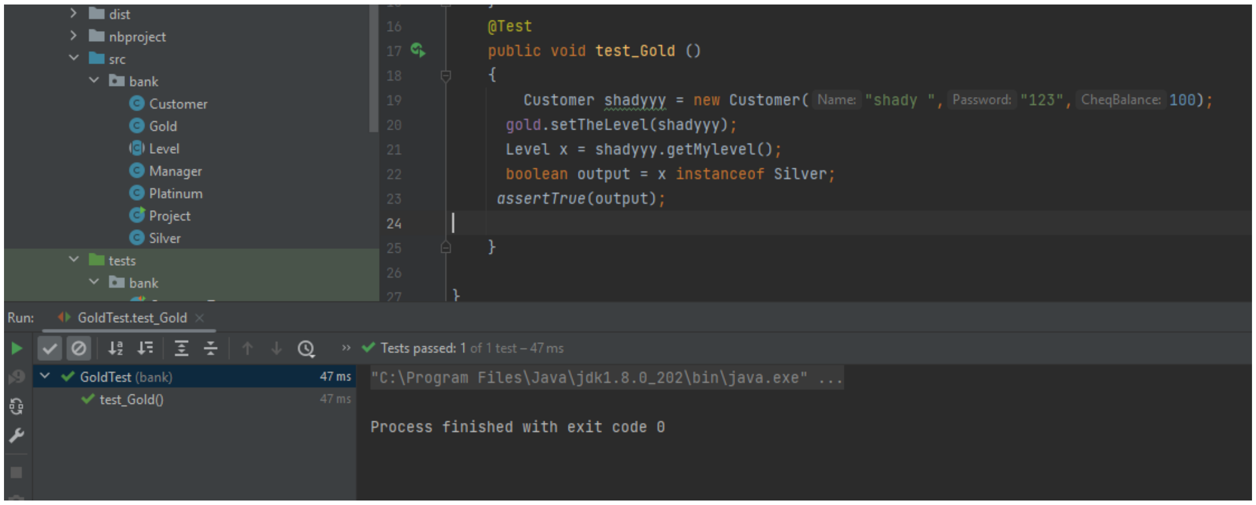
GoldTest.testcheck_Deposit

Tests failed: 1 of 1 test – 47 ms

GoldTest (bank) 47 ms
testcheck_Deposit() 47 ms

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
in function getcheckbalance

junit.framework.AssertionFailedError: expected:<15000> but was:<15100>
Expected :15000
Actual   :15100
```

This is a function to test the setthelevel of gold which use another functions like get my level in customer



```
@Test
public void test_Gold ()
{
    Customer shadyyy = new Customer( Name: "shady ", Password: "123", CheqBalance: 100);
  gold.setTheLevel(shadyyy);
  Level x = shadyyy.getMylevel();
  boolean output = x instanceof Silver;
 assertTrue(output);

  }

}
```

Run: GoldTest.test_Gold

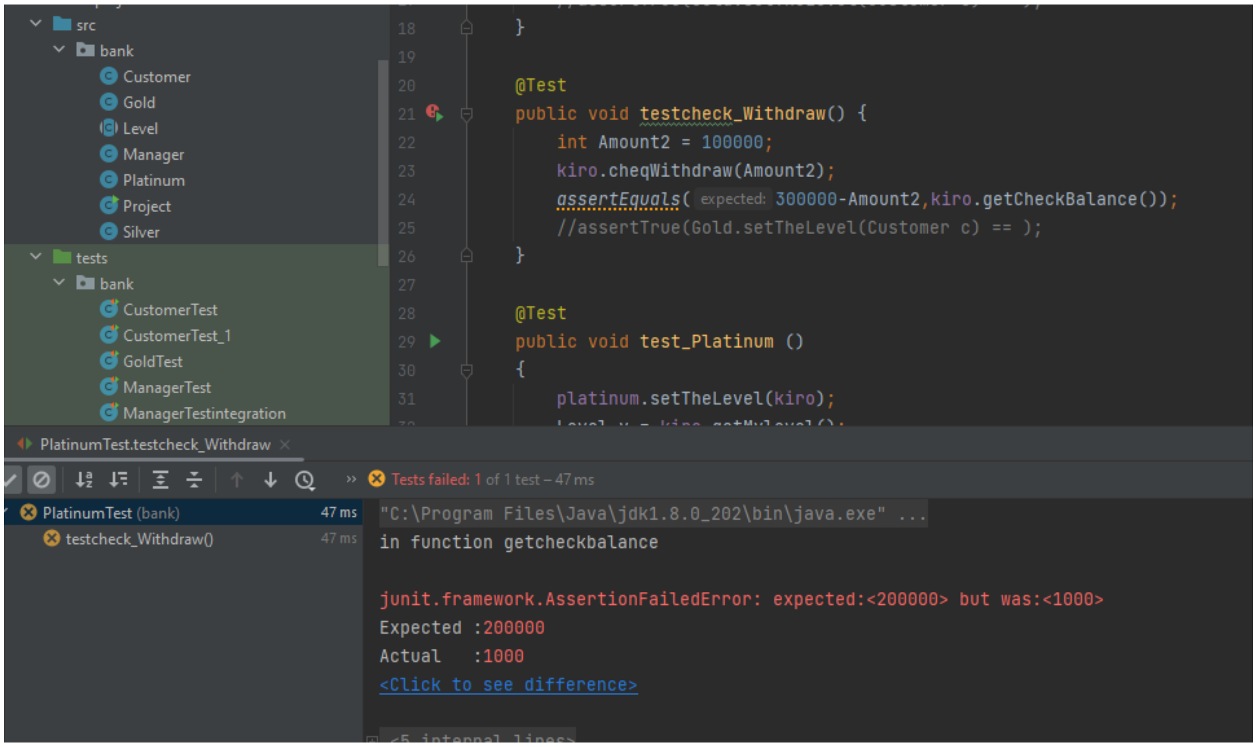
Tests passed: 1 of 1 test – 47 ms

GoldTest (bank) 47 ms
test_Gold() 47 ms

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

Process finished with exit code 0
```

## this is a function to test to check the withdraw action from a customer

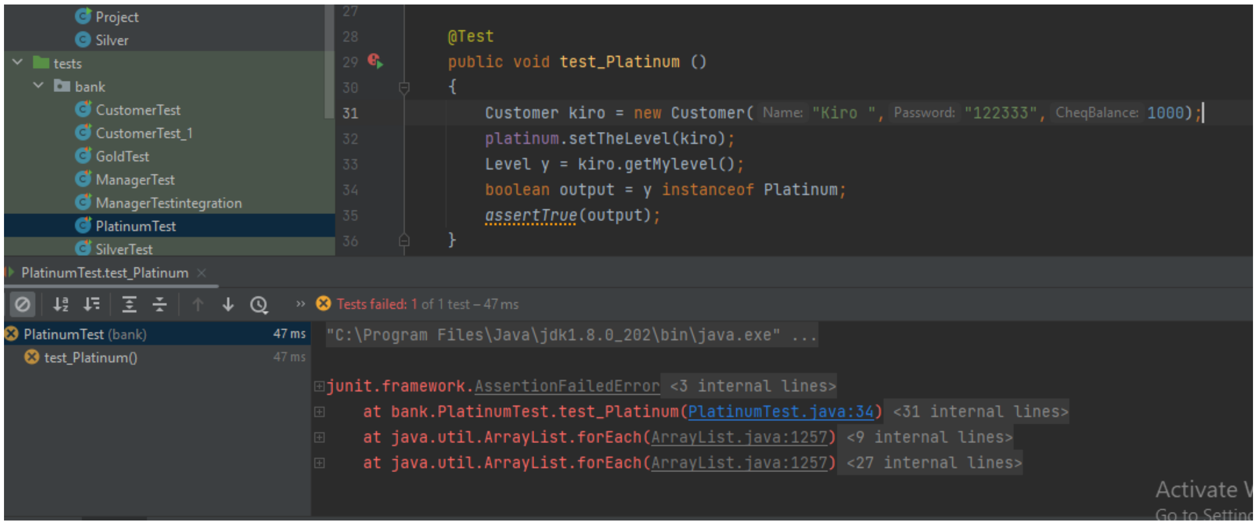
in this example the user want to withdraw an amount which is invalid to withdraw which makes sense that the test case is false



```
    }

    @Test
    public void testcheck_Withdraw() {
        int Amount2 = 100000;
        kiro.cheqWithdraw(Amount2);
        assertEquals( expected: 300000-Amount2,kiro.getCheckBalance());
        //assertTrue(Gold.setTheLevel(Customer c) == );
    }

    @Test
    public void test_Platinum ()
    {
        platinum.setTheLevel(kiro);
```

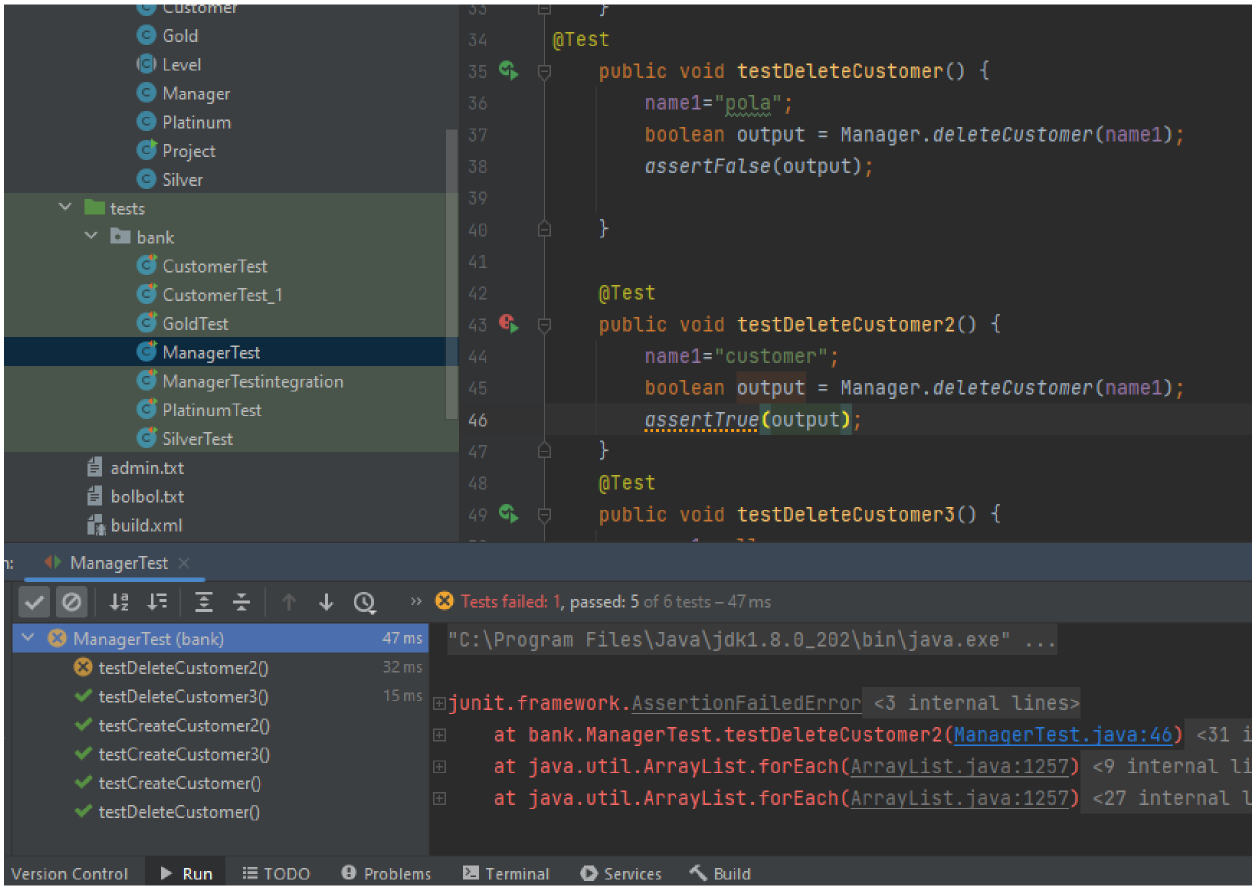
```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...
in function getcheckbalance

junit.framework.AssertionFailedError: expected:<200000> but was:<1000>
Expected :200000
Actual   :1000
<Click to see difference>

<5 internal lines>
```

**this is a function like the gold to set the level and in this case the user balance is lower the platinuim category so the test case will fail which is true**



```
@Test
public void test_Platinum ()
{
    Customer kiro = new Customer( Name: "Kiro ", Password: "122333", CheqBalance: 1000);
    platinum.setTheLevel(kiro);
    Level y = kiro.getMylevel();
    boolean output = y instanceof Platinum;
    assertTrue(output);
}
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

junit.framework.AssertionFailedError <3 internal lines>
    at bank.PlatinumTest.test_Platinum(PlatinumTest.java:34) <31 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <9 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <27 internal lines>
```



```
    }
    @Test
        public void testDeleteCustomer() {
            name1="pola";
            boolean output = Manager.deleteCustomer(name1);
            assertFalse(output);

        }

        @Test
        public void testDeleteCustomer2() {
            name1="customer";
            boolean output = Manager.deleteCustomer(name1);
            assertTrue(output);
        }
        @Test
        public void testDeleteCustomer3() {
```

```
Tests failed: 1, passed: 5 of 6 tests – 47 ms

ManagerTest (bank)          47 ms
    testDeleteCustomer2()   32 ms
    testDeleteCustomer3()   15 ms
    testCreateCustomer2()
    testCreateCustomer3()
    testCreateCustomer()
    testDeleteCustomer()

"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

junit.framework.AssertionFailedError <3 internal lines>
    at bank.ManagerTest.testDeleteCustomer2(ManagerTest.java:46) <31 i
    at java.util.ArrayList.forEach(ArrayList.java:1257) <9 internal li
    at java.util.ArrayList.forEach(ArrayList.java:1257) <27 internal l
```

**There we test the manager functions create and delete customer so we have made multiple test case**

But the second test case of testdeletecustomer2 wil be failed assert true

## Integration Testing

The ManagerTestIntegration class has all the integration testing functions , 2 Integration Testing were done in this class .
One using Top down approach with stubs , and the other using bottom up approach using drivers.

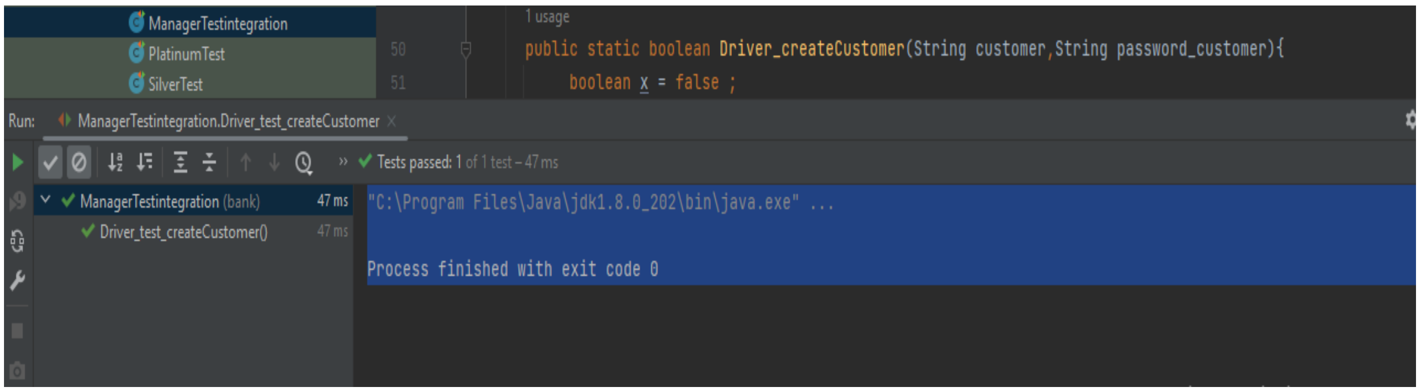
We made a static inner class named ("integ") in the testing class .That inner class contains the Stubs and the Drivers fuctions .

```
public static class integ {

  public  static ArrayList<String> arr = new ArrayList<>();

   public static boolean createcustomer(String nam,String pass) {
       if ((nam==null)||(pass==null))
               return false;
       else if(arr.contains(nam+" "+pass)){
           return false;
       }
       else {
           arr.add(nam + " " + pass);
           return true;
       }
   }

   public static boolean del_customer(String namee){
       if (namee==null)
               return false;
       else if  (arr.contains(namee)) {
           arr.remove(namee);
           return true;
       }
       return false;
   }


  public static boolean Driver_createCustomer(String customer,String
password_customer){
       boolean x = false ;
       if (customer!=null && password_customer!=null)
          x =  Manager.createCustomer(customer,password_customer);
       return  x;
   }

   public static boolean Driver_deleteCustomer(String customer){
       boolean y ;
      y= Manager.deleteCustomer(customer);
       return y ;
   }

}
```

The Driver method receives the returned value from the function that is called inside of the driver method and return it in order to test that value by the Junit .

```
@Test
    public void Driver_test_createCustomer(){

        assertTrue(integ.Driver_createCustomer(customer_name,password));

}
```

And by entering the right input to the function called inside the driver function,the testing is passed .


ManagerTestintegration
PlatinumTest
SilverTest
Run:
ManagerTestintegration.Driver_test_createCustomer
ManagerTestintegration (bank) 47 ms
Driver_test_createCustomer() 47 ms

```
1 usage
public static boolean Driver_createCustomer(String customer,String password_customer){
    boolean x = false ;
```

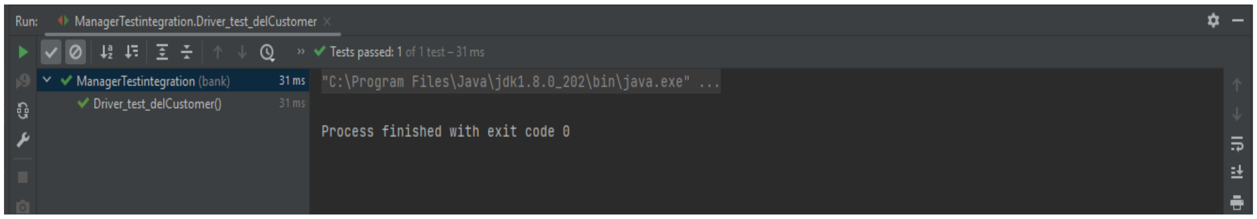
```
✓ Tests passed: 1 of 1 test – 47 ms
```

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

Process finished with exit code 0
```

And that was the first function that creates a customer inside the class ("Manager") to be tested as a Driver .By testing the second function inside that class that deletes the customer .

```
@Test
public void Driver_test_delCustomer(){
        assertTrue(integ.Driver_deleteCustomer(customer_name));
}

}
```

And the test of that function is passed as well as we used the customer_name previously entered for customers .

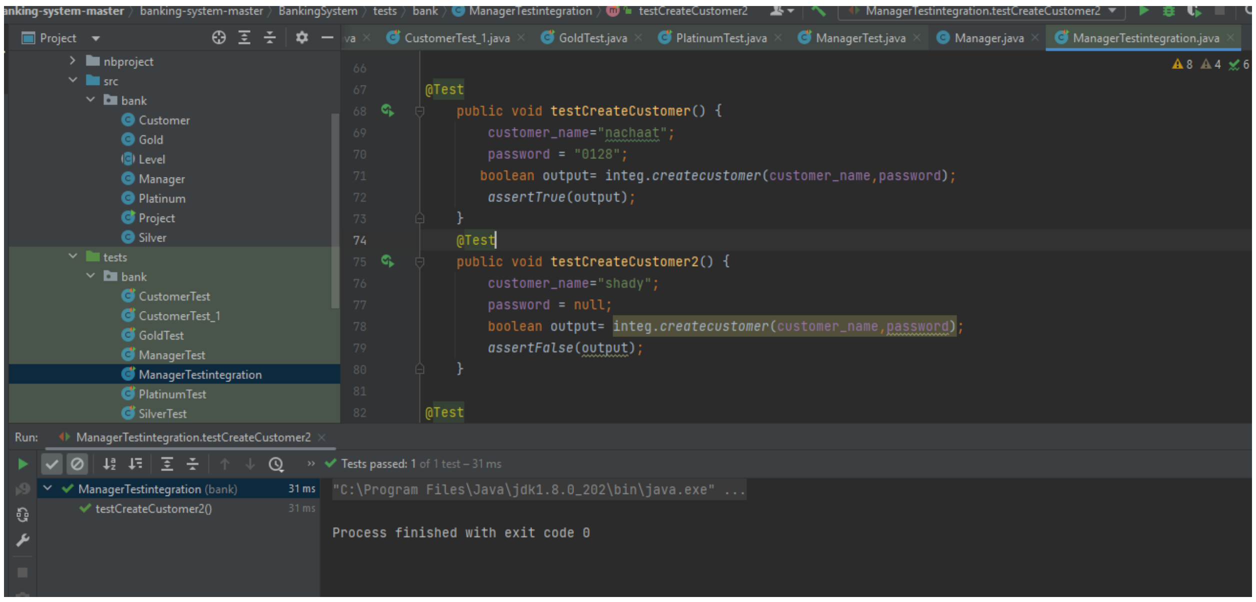


```
Run:   ManagerTestintegration.Driver_test_delCustomer ×
✓ Tests passed: 1 of 1 test – 31 ms
✓ ManagerTestintegration (bank)   31 ms
    ✓ Driver_test_delCustomer()   31 ms

"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

Process finished with exit code 0
```

Moreover , we tested the stubs created for the functions inside the class ("Manager") .In the case of the stubs we created two functions that mimic the authentic functions in the class Manager .By that method if you entered your inputs ,the right output will be returned from those functions .

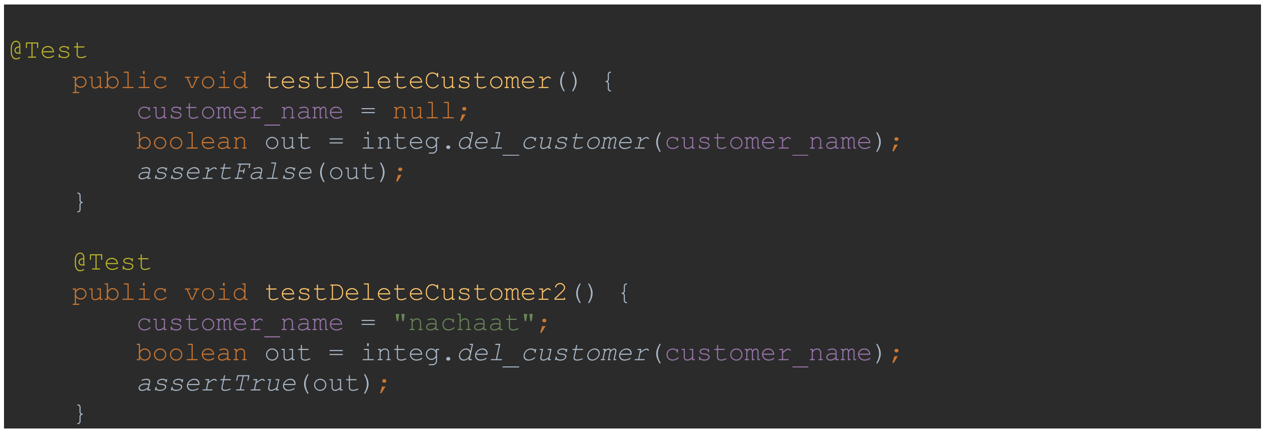
```
@Test
    public void testCreateCustomer() {
        customer_name="nachaat";
        password = "0128";
       boolean output= integ.createcustomer(customer_name,password);
        assertTrue(output);
    }
    @Test
    public void testCreateCustomer2() {
        customer_name="shady";
        password = null;
        boolean output= integ.createcustomer(customer_name,password);
        assertFalse(output);
    }
```

The first test used for the creation stub function was used Assert True . The second test used assert false and both was passed.



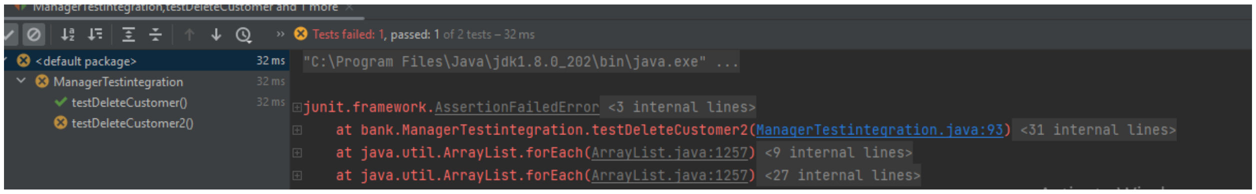
```java

@Test
    public void testCreateCustomer() {
        customer_name="nachaat";
        password = "0128";
       boolean output= integ.createcustomer(customer_name,password);
        assertTrue(output);
    }
    @Test
    public void testCreateCustomer2() {
        customer_name="shady";
        password = null;
        boolean output= integ.createcustomer(customer_name,password);
        assertFalse(output);
    }

@Test
```

```
Tests passed: 1 of 1 test – 31 ms
ManagerTestintegration (bank)   31 ms
  testCreateCustomer2()         31 ms

"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

Process finished with exit code 0
```



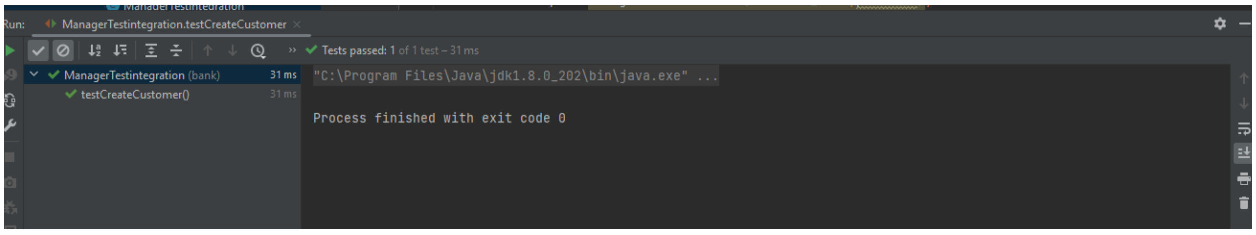
```java
@Test
    public void testDeleteCustomer() {
        customer_name = null;
        boolean out = integ.del_customer(customer_name);
        assertFalse(out);
    }

    @Test
    public void testDeleteCustomer2() {
        customer_name = "nachaat";
        boolean out = integ.del_customer(customer_name);
        assertTrue(out);
    }
```

The first test failed as you shouldn't delete a null value



```
Tests failed: 1, passed: 1 of 2 tests – 32 ms
<default package>             32 ms
  ManagerTestintegration      32 ms
    testDeleteCustomer()      32 ms
    testDeleteCustomer2()

"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

junit.framework.AssertionFailedError <3 internal lines>
    at bank.ManagerTestintegration.testDeleteCustomer2(ManagerTestintegration.java:93) <31 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <9 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <27 internal lines>
```

In case the name is (nachaat1) ,then test passes because it was previously inserted .


Run: ManagerTestintegration.testCreateCustomer
Tests passed: 1 of 1 test – 31 ms
ManagerTestintegration (bank) 31 ms
testCreateCustomer() 31 ms

```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

Process finished with exit code 0
```

## GUI Testing

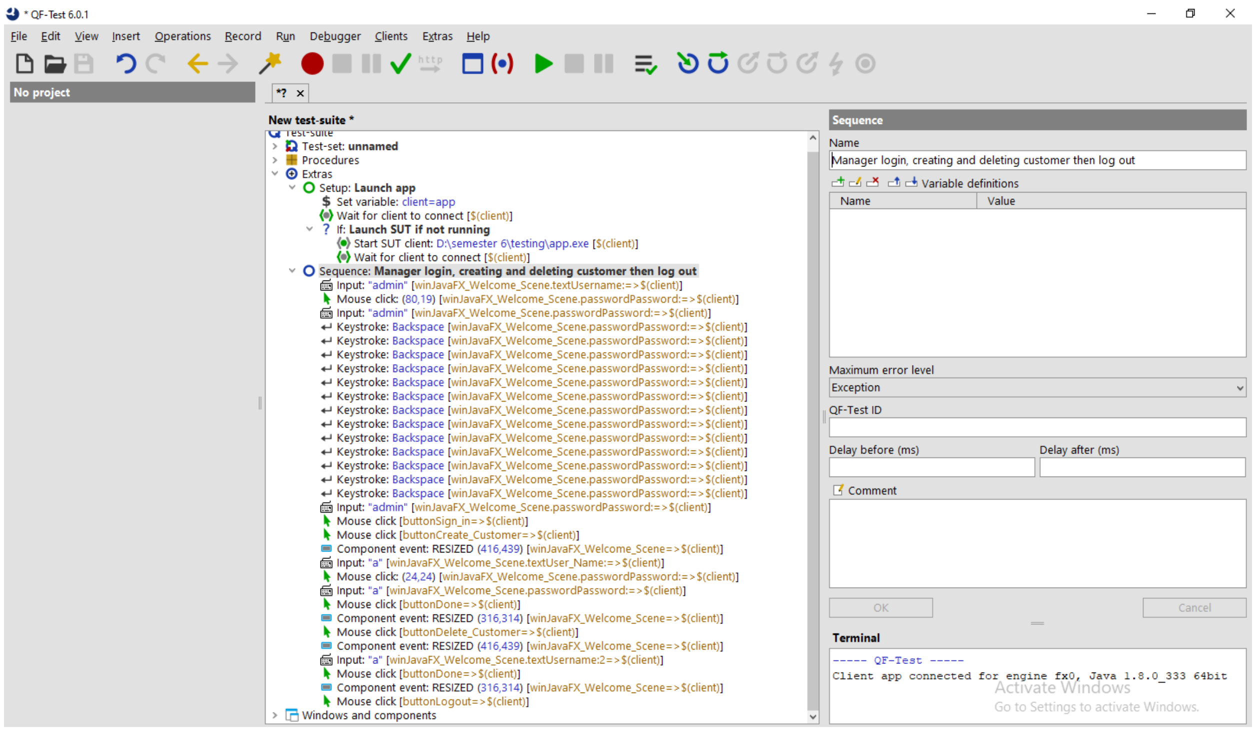
There is a tool used for GUI testing called QF , A German software used for testing projects with java GUI, something anyone must know about JAVAFX is JavaFX is the successor of Swing and an integral part of Java 8. With Java 11 JavaFX was moved out of the JDK into the open source project OpenJFX. Since 2014 QF-Test supports this GUI technology and provides advanced testing support including complex components like trees and tables. QF-Test can test JavaFX applications with an embedded browser like e.g. JxBrowser or WebView. You can access both, the Java and Web part of the application.
When you migrate your application from Java Swing to JavaFX, you can reuse Swing GUI tests in QF-Test for JavaFX automated testing with little effort.

As of March 2014, JavaFX has been a permanent part of JDK 8. In future, JavaFX 8 will be used to build mission critical business applications. On the one hand, this requires a degree of maturity of new JavaFX UI technology to be acceptable. On the other hand, it is necessary that user interfaces of business applications could be tested extensively for their correct application functionality. The focus is on the high data volume that is fed to automated tests into the graphical user interface. Please see below for an illustration of how such a test scenario could be set up with the QF-Test tool. During applications development, testing plays an important role. According to the current World Quality Report, a quarter of the IT budget is already allotted for testing. The increasing complexity of the application development also need the testing requirements to increase. High test coverage can be achieved by test automation. Suitable tools help facilitate the setting up of automated tests and ensure a high integration with the developed application. The benefit of automated UI tests is that they can cover a whole process with one test in contrast to unit tests which cover only an isolated unit. By using the correct test environment it is possible to have the bulk of tests set up by professional testers, thus freeing resources for development work. For this, the test environment should allow the tester to work with familiar terms and objects. Especially in GUI tests it is important that testers work with objects they know and recognize on the GUI of the application, even though the actual structure of the GUI is much more complex. Even in simple Java FX applications the GUI consists of many single elements in a highly complex tree structure. Figure 1 shows a demo application for the configuration of new cars. The GUI structure of this application becomes visible in a 3D ,three-dimensionally highlighting the complexity of the nesting. However, only a small part of the elements is relevant for testing, all others are integrated on a technical level. The test tool allows the reduction of these complex GUIs to the essential. Figure 3 shows the simplified structure as it is made available to the tester by the test tool. In specific cases it is possible to work with the full hierarchy at any time, and programmable tests are possible via scripting. Through simplification and generalization of the UI components, the support of a modular structure allows parts of the test to be Figure 2: Demo application “configuration of new cars” (3dimensional). Thus the effort required for test creation is reduced and at the same time maintainability increased. It provides the opportunity to create modules and libraries professional testers can work with. Due to the additional integration of software drivers, data-driven testing is also possible on the GUI, and mass testing can be created. Conclusion GUI testing of an application is essential to the corresponding development. By directly integrating a test tool into the development process, a continuous integration scenario could be set up. Test tools are facilitating the JavaFX 8 applications testing by simplifying the component hierarchy and due to its modular structure. Also this is a valid option for testers, who haven't any programming knowledge.

First we transform the project into an executable file, then we open it using the QF test tool which is very easy.

We record a sequence with events ,

So the first sequence is the following for the manager


* QF-Test 6.0.1
File Edit View Insert Operations Record Run Debugger Clients Extras Help
No project
*?
New test-suite *
Sequence
OK
Cancel
Activate Windows
Go to Settings to activate Windows.

```
Test-suite
> Test-set: unnamed
> Procedures
v Extras
  v Setup: Launch app
      Set variable: client=app
      Wait for client to connect [$(client)]
    v If: Launch SUT if not running
        Start SUT client: D:\semester 6\testing\app.exe [$(client)]
        Wait for client to connect [$(client)]
  v Sequence: Manager login, creating and deleting customer then log out
      Input: "admin" [winJavaFX_Welcome_Scene.textUsername:=>$(client)]
      Mouse click: (80,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Input: "admin" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Keystroke: Backspace [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Input: "admin" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Mouse click [buttonSign_in=>$(client)]
      Mouse click [buttonCreate_Customer=>$(client)]
      Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
      Input: "a" [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
      Mouse click: (24,24) [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Input: "a" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Mouse click [buttonDone=>$(client)]
      Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
      Mouse click [buttonDelete_Customer=>$(client)]
      Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
      Input: "a" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
      Mouse click [buttonDone=>$(client)]
      Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
      Mouse click [buttonLogout=>$(client)]
> Windows and components
```

Name: Manager login, creating and deleting customer then log out

Variable definitions

| Name | Value |
|---|---|
| | |

Maximum error level: Exception

QF-Test ID:

Delay before (ms):

Delay after (ms):

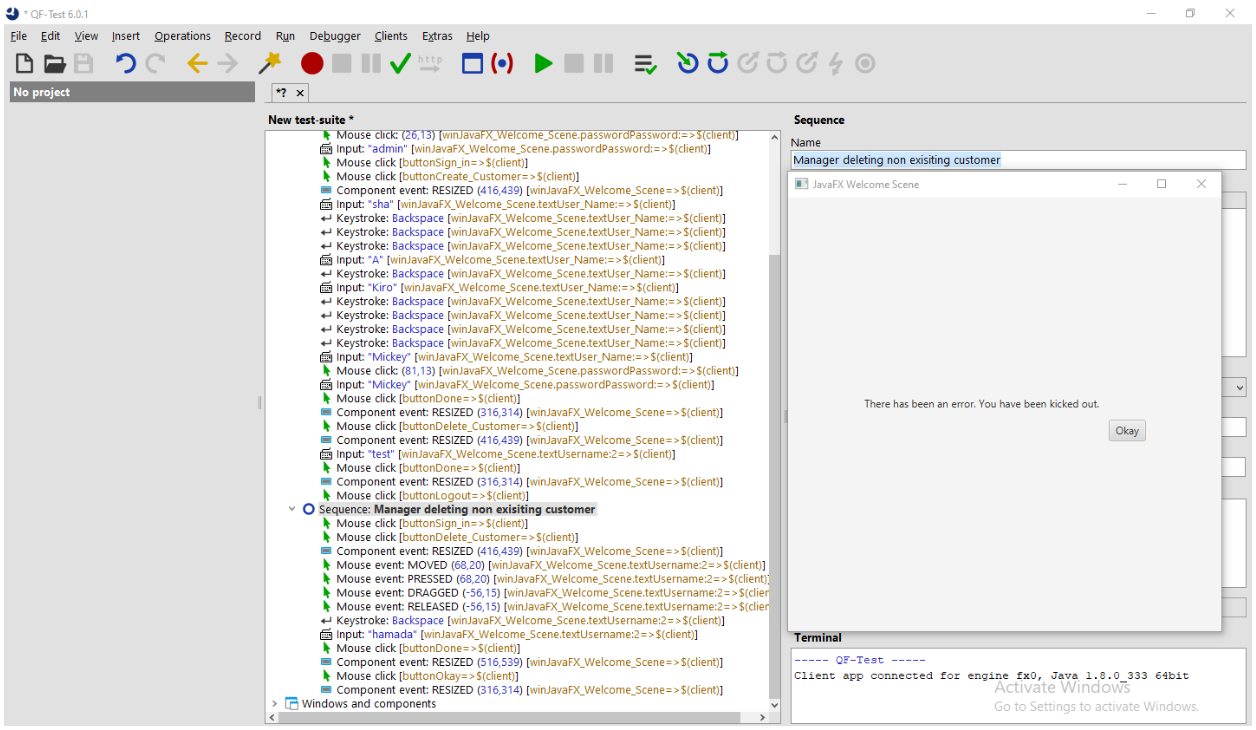
Comment:

Terminal

```
----- QF-Test -----
Client app connected for engine fx0, Java 1.8.0_333 64bit
```

Manager login, creating and deleting customer then log out and we found no errors in this sequence .

Then we will try to delete a customer that does not exist



New test-suite *

```
Mouse click: (26,13) [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
Input: "admin" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
Mouse click [buttonSign_in=>$(client)]
Mouse click [buttonCreate_Customer=>$(client)]
Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
Input: "sha" [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Input: "A" [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Input: "Kiro" [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Input: "Mickey" [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
Mouse click: (81,13) [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
Input: "Mickey" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
Mouse click [buttonDone=>$(client)]
Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
Mouse click [buttonDelete_Customer=>$(client)]
Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
Input: "test" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
Mouse click [buttonDone=>$(client)]
Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
Mouse click [buttonLogout=>$(client)]
Sequence: Manager deleting non exisiting customer
    Mouse click [buttonSign_in=>$(client)]
    Mouse click [buttonDelete_Customer=>$(client)]
    Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
    Mouse event: MOVED (68,20) [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Mouse event: PRESSED (68,20) [winJavaFX_Welcome_Scene.textUsername:2=>$(client)
    Mouse event: DRAGGED (-56,15) [winJavaFX_Welcome_Scene.textUsername:2=>$(clien
    Mouse event: RELEASED (-56,15) [winJavaFX_Welcome_Scene.textUsername:2=>$(clien
    Keystroke: Backspace [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Input: "hamada" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Mouse click [buttonDone=>$(client)]
    Component event: RESIZED (516,539) [winJavaFX_Welcome_Scene=>$(client)]
    Mouse click [buttonOkay=>$(client)]
    Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
Windows and components
```

Sequence

Name: Manager deleting non exisiting customer

JavaFX Welcome Scene

There has been an error. You have been kicked out.

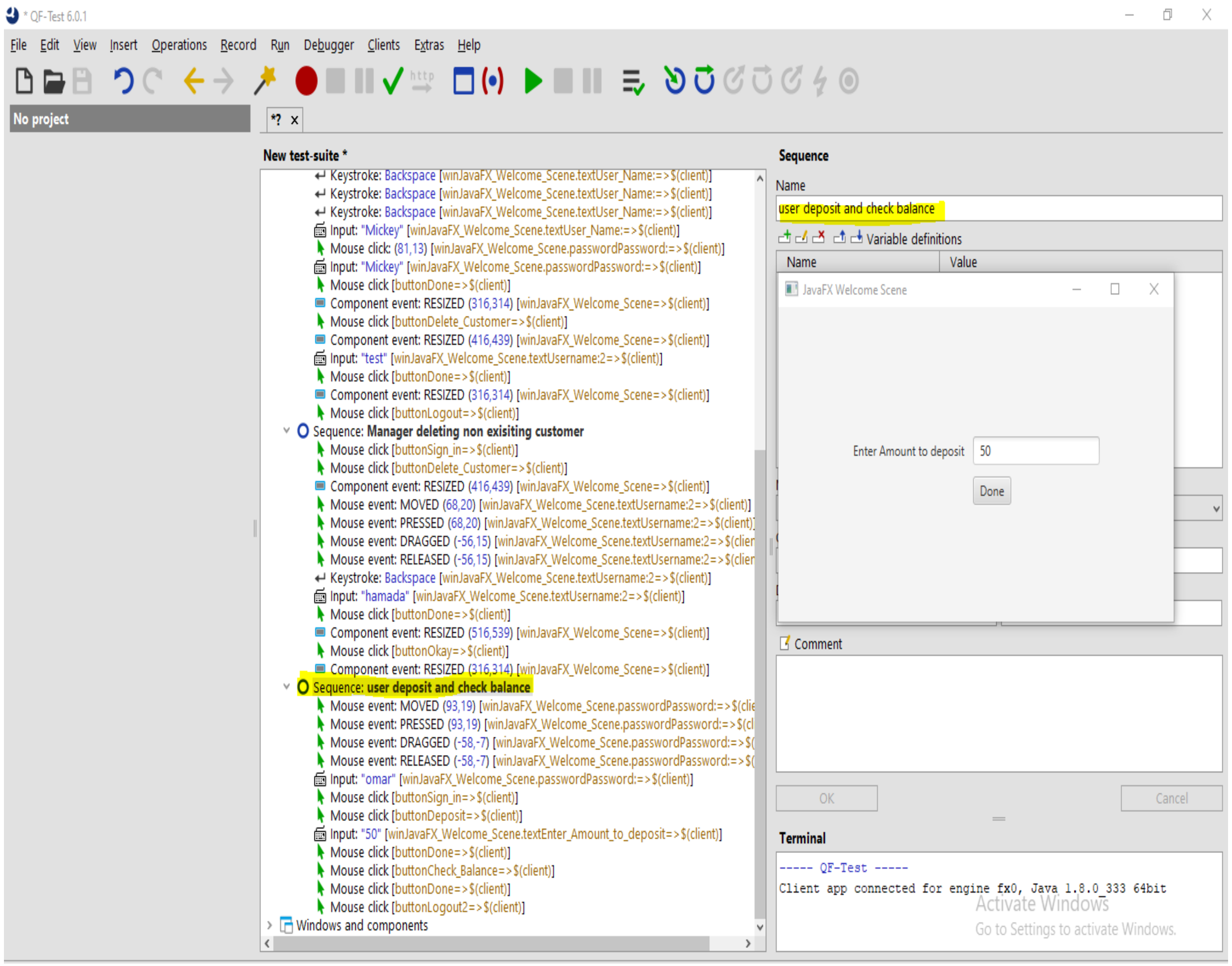
Okay

Terminal

```
----- QF-Test -----
Client app connected for engine fx0, Java 1.8.0_333 64bit
```

Here we find that the system interacts and show you an error message without any errors in the terminal , so this sequence also is free of bugs

We try to login as customer and add amount and check the total after that



```
New test-suite *
    Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
    Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
    Keystroke: Backspace [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
    Input: "Mickey" [winJavaFX_Welcome_Scene.textUser_Name:=>$(client)]
    Mouse click: (81,13) [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
    Input: "Mickey" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
    Mouse click [buttonDone=>$(client)]
    Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
    Mouse click [buttonDelete_Customer=>$(client)]
    Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
    Input: "test" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Mouse click [buttonDone=>$(client)]
    Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
    Mouse click [buttonLogout=>$(client)]
  Sequence: Manager deleting non exisiting customer
    Mouse click [buttonSign_in=>$(client)]
    Mouse click [buttonDelete_Customer=>$(client)]
    Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
    Mouse event: MOVED (68,20) [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Mouse event: PRESSED (68,20) [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Mouse event: DRAGGED (-56,15) [winJavaFX_Welcome_Scene.textUsername:2=>$(clien
    Mouse event: RELEASED (-56,15) [winJavaFX_Welcome_Scene.textUsername:2=>$(clien
    Keystroke: Backspace [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Input: "hamada" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
    Mouse click [buttonDone=>$(client)]
    Component event: RESIZED (516,539) [winJavaFX_Welcome_Scene=>$(client)]
    Mouse click [buttonOkay=>$(client)]
    Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
  Sequence: user deposit and check balance
    Mouse event: MOVED (93,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(clie
    Mouse event: PRESSED (93,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(cl
    Mouse event: DRAGGED (-58,-7) [winJavaFX_Welcome_Scene.passwordPassword:=>$(
    Mouse event: RELEASED (-58,-7) [winJavaFX_Welcome_Scene.passwordPassword:=>$(
    Input: "omar" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
    Mouse click [buttonSign_in=>$(client)]
    Mouse click [buttonDeposit=>$(client)]
    Input: "50" [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(client)]
    Mouse click [buttonDone=>$(client)]
    Mouse click [buttonCheck_Balance=>$(client)]
    Mouse click [buttonDone=>$(client)]
    Mouse click [buttonLogout2=>$(client)]
Windows and components
```

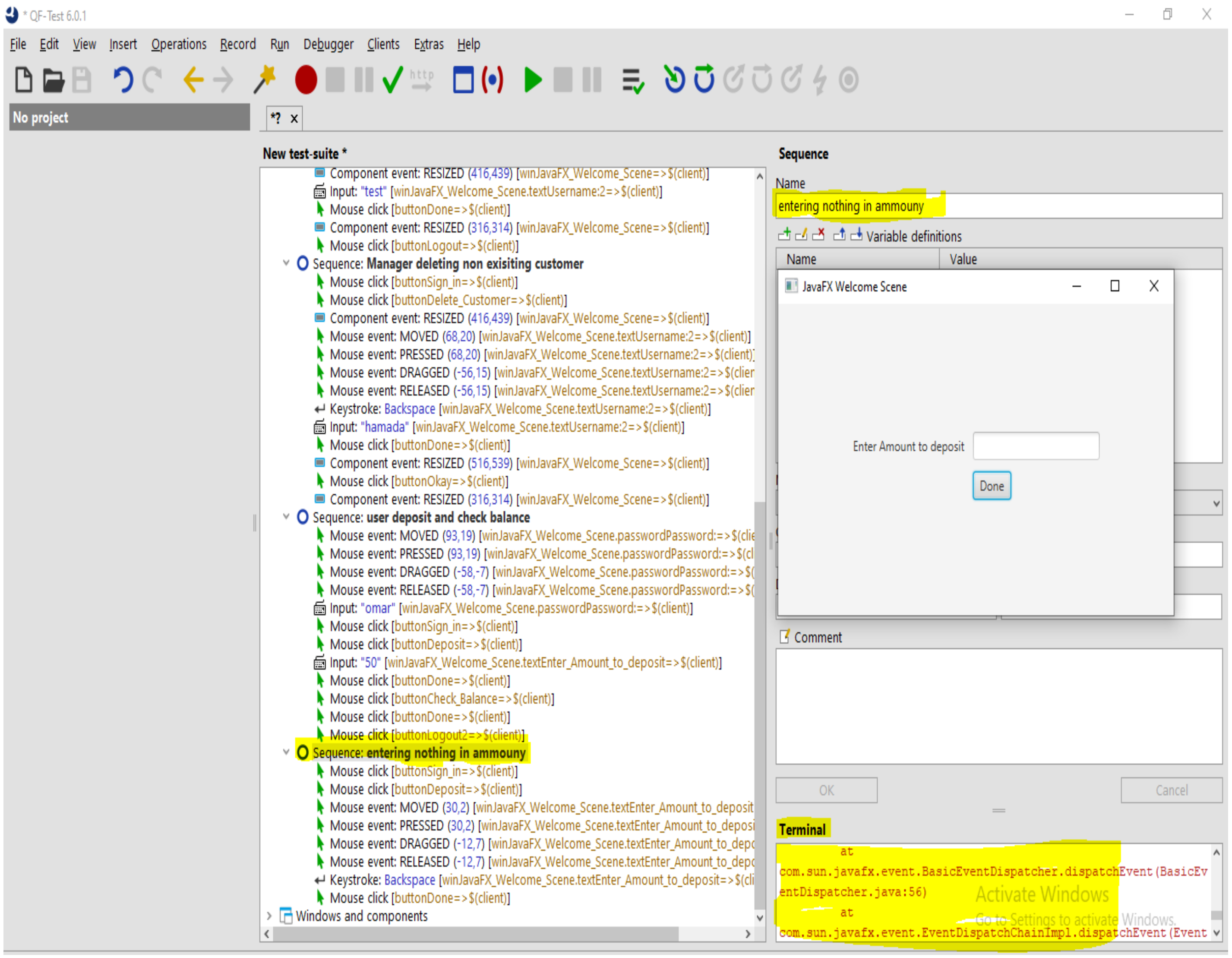
Sequence
Name: user deposit and check balance

Enter Amount to deposit: 50

```
----- QF-Test -----
Client app connected for engine fx0, Java 1.8.0_333 64bit
```

We found no error in his sequence and the logic also goes well because the user entered an amount and he previously has 50 in his credit so now he has 100 in total which also true

Some sequences will give us error , for example when we want to enter nothing in deposit



```
New test-suite *
      Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
      Input: "test" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
      Mouse click [buttonDone=>$(client)]
      Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
      Mouse click [buttonLogout=>$(client)]
  Sequence: Manager deleting non exisiting customer
      Mouse click [buttonSign_in=>$(client)]
      Mouse click [buttonDelete_Customer=>$(client)]
      Component event: RESIZED (416,439) [winJavaFX_Welcome_Scene=>$(client)]
      Mouse event: MOVED (68,20) [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
      Mouse event: PRESSED (68,20) [winJavaFX_Welcome_Scene.textUsername:2=>$(client)
      Mouse event: DRAGGED (-56,15) [winJavaFX_Welcome_Scene.textUsername:2=>$(clier
      Mouse event: RELEASED (-56,15) [winJavaFX_Welcome_Scene.textUsername:2=>$(clier
      Keystroke: Backspace [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
      Input: "hamada" [winJavaFX_Welcome_Scene.textUsername:2=>$(client)]
      Mouse click [buttonDone=>$(client)]
      Component event: RESIZED (516,539) [winJavaFX_Welcome_Scene=>$(client)]
      Mouse click [buttonOkay=>$(client)]
      Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
  Sequence: user deposit and check balance
      Mouse event: MOVED (93,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(clie
      Mouse event: PRESSED (93,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(cl
      Mouse event: DRAGGED (-58,-7) [winJavaFX_Welcome_Scene.passwordPassword:=>$(
      Mouse event: RELEASED (-58,-7) [winJavaFX_Welcome_Scene.passwordPassword:=>$(
      Input: "omar" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
      Mouse click [buttonSign_in=>$(client)]
      Mouse click [buttonDeposit=>$(client)]
      Input: "50" [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(client)]
      Mouse click [buttonDone=>$(client)]
      Mouse click [buttonCheck_Balance=>$(client)]
      Mouse click [buttonDone=>$(client)]
      Mouse click [buttonLogout2=>$(client)]
  Sequence: entering nothing in ammouny
      Mouse click [buttonSign_in=>$(client)]
      Mouse click [buttonDeposit=>$(client)]
      Mouse event: MOVED (30,2) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit
      Mouse event: PRESSED (30,2) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposi
      Mouse event: DRAGGED (-12,7) [winJavaFX_Welcome_Scene.textEnter_Amount_to_depo
      Mouse event: RELEASED (-12,7) [winJavaFX_Welcome_Scene.textEnter_Amount_to_depo
      Keystroke: Backspace [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(cli
      Mouse click [buttonDone=>$(client)]
  Windows and components
```

Sequence

Name: entering nothing in ammouny

JavaFX Welcome Scene

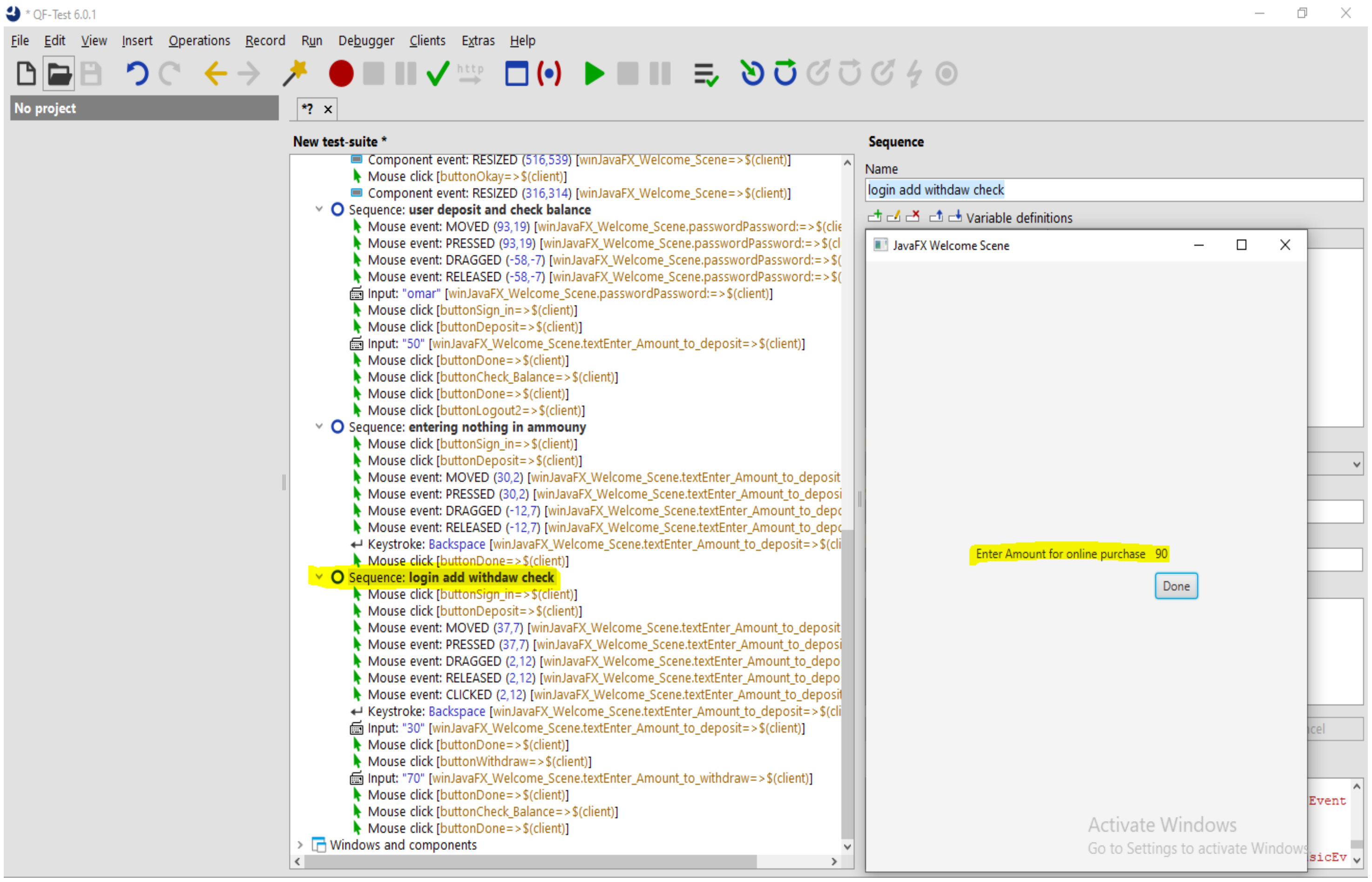
Enter Amount to deposit

Terminal

```
        at
com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEv
entDispatcher.java:56)
        at
com.sun.javafx.event.EventDispatchChainImpl.dispatchEvent(Event
```

The terminal gives us some red errors in the highlighted area , because nothing is written in the text field and we want to read it in the program

This time we get a more complex sequence , by login we  add amount then withdraw an amount after that we  check the balance



login add withdaw check

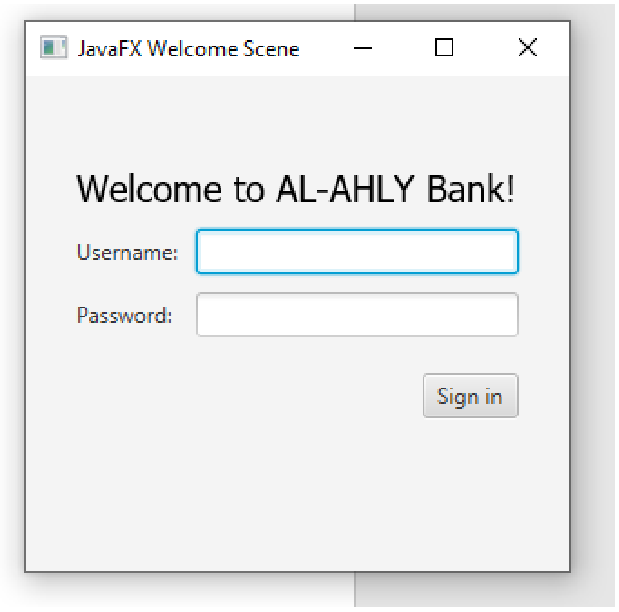
Enter Amount for online purchase  90

```
        Component event: RESIZED (516,539) [winJavaFX_Welcome_Scene=>$(client)]
        Mouse click [buttonOkay=>$(client)]
        Component event: RESIZED (316,314) [winJavaFX_Welcome_Scene=>$(client)]
    Sequence: user deposit and check balance
        Mouse event: MOVED (93,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(clie
        Mouse event: PRESSED (93,19) [winJavaFX_Welcome_Scene.passwordPassword:=>$(cl
        Mouse event: DRAGGED (-58,-7) [winJavaFX_Welcome_Scene.passwordPassword:=>$(
        Mouse event: RELEASED (-58,-7) [winJavaFX_Welcome_Scene.passwordPassword:=>$(
        Input: "omar" [winJavaFX_Welcome_Scene.passwordPassword:=>$(client)]
        Mouse click [buttonSign_in=>$(client)]
        Mouse click [buttonDeposit=>$(client)]
        Input: "50" [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(client)]
        Mouse click [buttonDone=>$(client)]
        Mouse click [buttonCheck_Balance=>$(client)]
        Mouse click [buttonDone=>$(client)]
        Mouse click [buttonLogout2=>$(client)]
    Sequence: entering nothing in ammouny
        Mouse click [buttonSign_in=>$(client)]
        Mouse click [buttonDeposit=>$(client)]
        Mouse event: MOVED (30,2) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit
        Mouse event: PRESSED (30,2) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposi
        Mouse event: DRAGGED (-12,7) [winJavaFX_Welcome_Scene.textEnter_Amount_to_depo
        Mouse event: RELEASED (-12,7) [winJavaFX_Welcome_Scene.textEnter_Amount_to_depo
        Keystroke: Backspace [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(cli
        Mouse click [buttonDone=>$(client)]
    Sequence: login add withdaw check
        Mouse click [buttonSign_in=>$(client)]
        Mouse click [buttonDeposit=>$(client)]
        Mouse event: MOVED (37,7) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit
        Mouse event: PRESSED (37,7) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposi
        Mouse event: DRAGGED (2,12) [winJavaFX_Welcome_Scene.textEnter_Amount_to_depo
        Mouse event: RELEASED (2,12) [winJavaFX_Welcome_Scene.textEnter_Amount_to_depo
        Mouse event: CLICKED (2,12) [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit
        Keystroke: Backspace [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(cli
        Input: "30" [winJavaFX_Welcome_Scene.textEnter_Amount_to_deposit=>$(client)]
        Mouse click [buttonDone=>$(client)]
        Mouse click [buttonWithdraw=>$(client)]
        Input: "70" [winJavaFX_Welcome_Scene.textEnter_Amount_to_withdraw=>$(client)]
        Mouse click [buttonDone=>$(client)]
        Mouse click [buttonCheck_Balance=>$(client)]
        Mouse click [buttonDone=>$(client)]
  Windows and components
```

It is clear that this sequence has no errors as the checked amount is right for the values entered in the last steps

## Manual GUI Testing:

1. Sign In Window

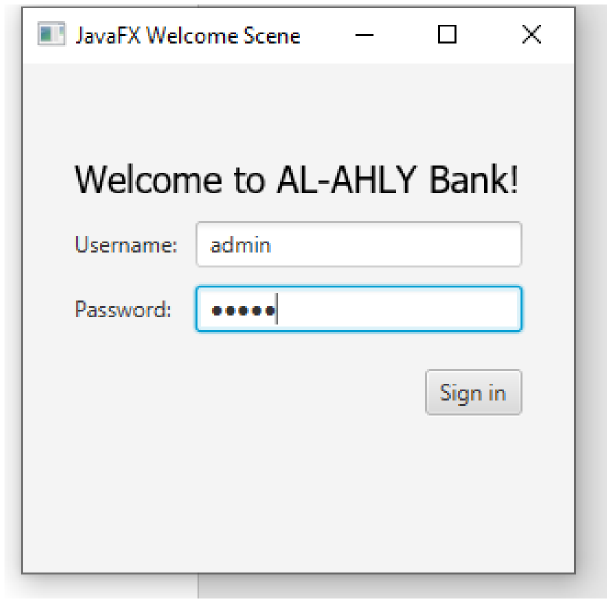

JavaFX Welcome Scene

Welcome to AL-AHLY Bank!

Username:

Password:

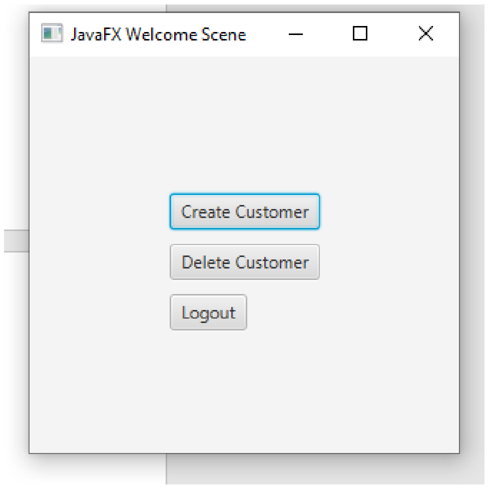
By entering a known/saved manager account with username: admin & password: admin (implemented in the code), we go to Manager Options window.


JavaFX Welcome Scene

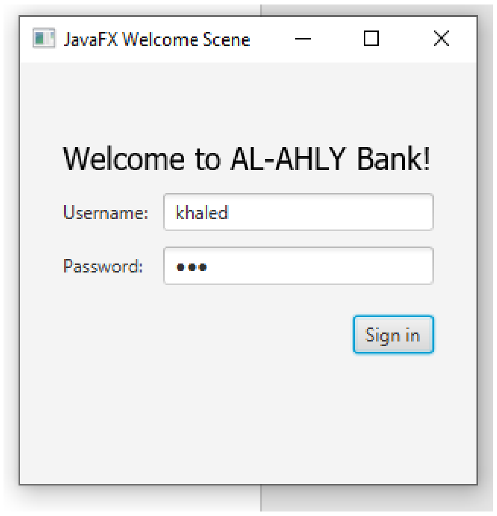
Welcome to AL-AHLY Bank!

Username: admin

Password: •••••



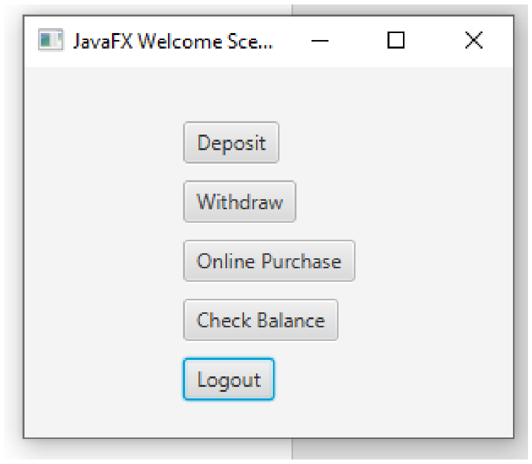
If we enter a valid username or password for a customer, Customer Options window appears.



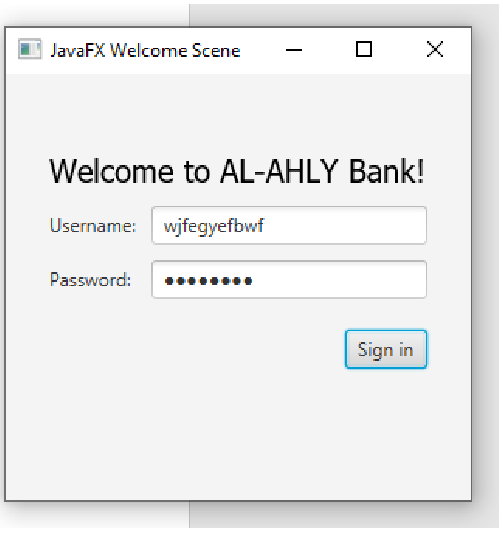
Welcome to AL-AHLY Bank!

Username: khaled

Password: •••



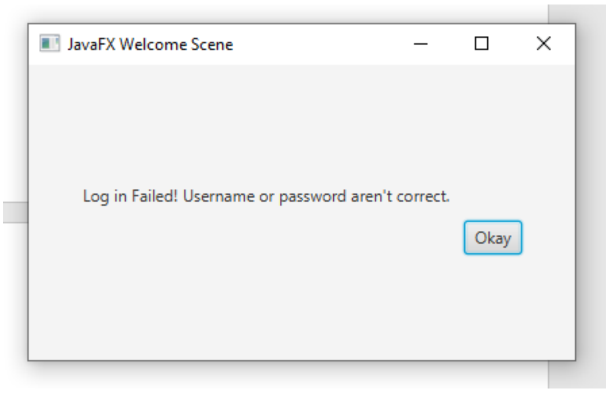
Error Windows:
If we enter invalid username or password, we get the “Failed to login” error window.



Welcome to AL-AHLY Bank!

Username: wjfegyefbwf

Password: ●●●●●●●●

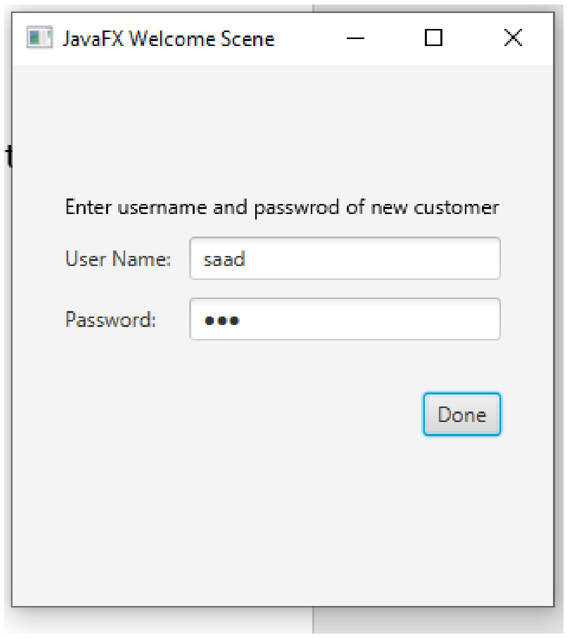


Log in Failed! Username or password aren't correct.

2. Manager Options Window

I. Create Customer

Simply type a non-used username and a password for the created customer.


JavaFX Welcome Scene

Enter username and passwrod of new customer

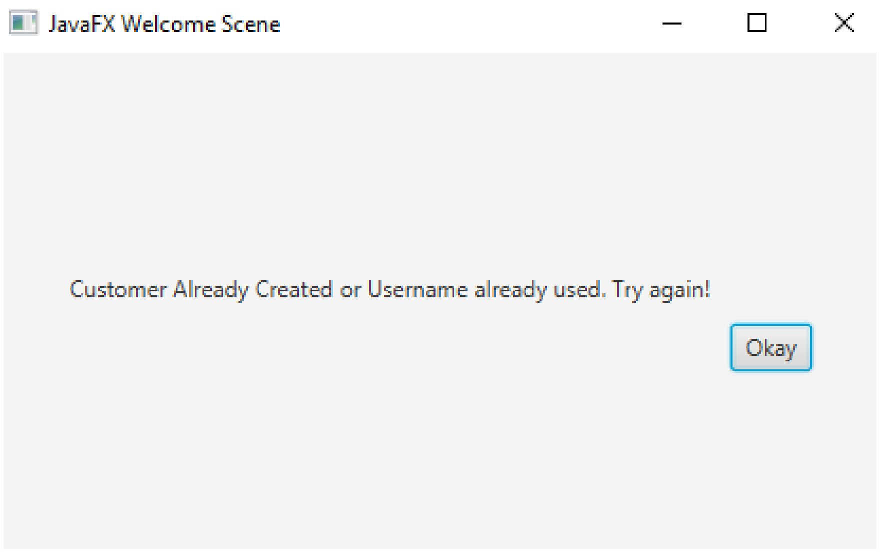
User Name: saad

Password: ●●●

Done

Error Window:

If we enter same username, a "customer already used" error will pop up.

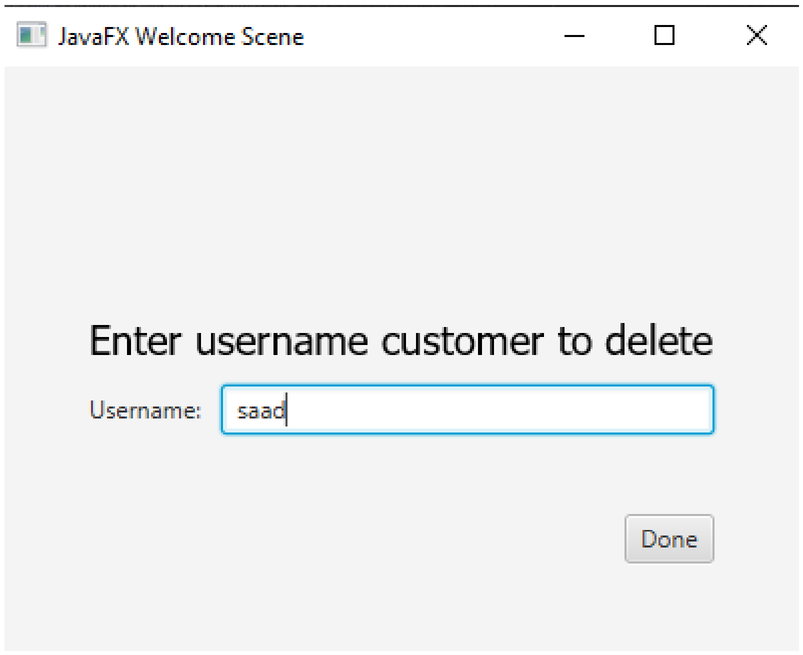

JavaFX Welcome Scene

Customer Already Created or Username already used. Try again!

Okay

II. Delete Customer

By entering the username of a customer, you can delete him.

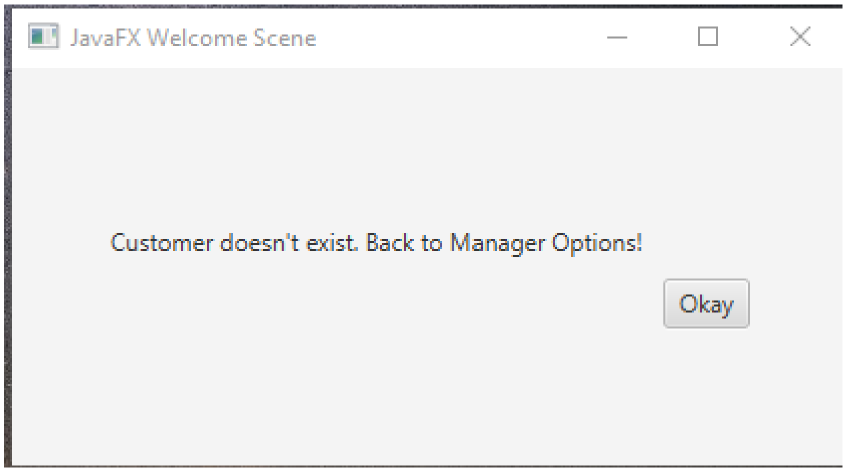


Enter username customer to delete

Username: saad

Error Window:

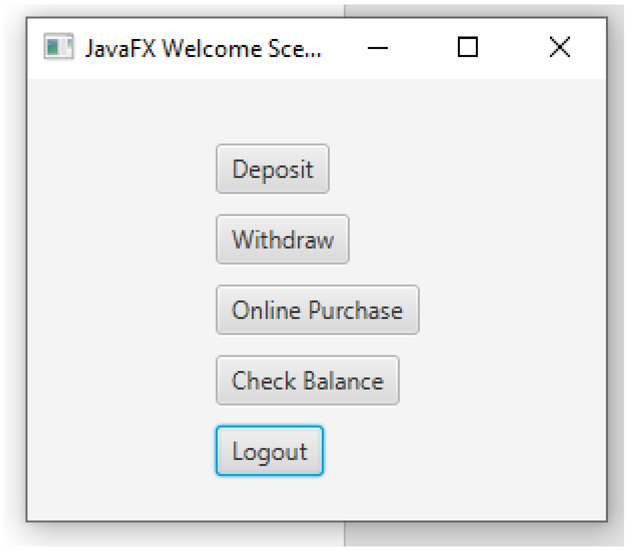
If we enter a username that doesn’t exist or already deleted, this window will pop up.



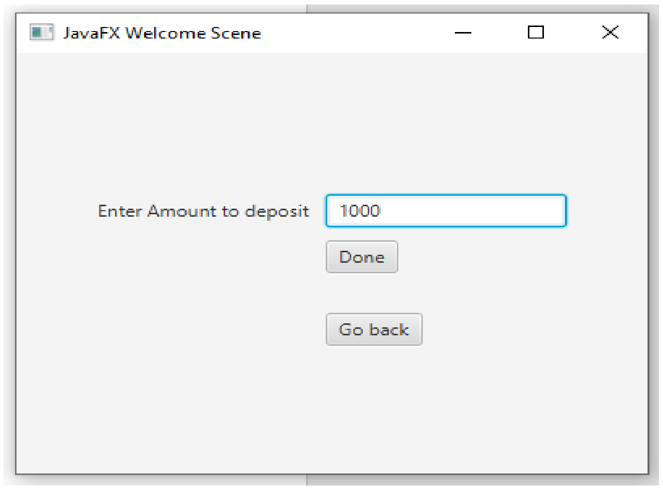
Customer doesn't exist. Back to Manager Options!

3. Customer Options

After entering a valid customer information. Every Customer account has initially 100 Pounds when created and with Silver Subscription which is -20 pounds for every online purchase.

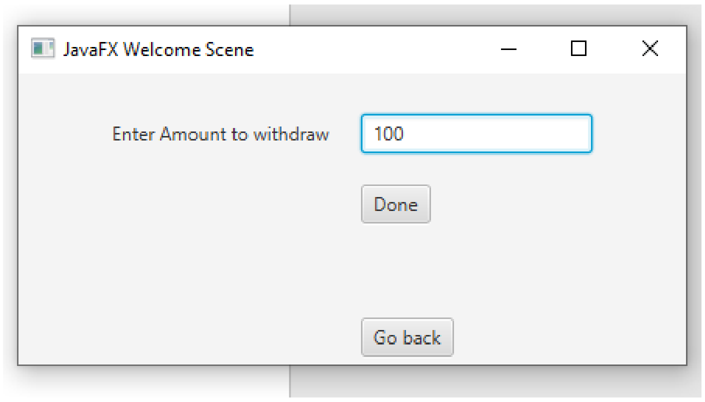


I. Deposit



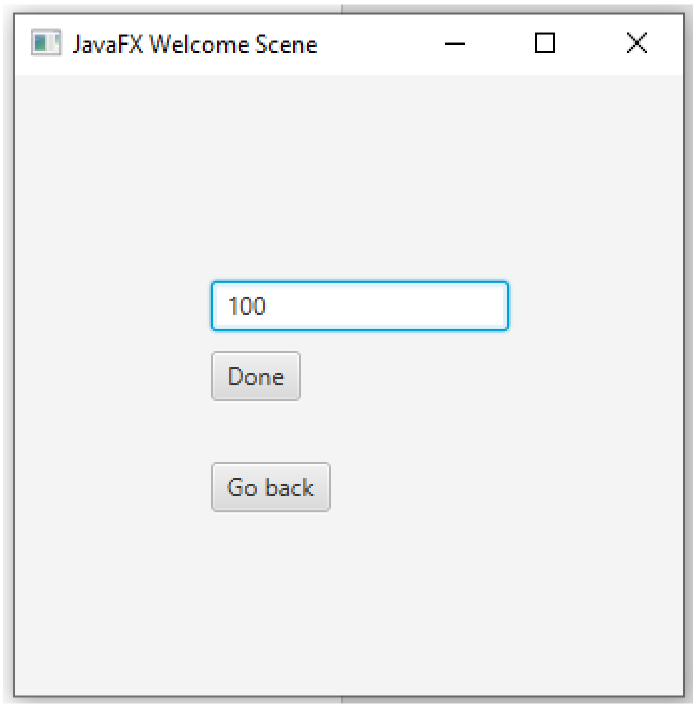
Enter Amount to deposit 1000

II. Withdraw



III. Online Purchase

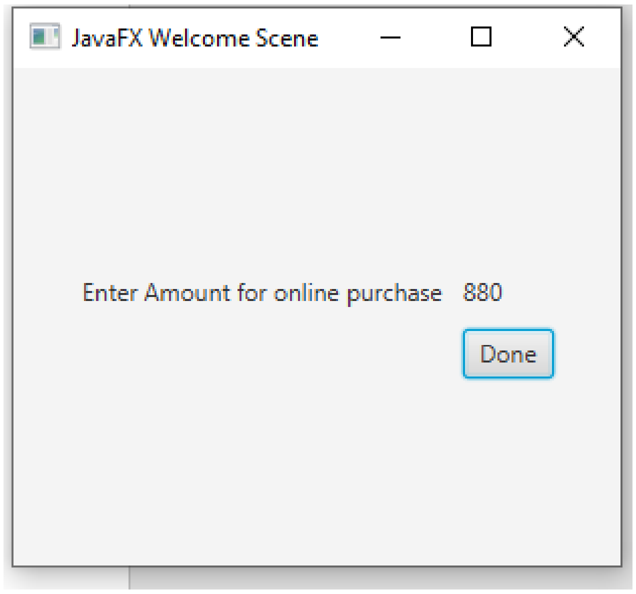
It takes the value of your purchase and takes it from your balance.



We deposited 1000, withdrawn 100, purchased online 100, initial balance of 100 and Sliver Subscription for 20. So total is: 1000 – 100 – 100 + 100 – 20 = **880 Pounds**.

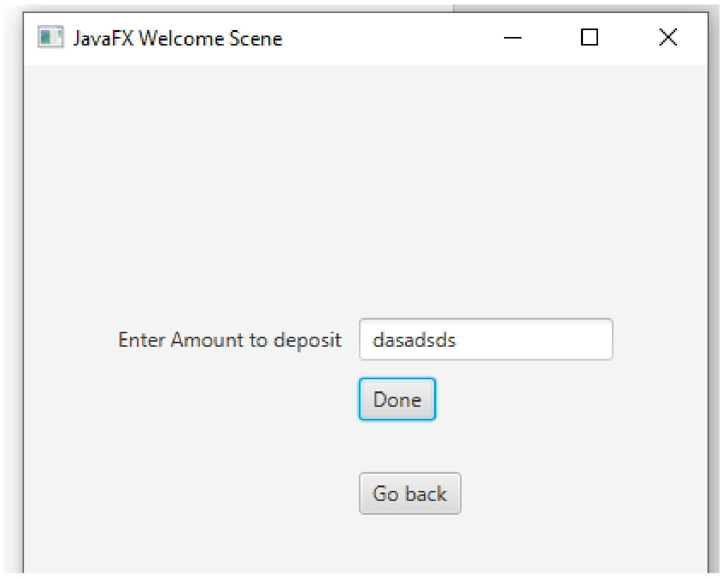
Now, lets try checking balance and see if the expected calculation is correct.

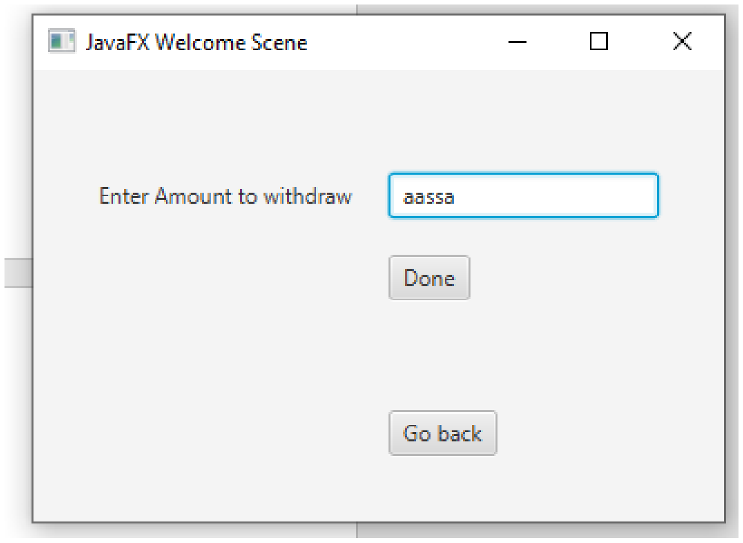
IV. Check Balance

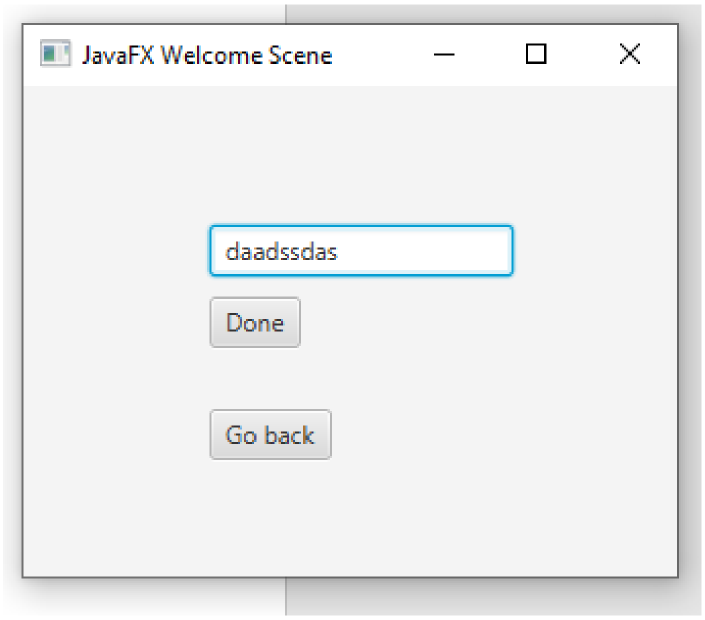


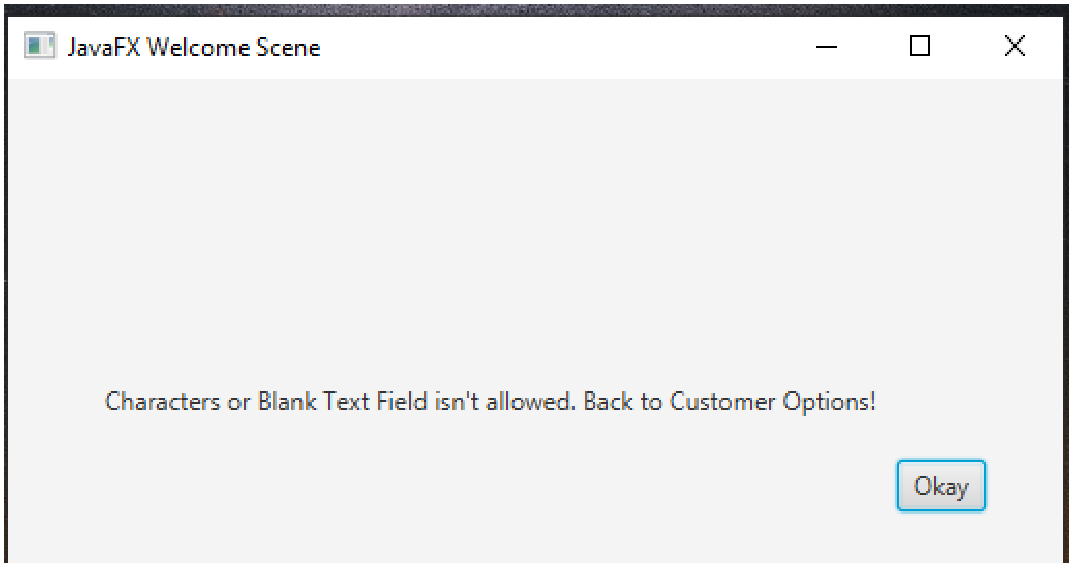
Enter Amount for online purchase 880

Error Windows:
When you enter at Deposit or Withdraw or Online Purchase Windows, characters or Blank instead of numbers, the upcoming error window appears.



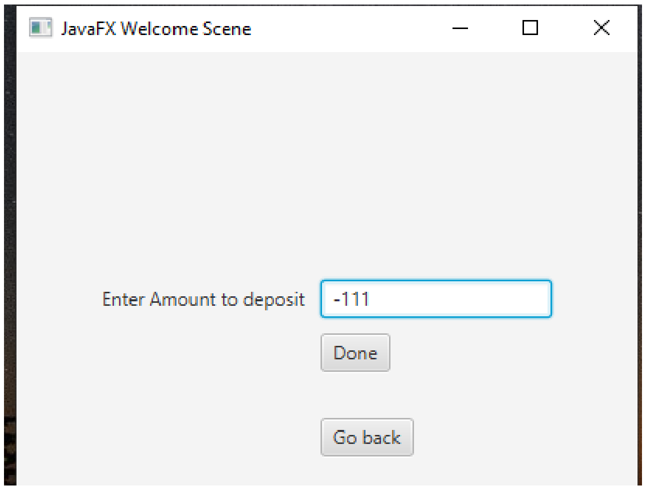




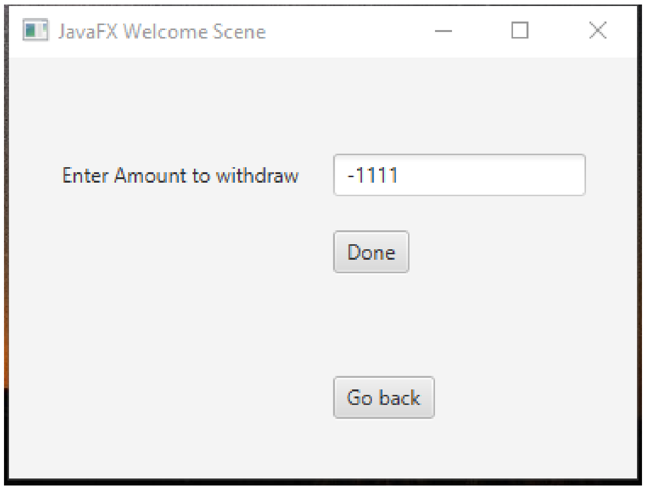


Characters or Blank Text Field isn't allowed. Back to Customer Options!

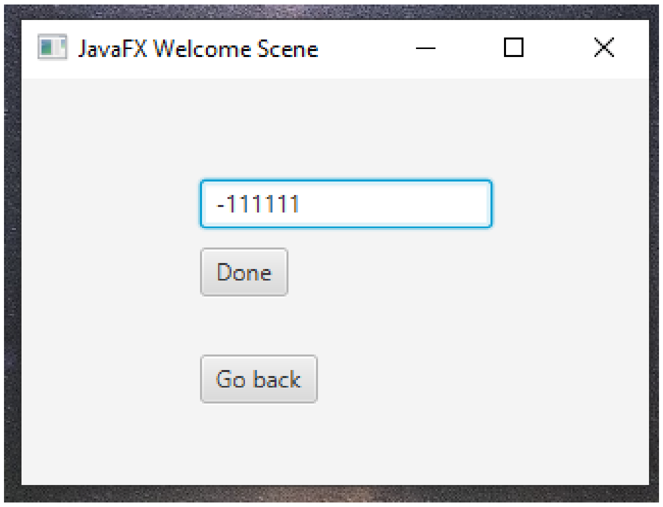
When entering in the same windows negative values or zero, the upcoming error window appears.



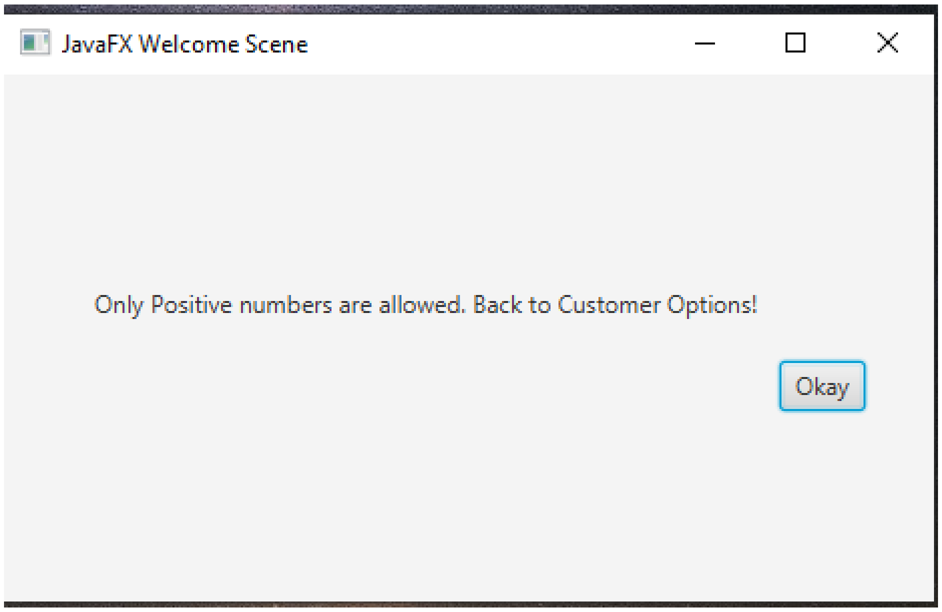
Enter Amount to deposit -111



Enter Amount to withdraw -1111



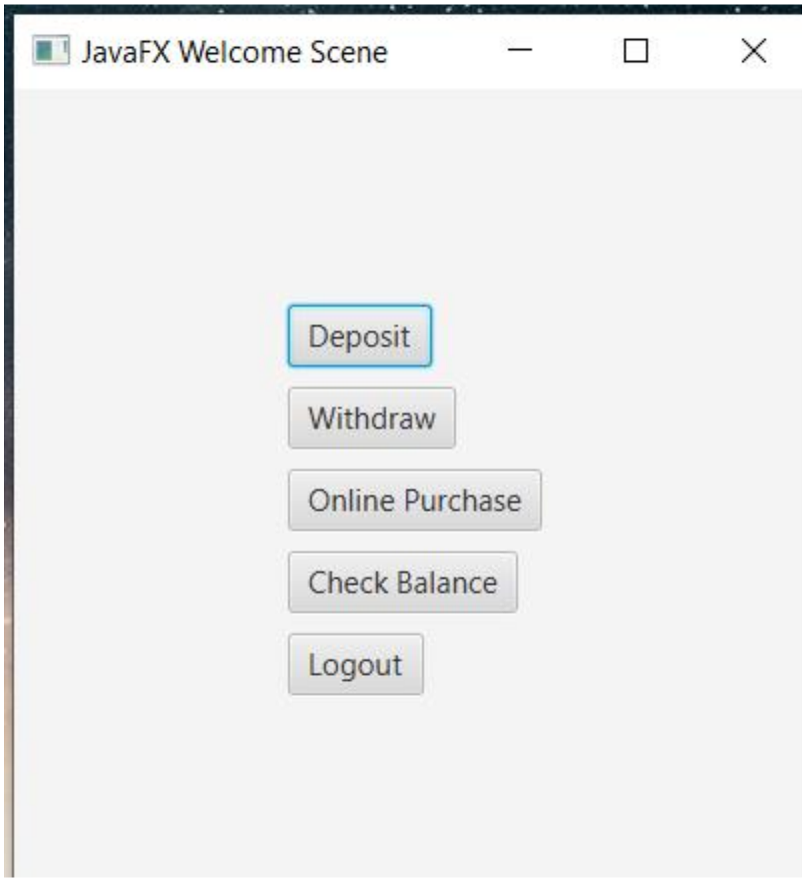
-111111



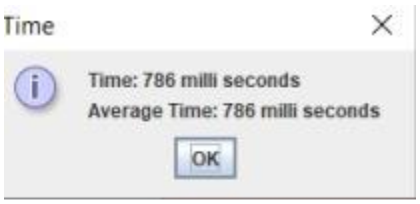
Only Positive numbers are allowed. Back to Customer Options!

## Performance testing:

when you click to call any function there is a label will show you the time that function takes and it will show you if you press more than one time.

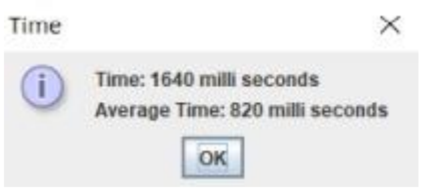


### After one call function



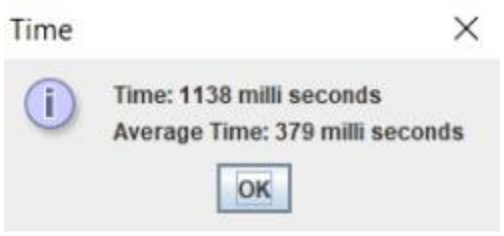
Time: 786 milli seconds
Average Time: 786 milli seconds

### After two call function



Time: 1640 milli seconds
Average Time: 820 milli seconds

## After three call function



```
Time: 1138 milli seconds
Average Time: 379 milli seconds
```