Software engineering
Project

Author: Mehran Nasser
Mahboobeh Abdal
KTH 10/10/2012

Table of Contents

1. User stories

User Stories	Imp status
1. As an employee, they need to login to the system, so that they can access available functionalities	✓
2. Customers need to log into the system, so that they can submit their claim.	✓
3. As a CD employee, I want to be able to receive customers' claim online so that I can register the claim faster.	✓
4. As a CD employee, I want to be able to access customers' claim at any time so that it will be possible to check the complete history.	✓
5. As a CD employee, I want to have access to the list of insured customers so that I can check and accept clime of insured customer.	✓
6. As a CD employee, I want to have access to the customer history, to classify their claim into complex OR simple	✓
7. In order to handle the simple and complex claim CD employee needs to check the insurance as the first step of decision-making.	✓
8. As a CD employee, I want to be able to send a web form to the garage so that they can inform us about the severity of damage and repair cost.	
9. As a utility, returned form from the garage need to be recorded and accessible by CD employees so that they can access it any time to make decision. This is the second task of the simple claims and third task of the complex claims	
10. In order to handle the complex claim CD employee needs to check the history of the customer as second step of decision-making.	✓
11. As a CD employee, I want to be able to inform the client about the decision through the system so that we can achieve better customer relationship management.	
12. As a CD employee, I want to be able to record decisions regarding a particular claim so that all decision will be accessible by other employees.	
13. As a CD employee, I should be able to send the relevant form to the claimant depend on claim classification.	
14. Customers should fill all of the fields in the form so that It can be stored in the system	✓
15. As financial employee, I want to be notified about the handlers' decision so that I can perform the payment.	
16. As financial employee, I should be able to deposit payment to the customer account so that they can receive money immediately.	
17. As financial employee, I should be able to record payment history for each customer so that it can be used for accounting purpose.	

2. Release planning

This step consists of both customers and developers in which all requirements will be determined, in which near-term releases, and when they will be delivered to customers. Release Planning consist of three phases:

- Exploration Phase: Customer will give selected high-value requirements for the system and they will be written down on user stories card.
- Commitment Phase: Business people and developers will commit themselves to the wanted functionality and the date of the next release.
- Steering Phase: This is the phase in which plan can be adjusted, new requirements can be added and/or existing requirements can be changed or removed

2.1. **Exploration Phase**

- **Write a story:** the user stories are defined in previous task of the project.
- **Estimate a story:**

Story number	Implementation time/h
1.	1
2.	0.5
3.	0.5
4.	1
5.	1.5
6.	1
7.	1
8.	0.5
9.	1
10.	1.75
11.	1
12.	1
13.	1
14.	0.25
15.	0.25
16.	1
17.	0.5

2.2. **Commitment Phase**

- **Sort by Value**

Story number	Value (Critical, Significant, Nice to have)
1.	Critical
2.	Critical
3.	Significant
4.	Significant
5.	Critical
6.	Critical
7.	Critical
8.	Nice to have
9.	Nice to have
10.	Critical
11.	Significant
12.	Critical
13.	Nice to have
14.	Significant
15.	Significant
16.	Significant
17.	Critical

- **Sort by risk**

We gave each user story an index from 0 to 2

- Completeness (do we know all of the story details?). 0=complete, 1=incomplete, 2=unknown
- Volatility (is it likely to change?). 0=low, 1=medium, 2=high
- Complexity (how hard is it to build?). 0=low, 1=medium, 2=high

All indexes for a user story will be added, assigning the user stories a risk index of **low** (0–1)=**L**, **medium** (2–4)=**M**, or **high** (5–6)=**H**

Story number	Completeness	Volatility	Complexity	Risk
1.	0	0	2	M
2.	0	0	2	M
3.	1	1	0	M
4.	1	1	1	M
5.	1	1	1	M
6.	1	1	1	M
7.	1	1	1	M
8.	1	1	0	M
9.	1	1	0	M
10.	1	1	1	M
11.	1	1	0	M
12.	0	0	0	L
13.	1	1	0	M
14.	1	1	0	M
15.	1	1	0	M
16.	1	1	1	M
17.	1	1	0	M

- **Set Velocity:** it is assumed that the project requires four full working days to release.
- **Choose scope:** The user stories that will be finished in the next release will be picked. The selection was based on available time, risk, business value and dependencies between stories. The first release is supposed to be lunch by 9 October. First release include these stories: 1,2,3,4,5,6,7,10,14.

3. Iteration planning

It assumed that each release consists of three iterations. Iterations include the stories that have sequential dependencies with each other. In the first iteration, stories number one, two, three, and fourteen were considered together. In the second iteration, user stories number five and seven implemented. The last iteration was included the rest of the stories of the release plan, namely four, six, and ten.

3.1. Exploration phase

In this phase, all requirements were translated into set of tasks. Due to lack of enough programming knowledge, we could not make a correct estimation of the required time for each task.

3.2. Commitment phase

Commitment is about to share the workload between two programmers. This is not applicable to this project, since we were only group of two students and XP necessitate use of pair programming approach.

3.3. Steering phase

The implementation of the tasks is done during the steering phase of the iteration planning.

- Find a Partner: we were already group of two students to practice Pair Programming.
- Design the task: we worked together to design required functionality of the selected task.
- Write unit test: Before we start coding the functionality, we first write automated tests using Junit.
- Write code.
- Run test: The unit tests are run to test the code.
- Refactoring: testing and modification of the codes if necessary. We focused on improving code readability, reducing complexity, removing unnecessary imports as well as fixing syntax errors.

4. Pair programming

We worked alongside at one system and we were cooperating on the same design, algorithm, program code, and test. One of us as driver typed at the computer and wrote down a design. The other one as a navigator had many responsibilities. One of them was to look at the work of the driver and trying to find tactical and strategic defects in driver's work. One of the tactical defects that driver caused was calling wrong method but the navigator informed him immediately.

We brainstormed at any time was needed. We were both active in our position and communicate continually. Albeit we switched our roles between driver and navigator therefore, everyone knows what everyone is doing and everybody remains familiar with the whole system.

5. Refactoring

a) We got nullPointerException when we run the test for Access.log. therefore, we added 'if' statement for not getting a null pointer exception

```
if (u==null || p==null) {
return false;
}
```

b) The second failed again because we had wrong condition for the 'if' statement. We had condition to reject access if username or password fields was empty but we had used 'and' notation instead of 'or'.

```
 if (u==null && p==null) {
return false;
}
```

c) During implementation of 'Customer' class, we got plenty of syntax error. Solving these errors automatically solved semantic errors as well. Therefore, the written test for the 'Customer' class passed successfully at the first try. In addition, writing the test for the methods of this class required us to read more about testing of our expecting exception and refactoring the body of both test method and actual method several times and finally solving issue by throwing the exception in the signature of the method.

d) In the next step to design the claim form, we decided to show the customer ID automatically according to the customer user name. To achieve this, we added the customer ID corresponding to each user name in a line above it in the text file.

The text file looks like this:

Id
Username
Pass

Then, the body of the 'logCustomer()' was changed and 'readFile.next()' was added inside the 'while' statement to skip the id line when program wants to authorize access. The test failed after applying the change and NoSuchElementException was received. To solve the problem, the algorithm of the logCustomer() was reviewed and debugged several times. The problem was solved by refactoring the method as follow:

```
while(readFile.hasNext())
{
readFile.next();
user=readFile.next();
pass=readFile.next();
if(u.equals(user)&& p.equals(pass))
{
found=true;
break;
}
```

The same algorithm applied to the logEmployee() as it has the same behavior as logCustomer()

e) In order to pass the customer id to the ClaimFrame the getId() method is used. The test method for the getId() is written to see if it behaves as it expected. getId() must written null if user not exist or return the userId of the corresponding authorized user. However, the method could not pass the test. Refactoring was easy since we just had to reinitialize the value of variable (ID) at the start of each loop and setting the value to null.

6. User acceptance testing

Original Author: Mahboobeh & Mehran Original Date: 2012/10/09 Test Objective: To test the function of logging into the XXX system.						
Item No.	Test Condition	Operator Action	Input Specifications	Output Specifications (Expected Result)	Pass or Fail	Comments
1	Successful Login	1 - Enters Username 2 - Enters Password 3 - Selects "Login."	1 - Valid Username 2 - Valid Password	1- System displays dashboard for employees or claim page for customers.	Passed	
2	Invalid Username or Password	1- Enters Username 2 - Enters Password 3 - Selects "Login."	1 - Invalid Username or Password	1- System displays "Wrong username or password " error message and returns to step 1.	Passed	

Original Author: Mahboobeh & Mehran Original Date: 2012/10/09 Test Objective: To test the system XXX ability to save claims						
Item No.	Test Condition	Operator Action	Input Specifications	Output Specifications (Expected Result)	Pass or Fail	Comments
1	Successful submission	1 – login as a customer 2 – Enter cost and date 3 – Select "submit"	1– all fields are filled	1 – System saves claim in corresponding claim file. Or create a new file and save data if there is not any claim file. 2 – System displays success message. 3- System closes the form	Passed	Check to not overwrite the existing claims if any
2	Unsuccessful submission	1 – login as a customer 2 – Enter cost 3 – Select "submit	1- date field is empty	1– System displays error message. 2- System waits for new data entry.	Passed	
3	Unsuccessful submission	1 – login as a customer 2 – Enter date 3 – Select "submit	1- cost field is empty	1– System displays error message. 2- System waits for new data entry.	Passed	

7. Daily stand-up meeting

In the stand up meeting we were supposed to answer to three important questions as follow:

- What did we accomplish yesterday?
- What will we do today?
- What obstacles are impeding our progress?

Since we were always together, we knew about every details of the project, its difficulties and obstacles. So we had nothing to share or ask for suggestion about specific problem. In other words, during the day we used to work on the same issue, solving the problem and moving to the next step.

Appendixes

Appendix 1

Access class

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Project;


import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;


/**
 *
```

```
 * @author Mehran & Mahboobeh
 */
public class Access {
    /**
     *
     */
    public Access () {
    }
    static String ID;

    public static boolean logCustomer (String u, String p)
    {
        String path="customer account.txt";
        boolean found = false;
        String user;
        String pass;
        String tempID;
        if (u==null || p==null) {
            return false;
        }
        else {
            try {
                Scanner readFile;
                    readFile = new Scanner(new File(path));
                    ID=null;
                while(readFile.hasNext())
                {
```

```
                tempID=readFile.next();
                user=readFile.next();
                pass=readFile.next();
               if(u.equals(user)&& p.equals(pass))
               {
                  found=true;
                  ID=tempID;
                  break;
               }
               }
                  readFile.close();
            }

            catch (FileNotFoundException ex) {
               Logger.getLogger(Access.class.getName()).log(Level.SEVERE, null, ex);
               JOptionPane.showMessageDialog(null, "error 12324# conection faild");
            }
      }
         if (found) {
            return true;
         }
            else{
            return false;
         }
   }

   public static String getID(){
```

```
    return ID;
}

public static boolean logEmployee (String u, String p)
{
    String path="account.txt";
    boolean found= false;
    String user;
    String pass;
    if (u==null || p==null) {
        return false;
    }

    else{

try {
        Scanner readFile;
            readFile = new Scanner(new File(path));
        while(readFile.hasNext())
        {
             user=readFile.next();
             pass=readFile.next();
        if(u.equals(user)&& p.equals(pass))
        {
            found=true;
            break;
```

```
            }
            }
                readFile.close();
    }
        catch (FileNotFoundException ex) {
            Logger.getLogger(LoginFrame.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(null, "error 12324# conection faild");
                    }
        }

         if (found) {
            return true;
        }
            else {
            return false;
        }
}
}
```

Appendix 2

Access Test

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Project;
import java.io.FileNotFoundException;
import junit.framework.TestCase;


/**
 *
 * @author Mehran & Mahboobeh
 */
public class AccessTest extends TestCase {
    public AccessTest(String testName) {
        super(testName);
    }
```

```
/**
 * Test of logEmployee method, of class Access.
 */
public void testLogEmployee() throws FileNotFoundException {
    System.out.println("log");
    assertFalse(Access.logEmployee(null, null));
    assertFalse(Access.logEmployee("mahboobeh", null));
    assertFalse(Access.logEmployee(null, "mahboobeh"));
    assertFalse(Access.logEmployee("mah", "mahboobeh"));
    assertFalse(Access.logEmployee("mahboobeh", "mahboo"));

    assertFalse(Access.logEmployee("mahboobeh", "mahboobeh"));

    assertTrue(Access.logEmployee("mahboobeh", "mah"));
}

public void testLogCustomer() throws FileNotFoundException {
    System.out.println("log");
    assertFalse(Access.logCustomer(null, null));
    assertFalse(Access.logCustomer("mm", null));
    assertFalse(Access.logCustomer(null, "mm"));

    assertFalse(Access.logCustomer("m", "mm"));
    assertFalse(Access.logCustomer("mm", "m"));

    assertFalse(Access.logCustomer("nn", "nn"));
```

```
        assertTrue(Access.logCustomer("nn", "mm"));
    }

    public void testgetID(){
        System.out.println("ID");
        if (Access.logCustomer("nn", "mm")){
            assertEquals("2", Access.getID());
        }
        if (Access.logCustomer("nn", "nn")==false){
            assertNull(Access.getID());
        }
    }
}
```

Appendix 3

Customer class

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Project;


import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.util.Scanner;
import java.util.Vector;
/**
 *
 * @author Mehran & Mahboobeh
 */
public class Customer {
```

```
public Customer () {
}
// return CUSTOMER INFO
public static Vector<String> getInfo(String id)throws FileNotFoundException
{
    Vector<String> vec= new Vector<>();
    String path=id+ ".txt";

        Scanner readFile;
            readFile = new Scanner(new File(path));
        while (readFile.hasNext())
        {
            vec.add(readFile.next());
        }
            readFile.close();
    return vec;
}
// return CLAIM
public static Vector<String> getClaim(String id)throws FileNotFoundException
{
            Vector<String> vec2= new Vector<>();
            String path2=id+ "claim.txt";


        Scanner readFile;
            readFile = new Scanner(new File(path2));
```

```
            while (readFile.hasNext())
          {
            vec2.add(readFile.next());
          }
            readFile.close();
        return vec2;
    }
    public static boolean setClaim(String id, String c, String d){
        String file= id+"claim.txt";
        boolean s=false;
        try {
             BufferedWriter bw;
          bw = new BufferedWriter(new FileWriter(new File(file), true));
                    bw.write(c);
                    bw.newLine();
                    bw.write(d);
                    bw.newLine();
                    bw.write("nothandled");
                    bw.close();
                    s= true;
            } catch (Exception e) {
            }
        return s;
    }
}
```

Appendix 4

Customer Test

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Project;

import java.io.FileNotFoundException;
import java.util.Vector;
import junit.framework.TestCase;

/**
 *
 * @author Mehran & Mahboobeh
 */
public class CustomerTest extends TestCase {

    public CustomerTest(String testName) {
```

```
    super(testName);
}

/**
 * Test of getInfo method, of class Customer.
 */
public void testGetInfo() throws Exception {
          try
    {
        System.out.println("getInfo");

        Vector expResult = new Vector();
        // test if correct elements in a correct order is added to the vector
        expResult.add("mahboobeh");
        expResult.add("y");
        expResult.add("23/3");
        expResult.add("12341344");
        expResult.add("3/4");
        expResult.add("283456");
        expResult.add("4/2");
        expResult.add("8264");
        expResult.add("5/1");
        expResult.add("2344");
        Vector result = Customer.getInfo("2");
        assertEquals(expResult, result);

        // test if last content of the vector is replaced with new elements
```

```
            expResult.removeAllElements();
            expResult.add("mehran");
            expResult.add("x");
            expResult.add("12/5");
            expResult.add("234341");
            result = Customer.getInfo("1");
            assertEquals(expResult, result);
            }
        catch (FileNotFoundException ex)
                {
                }
        try
        {
//  test  if method throws exception when there is no claim record for a particular cus-
            tomer
            Vector expResult = new Vector();
            expResult.removeAllElements();
            Vector result = Customer.getInfo("4");
            assertEquals(expResult, result);
            fail("FileNotFoundException was expected");
        }
        catch (FileNotFoundException ex)
                {
                }
    }
    /**
     * Test of getClaim method, of class Customer.
```

```
 */
public void testGetClaim() {
    try
    {
        System.out.println("getClaim");

        Vector expResult = new Vector();
        // test  if correct elements in a correct order is added to the vector
        expResult.add("123456");
        expResult.add("9/8");
        Vector result = Customer.getClaim("1");
        assertEquals(expResult, result);

        // test if last content of vector is replaced with new elements
        expResult.removeAllElements();
        result = Customer.getClaim("2");
        assertEquals(expResult, result);
        }
    catch (FileNotFoundException ex)
          {
          }
    try
    {
// test if method throws exception when there is no claim record for a particular cus-
            tomer
        Vector expResult = new Vector();
        expResult.removeAllElements();
```

```
        Vector result = Customer.getClaim("4");
        assertEquals(expResult, result);
        fail("FileNotFoundException was expected");
    }
    catch (FileNotFoundException ex)
        {
        }
    }

public void testSetClaim() {
    System.out.println("setClaim");
    String id = "3";
    String c = "3456";
    String d = "9/8";
    boolean expResult = true;
    boolean result = Customer.setClaim(id, c, d);
    assertEquals(expResult, result);
}
    }
```

Appendix 5

Code to submit claim

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Project;

import javax.swing.JOptionPane;

/**
 *
 * @author Mehran & Mahboobeh
 */
public class ClaimFrame extends javax.swing.JFrame {

    /**
     * Creates new form ClaimFrame
     */
```

```
public ClaimFrame() {
    initComponents();
}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {
    idField = new javax.swing.JTextField();
    jLabel1 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    costField = new javax.swing.JTextField();
    dateField = new javax.swing.JTextField();
    submitB = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    idField.setEditable(false);
    idField.setText(Access.getID());
    idField.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            idFieldActionPerformed(evt);
        }
    });

    jLabel1.setText("CustomerID:");
```

```
jLabel2.setText("Cost:");


jLabel3.setText("Date:");


costField.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      costFieldActionPerformed(evt);
   }
});


submitB.setText("Submit");
submitB.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      submitBActionPerformed(evt);
   }
});


javax.swing.GroupLayout layout = new
      javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.TRAILING)
   .addGroup(layout.createSequentialGroup()

      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
      .LEADING)
```

```
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNREL
            ATED)
                .addComponent(idField, javax.swing.GroupLayout.PREFERRED_SIZE,
            77, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createSequentialGroup()
                .addGap(28, 28, 28)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
            .TRAILING)
                    .addComponent(jLabel2)
                    .addComponent(jLabel3))
                .addGap(29, 29, 29)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
            .LEADING, false)
                    .addComponent(costField)
                    .addComponent(dateField,
            javax.swing.GroupLayout.DEFAULT_SIZE, 76, Short.MAX_VALUE))))
        .addContainerGap(140, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, lay-
        out.createSequentialGroup()
        .addGap(0, 0, Short.MAX_VALUE)
        .addComponent(submitB)
        .addGap(20, 20, 20))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
.addGroup(layout.createSequentialGroup()
    .addContainerGap()

      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
      .BASELINE)

       .addComponent(idField,     javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)

       .addComponent(jLabel1))

    .addGap(18, 18, 18)

    .addComponent(jSeparator1,   javax.swing.GroupLayout.PREFERRED_SIZE,
      10, javax.swing.GroupLayout.PREFERRED_SIZE)


      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
      D)


      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
      .BASELINE)

       .addComponent(jLabel2)

       .addComponent(costField,   javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE))


      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNREL
      ATED)


      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
      .BASELINE)

       .addComponent(dateField,   javax.swing.GroupLayout.PREFERRED_SIZE,
      javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)

       .addComponent(jLabel3))


      .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
      D)
```

```
            .addComponent(submitB)
            .addContainerGap(18, Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>
private void submitBActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String cost= costField.getText();
    String date=dateField.getText();
    String id=idField.getText();
    boolean sub;
    if (!"".equals(cost) && !"".equals(date)){
        sub=Customer.setClaim(id, cost, date);
        if (sub){
        JOptionPane.showMessageDialog(null, "Your claim submited successfully");
        ClaimFrame.this.dispose();
        }
    }
    else{
        JOptionPane.showMessageDialog(null, "Please fill all required fields");
    }
}

/**
 * @param args the command line arguments
 */
```

```
public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ClaimFrame.class.getName()).log(java.
            util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ClaimFrame.class.getName()).log(java.
            util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ClaimFrame.class.getName()).log(java.
            util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ClaimFrame.class.getName()).log(java.
            util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
```

```
            new ClaimFrame().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JTextField costField;
private javax.swing.JTextField dateField;
private javax.swing.JTextField idField;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JButton submitB;
// End of variables declaration
}
```

Appendix 6

Log into the system

```
package Project;

import javax.swing.JOptionPane;
/**
 *
 * @author Mehran & Mahboobeh
 */
public class LoginFrame extends javax.swing.JFrame {

    /**
     * Creates new form LoginFrame
     */
    public LoginFrame() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
```

```
private void initComponents() {

    loginE = new javax.swing.JButton();
    userText = new javax.swing.JTextField();
    label1 = new java.awt.Label();
    label2 = new java.awt.Label();
    passField = new javax.swing.JPasswordField();
    loginC = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

    loginE.setText("Login employee");
    loginE.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            loginEActionPerformed(evt);
        }
    });
    getContentPane().add(loginE,                                    new
            org.netbeans.lib.awtextra.AbsoluteConstraints(120, 110, -1, -1));
    getContentPane().add(userText,                                  new
            org.netbeans.lib.awtextra.AbsoluteConstraints(200, 40, 108, -1));

    label1.setText("Username");
    getContentPane().add(label1,                                    new
            org.netbeans.lib.awtextra.AbsoluteConstraints(120, 40, -1, -1));

    label2.setText("Password");
```

```
    getContentPane().add(label2, new
            org.netbeans.lib.awtextra.AbsoluteConstraints(120, 70, -1, -1));

    getContentPane().add(passField, new
            org.netbeans.lib.awtextra.AbsoluteConstraints(200, 70, 110, -1));


    loginC.setText("Login customer");

    loginC.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(java.awt.event.ActionEvent evt) {

            loginCActionPerformed(evt);

        }

    });

    getContentPane().add(loginC, new
            org.netbeans.lib.awtextra.AbsoluteConstraints(270, 110, -1, -1));


    pack();

}// </editor-fold>


private void loginEActionPerformed(java.awt.event.ActionEvent evt) {

  boolean Aut;

        Aut = Access.logEmployee(userText.getText(),new
            String(passField.getPassword()));

            if (Aut)

        {

            new DashBoard().setVisible(true);

            LoginFrame.this.dispose();

        }

  else

        {
```

```
            JOptionPane.showMessageDialog(null, "WRONG USERNAME OR
              PASSWORD\n try again");

        }

}


private void loginCActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    boolean Aut;

        Aut = Access.logCustomer(userText.getText(),new
            String(passField.getPassword()));

            if (Aut)

        {

          new ClaimFrame().setVisible(true);

          LoginFrame.this.dispose();

        }

  else

        {

          JOptionPane.showMessageDialog(null, "WRONG USERNAME OR
            PASSWORD\n try again");

        }

}


/**

 * @param args the command line arguments

 */

public static void main(String args[]) {

    try {
```

```
        for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {

            if ("Nimbus".equals(info.getName())) {

                javax.swing.UIManager.setLookAndFeel(info.getClassName());

                break;

            }

        }

    } catch (ClassNotFoundException | InstantiationException | IllegalAccessExcep-
            tion | javax.swing.UnsupportedLookAndFeelException ex) {


            java.util.logging.Logger.getLogger(LoginFrame.class.getName()).log(java.u
            til.logging.Level.SEVERE, null, ex);

    }


    /* Create and display the form */

    java.awt.EventQueue.invokeLater(new Runnable() {

        @Override

        public void run() {

            LoginFrame login=new LoginFrame();

            login.setVisible(true);

        }

    });

}

// Variables declaration

private java.awt.Label label1;

private java.awt.Label label2;

private javax.swing.JButton loginC;

private javax.swing.JButton loginE;
```

```
    public static javax.swing.JPasswordField passField;
    public static javax.swing.JTextField userText;
    // End of variables declaration
}
```

Appendix 7

Employees dashboard

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Project;

import java.io.FileNotFoundException;
import java.util.Enumeration;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author Mehran & Mahboobeh
 */
```

```
public class DashBoard extends javax.swing.JFrame {

    /**
     * Creates new form DashBoard
     */
    public DashBoard() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        CustomerID = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        Search = new javax.swing.JButton();
        jSeparator1 = new javax.swing.JSeparator();
        jLabel2 = new javax.swing.JLabel();
        name = new javax.swing.JTextField();
        type = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        validity = new javax.swing.JTextField();
        jSeparator2 = new javax.swing.JSeparator();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
```

```
jLabel7 = new javax.swing.JLabel();
price = new javax.swing.JTextField();
jScrollPane2 = new javax.swing.JScrollPane();
damage_table = new javax.swing.JTextArea();
status = new javax.swing.JLabel();
jScrollPane3 = new javax.swing.JScrollPane();
claimTable = new javax.swing.JTextArea();
jRadioButton1 = new javax.swing.JRadioButton();
jRadioButton2 = new javax.swing.JRadioButton();
jButton1 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setText("CustomerID:");

Search.setText("Serach");
Search.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        SearchActionPerformed(evt);
    }
});

jLabel2.setText("Name:");

name.setEnabled(false);

type.setEnabled(false);
```

```
jLabel3.setText("Type of insurance:");

jLabel4.setText("valid until:");

validity.setEnabled(false);

jLabel5.setText("Damage History");

jLabel6.setText("Recent claim:");

jLabel7.setText("car price");

price.setEnabled(false);

damage_table.setEditable(false);
damage_table.setColumns(20);
damage_table.setRows(5);
jScrollPane2.setViewportView(damage_table);

claimTable.setColumns(20);
claimTable.setRows(5);
jScrollPane3.setViewportView(claimTable);

buttonGroup1.add(jRadioButton1);
jRadioButton1.setText("Simple case");
```

```
buttonGroup1.add(jRadioButton2);
jRadioButton2.setText("Complex case");
jRadioButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton2ActionPerformed(evt);
    }
});

jButton1.setText("Record Claim");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(jSeparator2)
    .addGroup(layout.createSequentialGroup()

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
        .LEADING)
         .addGroup(layout.createSequentialGroup()
            .addGap(54, 54, 54)
```

```
    .addComponent(jLabel1)

    .addGap(44, 44, 44)

    .addComponent(CustomerID,
javax.swing.GroupLayout.PREFERRED_SIZE,                    83,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addGap(40, 40, 40)

    .addComponent(Search))

 .addGroup(layout.createSequentialGroup()

    .addGap(22, 22, 22)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.TRAILING)

        .addComponent(jLabel4)

        .addComponent(jLabel3)

        .addComponent(jLabel2))

    .addGap(48, 48, 48)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING, false)

        .addComponent(type)

        .addComponent(validity, javax.swing.GroupLayout.DEFAULT_SIZE,
83, Short.MAX_VALUE)

        .addComponent(name))

    .addGap(62, 62, 62)

    .addComponent(jLabel7)

    .addGap(38, 38, 38)

    .addComponent(price,    javax.swing.GroupLayout.PREFERRED_SIZE,
84, javax.swing.GroupLayout.PREFERRED_SIZE))

 .addGroup(layout.createSequentialGroup()

    .addContainerGap()
```

```
            .addComponent(jScrollPane2,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(36, 36, 36)

            .addComponent(jScrollPane3,
        javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)


        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
        .LEADING)

                .addComponent(jRadioButton1)

                .addComponent(jRadioButton2)

                .addComponent(jButton1)))

         .addGroup(layout.createSequentialGroup()

            .addGap(10, 10, 10)

            .addComponent(jLabel5)

            .addGap(126, 126, 126)

            .addComponent(jLabel6)


        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
        D)

            .addComponent(status,    javax.swing.GroupLayout.PREFERRED_SIZE,
        186, javax.swing.GroupLayout.PREFERRED_SIZE)))

      .addContainerGap(114, Short.MAX_VALUE))

);

layout.setVerticalGroup(

   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

   .addGroup(layout.createSequentialGroup()

      .addContainerGap()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)

 .addComponent(CustomerID,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

 .addComponent(jLabel1)

 .addComponent(Search))


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)

.addComponent(jSeparator1,  javax.swing.GroupLayout.PREFERRED_SIZE,
10, javax.swing.GroupLayout.PREFERRED_SIZE)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)

 .addComponent(jLabel2)

 .addComponent(name,       javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

 .addComponent(jLabel7)

 .addComponent(price,       javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)

 .addComponent(type,       javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addComponent(jLabel3,
javax.swing.GroupLayout.Alignment.TRAILING))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.TRAILING)

.addComponent(jLabel4)

.addComponent(validity, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)

.addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE,
10, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D, 23, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASELINE)

.addComponent(jLabel5)

.addComponent(jLabel6)

.addComponent(status, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)

.addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
            .addComponent(jScrollPane3,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGroup(layout.createSequentialGroup()

                .addComponent(jRadioButton1)


            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNREL
            ATED)

                .addComponent(jRadioButton2)


            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNREL
            ATED)

                .addComponent(jButton1)))

        .addGap(29, 29, 29))

    );


    pack();

}// </editor-fold>


private void SearchActionPerformed(java.awt.event.ActionEvent evt) {


        damage_table.setText(null);

        claimTable.setText(null);

        name.setText("");

        type.setText("");

        validity.setText("");

        price.setText("");


    String id= CustomerID.getText();
```

```
if (!"".equals(id)) {
    Vector<String> detail= new Vector<> ();
    try {
        detail.addAll(Customer.getInfo(id));
    } catch (FileNotFoundException ex) {
        Logger.getLogger(DashBoard.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(null, "Customer not exist");
    }
    Enumeration e= detail.elements();
    name.setText(e.nextElement(). toString());
    type.setText(e.nextElement(). toString());
    validity.setText(e.nextElement(). toString());
    price.setText(e.nextElement(). toString());

    damage_table.setText("Date\t Cost\n");
    if (e.hasMoreElements()){
    for (int i=0; i<((detail.size()-4)/2); i++)
    {
    damage_table.append(e.nextElement(). toString()+"\t");
    damage_table.append(e.nextElement(). toString()+"\n");
    }
    }
    else{damage_table.append("There is no History");}
    claimTable.setText(null);
    try {
    Vector<String> claim= new Vector<String> ();
```

```
            claim.addAll(Customer.getClaim(id));
            Enumeration s= claim.elements();
            claimTable.setText("Date\t Cost\t status\n");
            status.setText(null);
               if (s.hasMoreElements())
               {        while (s.hasMoreElements())
               {
            claimTable.append(s.nextElement(). toString()+"\t");
            claimTable.append(s.nextElement(). toString()+"\t");
            claimTable.append(s.nextElement(). toString()+"\n");
               }
//          damage_cost.setText(s.nextElement(). toString());
//          claim_date.setText(s.nextElement(). toString());
               }
               else
               {
                  status.setText("No unhandled claim");
//                  damage_cost.setText(null);
//                  claim_date.setText(null);
                  claimTable.setText(null);

               }
      }

         catch (FileNotFoundException ex) {
            Logger.getLogger(DashBoard.class.getName()).log(Level.SEVERE, null, ex);
            status.setText("No claim record");
```

```
        claimTable.setText(null);
    }


    }
    else{
        JOptionPane.showMessageDialog(null, "Please enter CustomerID");
    }
}

private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(null, "Coming Soon.....");
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {

    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
```

```
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException | InstantiationException | IllegalAccessExcep-
        tion | javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(DashBoard.class.getName()).log(java.ut
        il.logging.Level.SEVERE, null, ex);
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new DashBoard().setVisible(true);
    }
});
}
// Variables declaration - do not modify
private javax.swing.JTextField CustomerID;
private javax.swing.JButton Search;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JTextArea claimTable;
private javax.swing.JTextArea damage_table;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
```

```
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JTextField name;
    private javax.swing.JTextField price;
    private javax.swing.JLabel status;
    private javax.swing.JTextField type;
    private javax.swing.JTextField validity;
    // End of variables declaration
}
```