# IDEA Feature Implementation Report (Team 3)

## 1. Introduction

This report is done by Sun Yin (U017236L) and Man Shujing (U027966J), to provide a formal documentation for the implementation of IDEA security functions, for the convenience of system maintenance.

The functions described here were implemented using Java. For the purpose of discussion, we shall describe the algorithm used in simple English or pseudo code. The report is aimed to provide programmers doing maintenance and future expansion, who may not be involved in the development of this very first version of IDEA, with a clearer understanding of how the security features were implemented.

Following the system's top-down hierarchical architecture, the security functions were developed independently of the functions of other aspects. The top-down hierarchical approach is the recommended standard method and is widely followed in the professional software development world.

## 2. IDEA Requirements Assigned to Team

The team has been assigned to implement the security features of the system.

| No. | Function | Purpose |
|---|---|---|
| 1 | GenerateID.java | To generate a random user outside the door, the User-ID and Status will be shown after generation. |
| 2 | EnterLeave.java | Update the location (inside/outside the room) of the user, upon entering or leaving the door. Update the Enter/Leave button according to the state of the user. |
| 3 | Checkstatus.java | Take in the User-ID of the person approaching the door and determine his status as occupant, friendly visitor or unwanted visitor. |

# 3. IDEA Function Implementation and Testing

3.1 Overview of the Implementation Plan.

The specifications of the functions are first obtained from the SDA team1 and understanding of the requirements ensured after discussion with the system design team. The schema of the database design is obtained from the SDA team 3 for reference. The required input and output of the functions are then determined appropriately according to the formats specified in the database table and we have decided to implement it using Java. The testing and verification is to be done manually by entering different inputs under the Unix interface.

3.2 Function Specifications

## 3.2.1 GenerateID.java

### *3.2.1a. Description*

This function is to simulate the selection of some listed user approaching the door by generating a random UserID outside the door. It is not activated by the user, but called via "screen refresh" of a small frame of the page. The output of the function will be sent to GUI via the controller, the ultimate result will be updated on the screen showing the status of person generated, and activate appropriate screens for occupant, friendly guest or unwanted people.

### *3.2.1b. Input/Output Specification*

No input is required for this function.

The User ID and Status of the simulation generated will be printed to the standard output as one single line with User ID (string) followed by Status (integer) and one space in between.

### *3.2.1c. Data Structures involved.*

No complicated abstract data structures involved.

Data type involved will be in agreement with database linked. User-ID generated is output as string, and friendliness status is sent out as integer 0,1 or 2.

### *3.2.1d. Algorithm.*

1. Establish connection with Database Manager;
2. Select all the records of individuals outside the door from the database by query the database with condition "pers_location = '0' " (0 indicate the person is located outside the door).
3. Count the total number "Total" of selected the records, to prepare for random number generation.
4. Generate a random number in the range of 0-Total, assign it to "Pick". One record is randomly generated.
5. Hash the selected record, and output the UserID and Status of this record.
6. Close all the files.
7. Prompt error message if exception error occurs.

### *3.2.1e Testing*

The program was run several times in Unix and one random user was generated each time the program was executed. The testing was completed when all the users with pers_location 0 have been generated at least once. (test data appended behind).

### *3.2.1f Future Expansion*

This simulation can be expanded to generate records not in the database as strangers. The simulation is just a temporary measure. It has not

included the function of generate a stranger currently is a consensus with the GUI team.

### 3.2.2 EnterLeave.java

#### *3.2.2a. Description*

This function is activated by clicking an Enter/Leave button. It flips the state of the button as well as the location status of the person base on the previous location state. The result viewed by the user will be a change of Enter button to Leave button when a person enters and a change of Leave button to Enter button when after the person has left the room. The pers_location field of the record in database is updated accordingly.

#### *3.2.2b. Input/Output Specification*

User ID of the person at the door will be passed into the function.

No parameter is passed out of this function, but the status of button and relevant record in the database is updated accordingly.

#### *3.2.2c. Data Structures involved.*

No complicated abstract data structures involved.

Data type involved will be in agreement with database linked. According to PEOPLE table, User-ID passed in will be string.

#### *3.2.2d. Algorithm.*

1. Establish connection with Database Manager;
2. Query the database for the pers_location of the user passed into the function with condition “pers_handle = User ID ”
3. Switch the value of pers_location via a statement

loc=1-loc;

Since 0 indicate the person is located outside the door, illegal value is prohibited automatically.

4. Update the value of pers_location in the database.
5. Close all the files
6. Prompt error message in any unexpected case.

#### *3.2.2e Testing*

The function CheckStatus was slightly modified to output the pers_location instead of pers_status for quick check of the result. After the program EnterLeave was executed by entering a pers_handle, the modified CheckStatus was executed to check whether the associated field has been updated (in this case pers_location). (test data appended behind).

#### *3.2.2f Future Expansion*

List of users in house/outside shown.

### 3.2.3 Checkstatus.java

#### *3.2.3a. Description*

This function is activated upon entering User ID (pers_handle). According to the User ID entered the friendliness status of the particular record will be shown, and activate appropriate screens for occupant, friendly guest or unwanted people.

#### *3.2.3b. Input/Output Specification*

pers_handle entered will be passed into the function.

The friendliness status of the person is output as an integer. With

0 for occupant;

1 for friendly;

2 for unwanted.

### *3.2.3c. Data Structures involved.*

No complicated abstract data structures involved.

Data type involved will be in agreement with database linked. According to PEOPLE table, pers_handle passed in will be string, and friendliness status is sent out as integer 0, 1 or 2.

### *3.2.3d. Algorithm.*

1. Establish connection with Database Manager;
2. Query the database, locate the record with the entered pers_handle, select pers_status of the user passed into the function.

   output the friendliness status of the record hashed.

3. Close all the files
4. Prompt error message if exception occurs.

### *3.2.3e Testing*

A query is done to the database manually to print out the table, so that we will have an idea what kind of status to expect for a particular input of pers_handle. The program is then executed by entering different pers_handle from the table to check its pers_status. All tested pers_handle outputs correct pers_status. (test data appended behind).

## 4. Summary and Conclusion

This project is a joint effort by Sun Yin and Man Shujing. All of them are active giving input to this project in the process of development. Sun Yin assumed the leadership role to coordinate with other development teams regarding problems and clarifications of the requirements to make sure smooth progress of the project. Shujing is a diligent worker who completed her assignments with due effort. The development was done by both of the team members concurrently and efficiently. All the functions implemented are the basic necessary functions to ensure the efficient working of the door. This project could be improved further by adding in more fancy features to the existing functions specified above to make it multi-functional.

## Appendix I. Source Code

GenerateID.java 4/13/2003

```
import java.sql.*;
import DBClasses.*;
import java.util.*;
import java.lang.*;

// this program implements security function 9 by
// generating an random user outside the door.
// the status and userid of the person is shown.
// no input is required for this program.

class GenerateID {

  public static void main(String[] args) {

     dbManager dbman = new dbManager();
     Connection conn = dbman.getConnection();

     int    status = 0;
     int    total = 0;
     int    counter = 0;
     int    pick = 0;
     Random generator = new Random();

     try {

       //all the individuals outside the door are extracted from the
       database.
       Statement stmt = conn.createStatement();
       String sql =
       "select pers_handle, pers_status from PEOPLE where pers_location =
       '0'";
       ResultSet rs = stmt.executeQuery(sql);

       while(rs.next()) //calculating the total number of tuples.
            total++;

       pick = Math.abs(generator.nextInt(total)); //randomly determining
       which tuple to be selected

       rs = stmt.executeQuery(sql);

       //selecting the tuple.
       while(rs.next()){
           if(counter==pick){
                String perhandle = new String(rs.getString(1).trim());
                status = rs.getInt(2);
                System.out.println(perhandle+" "+status); //printing out
                the result
                break;
            }
            counter++;
        }

       rs.close();       // close everything (rs, stmt, conn)
       stmt.close();
       conn.close();

    } catch (Exception e) {
      System.out.println("DB Error!"); // error message if exception occurs
      e.printStackTrace();
    }
  }
}
```

EnterLeave.java 4/13/2003

```
import java.sql.*;
import DBClasses.*;
// this program implements security function 5 by
// takeing in the pers handle of a person and
// changes his location accordingly when he enters or leaves.
// args[0] is used to denote the user ID (pers_handle).

class EnterLeave {

  public static void main(String[] args) {

     int loc = 0; //used to store the value of pers_location.

     dbManager dbman = new dbManager();
     Connection conn = dbman.getConnection();

     try {

       Statement stmt = conn.createStatement();
       String sql =
       "select pers location from PEOPLE where pers_handle =
       '"+args[0]+"'";
       ResultSet rs = stmt.executeQuery(sql);

       //getting the old value of pers_location and toggling the old value.
       rs.next();
       loc = Integer.valueOf(rs.getString(1).trim()).intValue();
       loc = 1 - loc;

       //updating the value of pers location in the database.
       Statement updateLocation = conn.createStatement();
       updateLocation.executeUpdate("UPDATE PEOPLE SET pers_location ='"+
       loc+"'where pers handle ='"+args[0]+"'");
       updateLocation.close();

    } catch (Exception e) {
      System.out.println("Not Found!"); // error message if exception
      occurs
    }
  }
}
```

CheckStatus.java 4/13/2003

```
import java.sql.*;
import DBClasses.*;
// this program implements security function 8 by
// takeinq in the pers handle of a person and
// determines his status as occupant, friendly or unwanted.
// args[0] is used as the input user ID (pers_handle).

class CheckStatus {

  public static void main(String[] args) {

     // 0=occupant, 1=friendly, 2=unwanted
     int status = 0;

     dbManager dbman = new dbManager();
     Connection conn = dbman.getConnection();

     try {

       //selecting the status of the wanted ID.
       Statement stmt = conn.createStatement();
       String sql = "select pers_status from PEOPLE where pers_handle = '"+
       args[0]+"'";
       ResultSet rs = stmt.executeQuery(sql);

       rs.next();
       status = Integer.valueOf(rs.getString(1).trim()).intValue();
       System.out.println(status);        // output the result;

       rs.close();       // close everything (rs, stmt, conn)
       stmt.close();
       conn.close();
    } catch (Exception e) {
       System.out.println("Not Found!"); // error message if exception
       occurs
     }
  }
}
```

**Appendix II. Test Data**

**Table PEOPLE**

```
//TABLE: PEOPLE
//------------------------------ -------- ----
// Name                    Null?    Type
//------------------------------ -------- ----
//PERS_ID                  NOT NULL NUMBER primary key
//PERS_HANDLE              NOT NULL CHAR(10) used for logon
//PERS_FIRST_NAME          NOT NULL VARCHAR2(20) used for greeting
//PERS_LAST_NAME                    VARCHAR2(20) used for formal greeting
//PERS_STATUS              NOT NULL NUMBER(1) 0-occupant/1-friendly/2-
unwanted
//PERS_BIRTHDATE                    DATE  used for birthday greeting, see
WELCOME
//PERS_LOCATION            NOT NULL NUMBER(1) 0-outside/1-inside
//------------------------------ -------- ----

// ID|Handle|FirstName|LastName|Status|DOB|Location

1|leonghw|Leong|Hon Wai|0|04-APR-1955|0
2|weichu|Heng|Wei Chu|1|03-DEC-1981|0
3|daniel|Lim|Daniel|1|10-JUL-1980|0
4|frankie|Teah|Huan Ying|1|23-JUN-1980|0
5|benching|Ching|Chung Siang Benjamin|1|23-DEC-1980|0
6|lijia|Li|Jia|0|05-JUN-1981|0
7|jiajie|Liang|Jiajie|1|03-FEB-1981|0
8|sunyin|Sun|Yin|2|03-MAR-1982|0
9|tongchoon|Koh|Tong Choon|2|02-FEB-1980|0
10|junyun|Tay|Junyun|1|27-FEB-1983|0
11|hongee|Low|Hong Ee|1|01-JAN-1981|0
12|chris|Mendis|Chris|2|09-SEP-1935|0
```

13|willie|Koh|Lok Kiang William|2|19-FEB-1944|0

14|santa|Claus|Santa|1|25-DEC-1982|0

15|bingo|Game|Card|2|01-JAN-1966|0

**i.CheckStatus**

eng11956@sf3:~/public_cgi[505]$ javac CheckStatus.java

eng11956@sf3:~/public_cgi[506]$ java CheckStatus leonghw

0

eng11956@sf3:~/public_cgi[507]$ java CheckStatus weichu

1

eng11956@sf3:~/public_cgi[508]$ java CheckStatus daniel

1

eng11956@sf3:~/public_cgi[509]$ java CheckStatus frankie

1

eng11956@sf3:~/public_cgi[510]$ java CheckStatus sunyin

2

eng11956@sf3:~/public_cgi[511]$ java CheckStatus tongchoon

2

eng11956@sf3:~/public_cgi[512]$ java CheckStatus junyun

1

eng11956@sf3:~/public_cgi[513]$ java CheckStatus hongee

1

eng11956@sf3:~/public_cgi[514]$

**ii. GenerateID**

eng11956@sf3:~/public_cgi[514]$ javac GenerateID.java
eng11956@sf3:~/public_cgi[515]$ java GenerateID
daniel 1
eng11956@sf3:~/public_cgi[516]$ java GenerateID
hongee 1
eng11956@sf3:~/public_cgi[517]$ java GenerateID
chris 2
eng11956@sf3:~/public_cgi[518]$ java GenerateID
benching 1
eng11956@sf3:~/public_cgi[519]$ java GenerateID
santa 1
eng11956@sf3:~/public_cgi[520]$ java GenerateID
chris 2
eng11956@sf3:~/public_cgi[521]$ java GenerateID
jiajie 1
eng11956@sf3:~/public_cgi[522]$ java GenerateID
weichu 1
eng11956@sf3:~/public_cgi[523]$ java GenerateID
bingo 2
eng11956@sf3:~/public_cgi[524]$ java GenerateID
sunyin 2
eng11956@sf3:~/public_cgi[525]$

**iii. EnterLeave. (the program CheckLocation was modified from CheckStatus.)**

```
eng11956@sf3:~/public_cgi[534]$ java CheckLocation leonghw
0
eng11956@sf3:~/public_cgi[535]$ java EnterLeave leonghw
eng11956@sf3:~/public_cgi[536]$ java CheckLocation leonghw
1
eng11956@sf3:~/public_cgi[537]$ java EnterLeave leonghw
eng11956@sf3:~/public_cgi[538]$ java CheckLocation leonghw
0
eng11956@sf3:~/public_cgi[539]$
```