****

**INFORMATICS COMPUTER SCHOOL**

Cover Page

**ICT420 –**

Object Oriented Programming in Java

**PROJECT**

**football club  
 Record System**

**Student name: Josh Lim Guorong**

**Student IC: S8424510J**

**Student ID: 018800022064**

**Submission Date: Nov 2016**

**Academic Year: Sept 2016**

**Lecturer: Kiwy Tan**

**Center Code: 0188 (Singapore)**

Contents

[1.0 Analysis & Design 3](#_Toc466923095)

[1.1 Introduction and Program Objective 3](#_Toc466923096)

[1.2 Current System and Existing Problem 3](#_Toc466923097)

[1.3 New System and Program Objective 3](#_Toc466923098)

[2.0 Program Design 4](#_Toc466923099)

[2.1 Flowchart 4](#_Toc466923100)

[2.2 Output Screen Design 5](#_Toc466923101)

[2.2.1 Login Page 5](#_Toc466923102)

[2.2.2 Menu Page 6](#_Toc466923103)

[2.2.3 Add New Player 7](#_Toc466923104)

[2.2.4 Display Player Records 8](#_Toc466923105)

[2.2.5 Search Player Records 9](#_Toc466923106)

[2.2.6 Update Player Record 10](#_Toc466923107)

[2.2.7 Delete Player Record 11](#_Toc466923108)

[3.0 Development & Testing 12](#_Toc466923109)

[3.1 Program Listing 12](#_Toc466923110)

[3.2 Program Testing 32](#_Toc466923111)

[3.2.1 Test Plan 32](#_Toc466923112)

[3.2.2 Test Cases and Result 33](#_Toc466923113)

[4.0 Implementation 54](#_Toc466923114)

[4.1 Operating Instruction 54](#_Toc466923115)

[4.1.1 System Requirements 54](#_Toc466923116)

[4.1.2 User Manual 55](#_Toc466923117)

[5.0 Conclusion 61](#_Toc466923118)

[5.1 Program Strength 61](#_Toc466923119)

[5.2 Program Weakness 61](#_Toc466923120)

[5.3 Program Enhancement 61](#_Toc466923121)

# 1.0 Analysis & Design

## 1.1 Introduction and Program Objective

FC Metz is a sporting club with 60 years of history. Players have retired and new players have join in almost every passing year. The club has no procedures in storing old and new players data and can only recognize these players with the official club issued ID card. This means that there are no online facilities for club players which is a shame in this modern era.

## 1.2 Current System and Existing Problem

Due to the club new promotion into non-league, FC Metz is unable to cope with all the increased attention and paper works. The call is urgent to establish a shared database platform that is simple to use, and is effective.

The club will retrench some of the staff and train new ones with this database system. Current work approach will be retired and external consultants will be hired to train club employees in using this new system.

During the transitional period, old system will be kept as backup and alternative until the new system is live and in production.

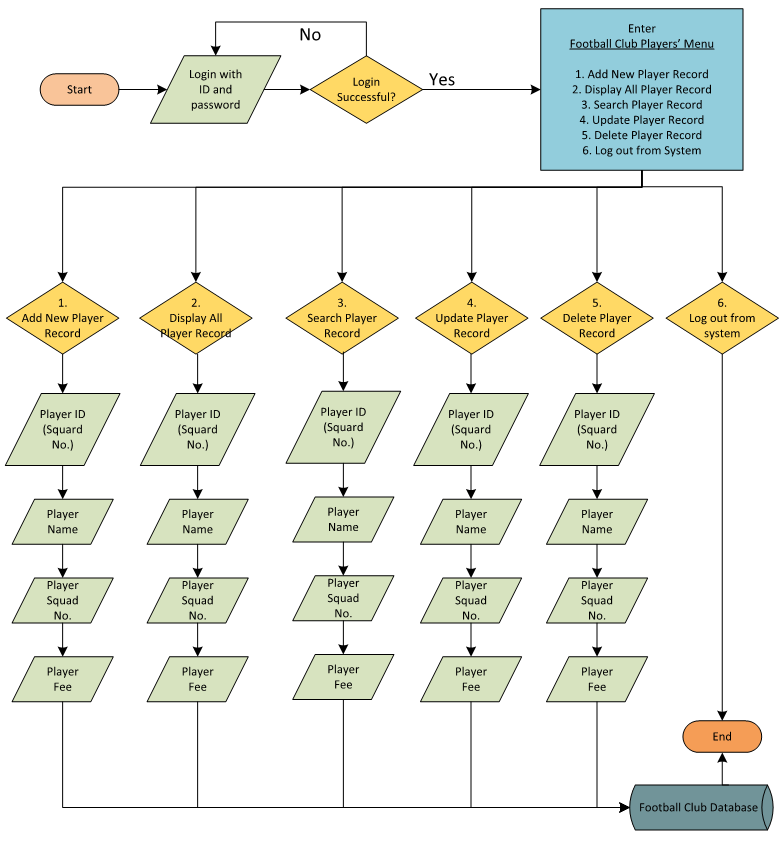
## 1.3 New System and Program Objective

President of FC Metz have ordered for an online database to be implemented for existing as well as the alumni. And here is the direction:

The system got to be easy to use and easy to maintain. It must also be tested and deflects free, so as not to jeopardise daily operations. It must be able to record up to maximum 30 characters. The layout must easy to read. It must have a simple search function.

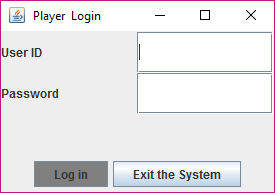
# 2.0 Program Design

## 2.1 Flowchart

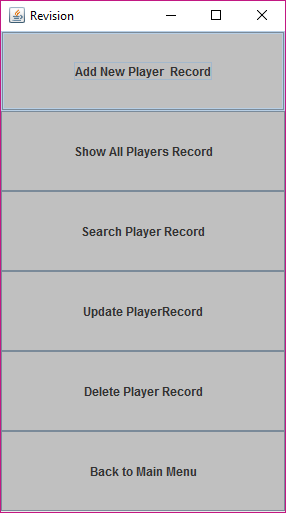


## 2.2 Output Screen Design

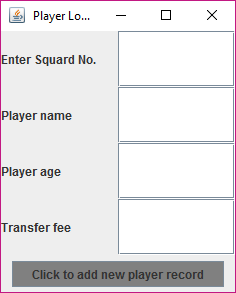
### 2.2.1 Login Page



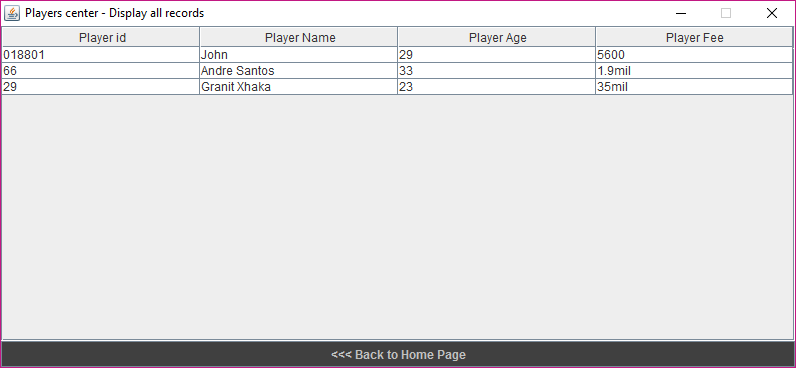
### 2.2.2 Menu Page



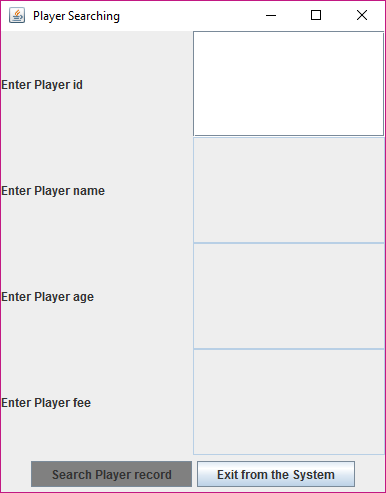
### 2.2.3 Add New Player



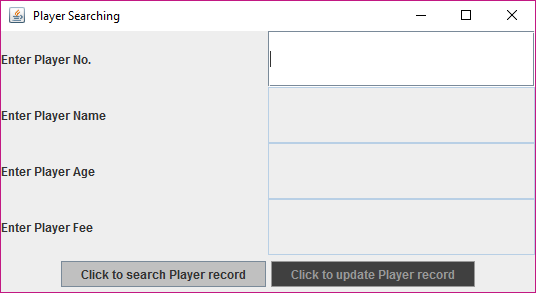
### 2.2.4 Display Player Records



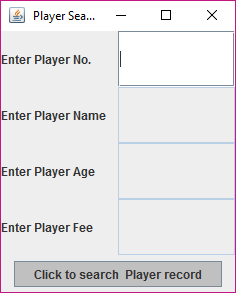
### 2.2.5 Search Player Records



### 2.2.6 Update Player Record



### 2.2.7 Delete Player Record



# 3.0 Development & Testing

## 3.1 Program Listing

* Playerlogin

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playermenu extends JFrame

implements ActionListener

{

JButton a;

JButton b;

JButton c;

JButton d;

JButton e;

JButton f;

public void actionPerformed(ActionEvent ee)

{

if( ee.getSource() == a)

{

playeradd asr = new playeradd();

this.dispose();

};

if( ee.getSource() == b)

{

playerecords asr = new playerecords();

this.dispose();

};

if( ee.getSource() == c)

{

playersearch asr = new playersearch();

this.dispose();

};

if( ee.getSource() == d)

{

playerupdate asr = new playerupdate();

this.dispose();

};

if( ee.getSource() == e)

{

playerdelete asr = new playerdelete();

this.dispose();

};

if( ee.getSource() == f)

{

playerlogin hay = new playerlogin();

this.dispose();

};

}

//create constructor

public playermenu()

{

setLayout(new GridLayout(6,1));

a =new JButton("Add New Player Record");

add(a);

a.addActionListener(this);

a.setBackground(Color.lightGray);

b =new JButton("Show All Players Record");

add(b);

b.addActionListener(this);

b.setBackground(Color.lightGray);

c =new JButton("Search Player Record");

add(c);

c.addActionListener(this);

c.setBackground(Color.lightGray);

d =new JButton("Update PlayerRecord");

add(d);

d.addActionListener(this);

d.setBackground(Color.lightGray);

e =new JButton("Delete Player Record");

add(e);

e.addActionListener(this);

e.setBackground(Color.lightGray);

f =new JButton("Back to Main Menu");

add(f);

f.addActionListener(this);

f.setBackground(Color.lightGray);

setTitle("Revision");

setSize(300,520);

setVisible(true);

}

public static void main(String arg[])

{

playermenu arsenalmenu= new playermenu();

}

}

* Playermenu

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playermenu extends JFrame

implements ActionListener

{

JButton a;

JButton b;

JButton c;

JButton d;

JButton e;

JButton f;

public void actionPerformed(ActionEvent ee)

{

if( ee.getSource() == a)

{

playeradd asr = new playeradd();

this.dispose();

};

if( ee.getSource() == b)

{

playerecords asr = new playerecords();

this.dispose();

};

if( ee.getSource() == c)

{

playersearch asr = new playersearch();

this.dispose();

};

if( ee.getSource() == d)

{

playerupdate asr = new playerupdate();

this.dispose();

};

if( ee.getSource() == e)

{

playerdelete asr = new playerdelete();

this.dispose();

};

if( ee.getSource() == f)

{

playerlogin hay = new playerlogin();

this.dispose();

};

}

//create constructor

public playermenu()

{

setLayout(new GridLayout(6,1));

a =new JButton("Add New Player Record");

add(a);

a.addActionListener(this);

a.setBackground(Color.lightGray);

b =new JButton("Show All Players Record");

add(b);

b.addActionListener(this);

b.setBackground(Color.lightGray);

c =new JButton("Search Player Record");

add(c);

c.addActionListener(this);

c.setBackground(Color.lightGray);

d =new JButton("Update PlayerRecord");

add(d);

d.addActionListener(this);

d.setBackground(Color.lightGray);

e =new JButton("Delete Player Record");

add(e);

e.addActionListener(this);

e.setBackground(Color.lightGray);

f =new JButton("Logout from this account");

add(f);

f.addActionListener(this);

f.setBackground(Color.lightGray);

setTitle("Revision");

setSize(300,520);

setVisible(true);

}

public static void main(String arg[])

{

playermenu arsenalmenu= new playermenu();

}

}

* Playerecords

import java.io.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playerecords extends JFrame

implements ActionListener

{

JButton back;

public playerecords()

{

super("Players center - Display all records");

System.out.println("1 set up is complete ");

try

{

String column[]=

{"Player id","Player Name","Player Age","Player Fee"};

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

String data[][];

int size=information.readInt();

data = new String[size][4];

for (int x=0; x<size;x++)

{

data[x][0]= information.readUTF();

data[x][1]= information.readUTF();

data[x][2]= information.readUTF();

data[x][3]= information.readUTF();

};

JTable jt = new JTable(data,column);

JScrollPane jsp = new JScrollPane(jt);

getContentPane().add(jsp);

setSize(800, 370);

setVisible(true);

setResizable(false);

}

catch (Exception e) {};

back = new JButton("<<< Back to Home Page");

back.setBackground(Color.darkGray);

back.setForeground(Color.lightGray);

back.addActionListener(this);

getContentPane().add(back, "South");

}

public void actionPerformed(ActionEvent e)

{

if (e.getSource() == back)

{

};

}

public static void main(String args[])

{

playerecords f = new playerecords();

}

}

* Playersearch

import javax.swing.\*;

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playersearch extends JFrame

implements ActionListener

{

JPanel pc;

JPanel ps;

JButton searchplayer;

JButton exit;

JLabel id\_label;

JLabel name\_label;

JLabel age\_label;

JLabel pay\_label;

JTextField id;

JTextField name;

JTextField age;

JTextField pay;

// declare array

static String s\_id[] = new String[2000];

static String s\_name[] = new String[2000];

static String s\_age[] = new String[2000];

static String s\_payment[] = new String[2000];

static int size = 0;

public static void save\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

information.writeInt(size);

for(int x = 0; x < size; x++)

{

information.writeUTF(s\_id[x]);

information.writeUTF(s\_name[x]);

information.writeUTF(s\_age[x]);

information.writeUTF(s\_payment[x]);

}

}

public static void retrieve\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

size = information.readInt();

for(int x = 0; x < size; x++)

{

s\_id[x] = information.readUTF();

s\_name[x] = information.readUTF();

s\_age[x] = information.readUTF();

s\_payment[x] = information.readUTF();

}; }

public void actionPerformed(ActionEvent e)

{

if( e.getSource() == searchplayer)

{

//

boolean found =false;

for(int x=0; x < size; x++)

if(s\_id[x].equals(id.getText()))

{

JOptionPane.showMessageDialog(null, "Player Record Found");

name.setText (s\_name[x] );

age.setText (s\_age[x] );

pay.setText (s\_payment[x]);

found = true;

}

if( found == false)

JOptionPane.showMessageDialog(null, "No such Player Record Found");

}

else

if(e.getSource() == exit)

{

try

{

save\_data();

} catch (Exception f) {};

JOptionPane.showMessageDialog(null, "Thank you");

System.exit(0);

}

}

public playersearch()

{

try

{

retrieve\_data();

}catch (Exception e) {};

pc = new JPanel();

pc.setLayout(new GridLayout(4,4));

id\_label = new JLabel( "Enter Player id ");

pc.add(id\_label);

id = new JTextField(41);

pc.add(id);

name\_label = new JLabel( "Enter Player name ");

pc.add(name\_label);

name = new JTextField(41);

pc.add(name);

name.setEditable(false);

age\_label = new JLabel( "Enter Player age ");

pc.add(age\_label);

age = new JTextField(41);

pc.add(age);

age.setEditable(false);

pay\_label = new JLabel( "Enter Player fee ");

pc.add(pay\_label);

pay = new JTextField(41);

pc.add(pay);

pay.setEditable(false);

add(pc);

ps = new JPanel();

ps.setLayout(new FlowLayout());

searchplayer = new JButton(" Search Player record ");

searchplayer.addActionListener(this);

ps.add(searchplayer);

searchplayer.setBackground(Color.GRAY);

exit = new JButton(" Exit from the System ");

exit.addActionListener(this);

ps.add(exit);

add(ps,"South");

setTitle(" Player Searching ");

setLocation(300,200);

setSize(380,480);

setVisible(true);

}

public static void main(String args[])

{

playersearch hay = new playersearch();

}

}

* Playeradd

import javax.swing.\*;

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playeradd extends JFrame

implements ActionListener

{

JPanel pc;

JPanel ps;

//

JButton addstud;

//

JButton exit;

JLabel id\_label;

JLabel name\_label;

JLabel age\_label;

JLabel pay\_label;

JTextField id;

JTextField name;

JTextField age;

JTextField pay;

// declare array

static String s\_id[] = new String[100];

static String s\_name[] = new String[100];

static String s\_age[] = new String[100];

static String s\_payment[] = new String[100];

static int size = 0;

public static void save\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

information.writeInt(size);

for(int x = 0; x < size; x++)

{

information.writeUTF(s\_id[x]);

information.writeUTF(s\_name[x]);

information.writeUTF(s\_age[x]);

information.writeUTF(s\_payment[x]);

}

}

public static void retrieve\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

size = information.readInt();

for(int x = 0; x < size; x++)

{

s\_id[x] = information.readUTF();

s\_name[x] = information.readUTF();

s\_age[x] = information.readUTF();

s\_payment[x] = information.readUTF();

}; }

public void actionPerformed(ActionEvent e)

{

if( e.getSource() == addstud)

{

//

JOptionPane.showMessageDialog(null, "Player Record Saved");

s\_id[size] = id.getText ();

s\_name[size] = name.getText ();

s\_age[size] = age.getText ();

s\_payment[size] = pay.getText ();

size ++;

}

else

if(e.getSource() == exit)

{

try

{

save\_data();

} catch (Exception f) {};

JOptionPane.showMessageDialog(null, "Thank you");

System.exit(0);

}

}

// create constructor

public playeradd()

{

try

{

retrieve\_data();

}catch (Exception e) {};

pc = new JPanel();

pc.setLayout(new GridLayout(4,2));

id\_label = new JLabel( "Enter Squard No. ");

pc.add(id\_label);

id = new JTextField(41);

pc.add(id);

name\_label = new JLabel( "Player name ");

pc.add(name\_label);

name = new JTextField(41);

pc.add(name);

age\_label = new JLabel( "Player age ");

pc.add(age\_label);

age = new JTextField(41);

pc.add(age);

pay\_label = new JLabel( "Transfer fee ");

pc.add(pay\_label);

pay = new JTextField(41);

pc.add(pay);

add(pc);

ps = new JPanel();

ps.setLayout(new FlowLayout());

addstud = new JButton(" Click to add new player record ");

addstud.addActionListener(this);

ps.add(addstud);

addstud.setBackground(Color.GRAY);

exit = new JButton(" Exit from the System ");

exit.addActionListener(this);

ps.add(exit);

add(ps,"South");

setTitle(" Player Login ");

//setResizable(false);

setLocation(350,350);

setSize(250,300);

setVisible(true);

}

public static void main(String args[])

{

playeradd hay = new playeradd();

}

}

* Playerdelete

import javax.swing.\*;

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playerdelete extends JFrame

implements ActionListener

{

JPanel pc;

JPanel ps;

//

JButton searchplayer;

JButton deleteplayer;

//

JButton exit;

JLabel id\_label;

JLabel name\_label;

JLabel age\_label;

JLabel pay\_label;

JTextField id;

JTextField name;

JTextField age;

JTextField pay;

// declare array

static String s\_id[] = new String[2000];

static String s\_name[] = new String[2000];

static String s\_age[] = new String[2000];

static String s\_payment[] = new String[2000];

static int size = 0;

static int currentx=0;

public static void save\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

information.writeInt(size);

for(int x = 0; x < size; x++)

{

information.writeUTF(s\_id[x]);

information.writeUTF(s\_name[x]);

information.writeUTF(s\_age[x]);

information.writeUTF(s\_payment[x]);

}

}

public static void retrieve\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

size = information.readInt();

for(int x = 0; x < size; x++)

{

s\_id[x] = information.readUTF();

s\_name[x] = information.readUTF();

s\_age[x] = information.readUTF();

s\_payment[x] = information.readUTF();

}; }

//

public void actionPerformed(ActionEvent e)

{

if( e.getSource() == deleteplayer)

{

//

JOptionPane.showMessageDialog(null, "Player Record Deleted");

for( int x= currentx ; x < size-1; x++)

{

s\_id[x] = s\_id[x+1];

s\_name[x] = s\_name[x+1];

s\_age[x] = s\_age[x+1];

s\_payment[x] = s\_payment[x+1];

}

size--;

deleteplayer.setEnabled(false);

}

else

if( e.getSource() == searchplayer)

{

//

boolean found =false;

for(int x=0; x < size; x++)

if(s\_id[x].equals(id.getText()))

{

JOptionPane.showMessageDialog(null, "Player Record Found");

name.setText (s\_name[x] );

age.setText (s\_age[x] );

pay.setText (s\_payment[x]);

found = true;

currentx =x;

deleteplayer.setEnabled(true);

}

if( found == false)

JOptionPane.showMessageDialog(null, "No such Player Record Found");

}

else

if(e.getSource() == exit)

{

try

{

save\_data();

} catch (Exception f) {};

JOptionPane.showMessageDialog(null, "Thank you");

System.exit(0);

}

}

// create constructor

public playerdelete()

{

try

{

retrieve\_data();

}catch (Exception e) {};

pc = new JPanel();

pc.setLayout(new GridLayout(4,2));

id\_label = new JLabel( "Enter Player No. ");

pc.add(id\_label);

id = new JTextField(30);

pc.add(id);

name\_label = new JLabel( "Enter Player Name ");

pc.add(name\_label);

name = new JTextField(30);

pc.add(name);

name.setEditable(false);

age\_label = new JLabel( "Enter Player Age ");

pc.add(age\_label);

age = new JTextField(30);

pc.add(age);

age.setEditable(false);

pay\_label = new JLabel( "Enter Player Fee ");

pc.add(pay\_label);

pay = new JTextField(30);

pc.add(pay);

pay.setEditable(false);

add(pc);

ps = new JPanel();

ps.setLayout(new FlowLayout());

searchplayer = new JButton(" Click to search Player record ");

searchplayer.addActionListener(this);

ps.add(searchplayer);

searchplayer.setBackground(Color.lightGray);

deleteplayer = new JButton(" Click to delete Player record ");

deleteplayer.addActionListener(this);

ps.add(deleteplayer);

deleteplayer.setBackground(Color.darkGray);

deleteplayer.setEnabled(false);

exit = new JButton(" Exit from the System ");

exit.addActionListener(this);

ps.add(exit);

add(ps,"South");

setTitle(" Player Searching ");

//setResizable(false);

setLocation(350,350);

setSize(250,300);

setVisible(true);

}

public static void main(String args[])

{

playerdelete hay = new playerdelete();

}

}

* Playerupdate

import javax.swing.\*;

import java.io.\*;

import java.awt.\*;

import java.awt.event.\*;

public class playerupdate extends JFrame

implements ActionListener

{

JPanel pc;

JPanel ps;

JButton searchplayer;

JButton updateplayer;

JButton exit;

JLabel id\_label;

JLabel name\_label;

JLabel age\_label;

JLabel pay\_label;

JTextField id;

JTextField name;

JTextField age;

JTextField pay;

// declare array

static String s\_id[] = new String[2000];

static String s\_name[] = new String[2000];

static String s\_age[] = new String[2000];

static String s\_payment[] = new String[2000];

static int size = 0;

static int currentx=0;

public static void save\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

information.writeInt(size);

for(int x = 0; x < size; x++)

{

information.writeUTF(s\_id[x]);

information.writeUTF(s\_name[x]);

information.writeUTF(s\_age[x]);

information.writeUTF(s\_payment[x]);

}

}

public static void retrieve\_data() throws Exception

{

RandomAccessFile information = new RandomAccessFile("abc.dat","rw");

size = information.readInt();

for(int x = 0; x < size; x++)

{

s\_id[x] = information.readUTF();

s\_name[x] = information.readUTF();

s\_age[x] = information.readUTF();

s\_payment[x] = information.readUTF();

}; }

public void actionPerformed(ActionEvent e)

{

if( e.getSource() == updateplayer)

{

//

JOptionPane.showMessageDialog(null, "Player Record Updated");

s\_name[currentx]= name.getText ( );

s\_age[currentx]=age.getText ( );

s\_payment[currentx]= pay.getText ();

updateplayer.setEnabled(false);

name.setEditable(false);

age.setEditable(false);

pay.setEditable(false);

}

else

if( e.getSource() == searchplayer)

{

//

boolean found =false;

for(int x=0; x < size; x++)

if(s\_id[x].equals(id.getText()))

{

JOptionPane.showMessageDialog(null, "Player Record Found");

name.setText (s\_name[x] );

age.setText (s\_age[x] );

pay.setText (s\_payment[x]);

found = true;

currentx =x;

updateplayer.setEnabled(true);

name.setEditable(true);

age.setEditable(true);

pay.setEditable(true);

}

if( found == false)

JOptionPane.showMessageDialog(null, "No such Player Record Found");

}

else

if(e.getSource() == exit)

{

try

{

save\_data();

} catch (Exception f) {};

JOptionPane.showMessageDialog(null, "Thank you");

System.exit(0);

}

}

public playerupdate()

{

try

{

retrieve\_data();

}catch (Exception e) {};

pc = new JPanel();

pc.setLayout(new GridLayout(4,2));

id\_label = new JLabel( "Enter Player No. ");

pc.add(id\_label);

id = new JTextField(30);

pc.add(id);

name\_label = new JLabel( "Enter Player Name ");

pc.add(name\_label);

name = new JTextField(30);

pc.add(name);

name.setEditable(false);

age\_label = new JLabel( "Enter Player Age ");

pc.add(age\_label);

age = new JTextField(30);

pc.add(age);

age.setEditable(false);

pay\_label = new JLabel( "Enter Player Fee ");

pc.add(pay\_label);

pay = new JTextField(30);

pc.add(pay);

pay.setEditable(false);

add(pc);

ps = new JPanel();

ps.setLayout(new FlowLayout());

searchplayer = new JButton(" Click to search Player record ");

searchplayer.addActionListener(this);

ps.add(searchplayer);

searchplayer.setBackground(Color.lightGray);

updateplayer = new JButton(" Click to update Player record ");

updateplayer.addActionListener(this);

ps.add(updateplayer);

updateplayer.setBackground(Color.darkGray);

updateplayer.setEnabled(false);

exit = new JButton(" Exit from the System ");

exit.addActionListener(this);

ps.add(exit);

add(ps,"South");

setTitle(" Player Searching ");

//setResizable(false);

setLocation(350,550);

setSize(550,300);

setVisible(true);

}

public static void main(String args[])

{

playerupdate hay = new playerupdate();

}

}

## 3.2 Program Testing

### 3.2.1 Test Plan

|  |  |  |
| --- | --- | --- |
| **TEST**  **CASE** | TEST DESCRIPTIONS | **TEST RESULT** |
| 1 | Test can user log in with incorrect id | Pass |
| 2 | Test can user log in with incorrect password | Pass |
| 3 | Test can user log in with correct id and password | Pass |
| 4 | Test if user can access “playerlogin” right from command prompt | Pass |
| 5 | Test if user can access “playermenu” right from command prompt | Pass |
| 6 | Test if user can access “playerupdate” right from command prompt | Pass |
| 7 | Test if user can access “playerecords” right from command prompt | Pass |
| 8 | Test if user can access “playerdelete” right from command prompt | Pass |
| 9 | Test if user can access “playeradd” right from “playermenu” | Pass |
| 10 | Test if user can access “playerecords” right from “playermenu” | Pass |
| 11 | Test if user can access “playersearch” right from “playermenu” | Pass |
| 12 | Test if user can access “playeradd” right from “playermenu” | Pass |
| 13 | Test if user can access “playerdelete” right from “playermenu” | Pass |
| 14 | Test if user is able search for non-existent player with the search module | Pass |
| 15 | Test if user is able search existing player record using the search module | Pass |
| 16 | Test if user is able to add a record using the “Add new Playere record module.” | Pass |
| 17 | Test if the newly added record is reflected into database as an added record | Pass |
| 18 | Test if user is able to delete a record in database | Pass |
| 19 | Test if deleted player is erased off from player’s database | Pass |
| 20 | Test can user update existing player record | Pass |
| 21 | Test if system can handle more than 30 characters input | Pass |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### 3.2.2 Test Cases and Result

|  |  |
| --- | --- |
| **TC #01** | TEST DESCRIPTIONS |
| Objective | Test can user log in with incorrect id |
| Test Data | 123456, password |
| Expected Test Result | Program returns with “Invalid login” window |
| Actual Test Result | Program returns with “Invalid login” window |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #02** | TEST DESCRIPTIONS |
| Objective | Test can user log in with incorrect password |
| Test Data | player1, 1 |
| Expected Test Result | Program returns with “Invalid login” window |
| Actual Test Result | Program returns with “Invalid login” window |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #03** | TEST DESCRIPTIONS |
| Objective | Test can user log in with correct id and password |
| Test Data | player1, password |
| Expected Test Result | Program returns with “Thank you” message |
| Actual Test Result | Program returns with “Thank you” message |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #04** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playerlogin” right from command prompt |
| Test Data |  |
| Expected Test Result | System enters into “playerlogin” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #05** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playermenu” right from command prompt |
| Test Data |  |
| Expected Test Result | System enters into “playermenu” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #06** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playerupdate” right from command prompt |
| Test Data |  |
| Expected Test Result | System enters into “playerupdate” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #07** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playerecords” right from command prompt |
| Test Data |  |
| Expected Test Result | System enters into “playerecords” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #08** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playerdelete” right from command prompt |
| Test Data |  |
| Expected Test Result | System enters into “playerdelete” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #09** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playeradd” right from “playermenu” |
| Test Data |  |
| Expected Test Result | System enters into “playeradd” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #10** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playerecords” right from “playermenu” |
| Test Data |  |
| Expected Test Result | System enters into “playerecords” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #11** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playersearch” right from “playermenu” |
| Test Data |  |
| Expected Test Result | System enters into “playersearch” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #12** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playeradd” right from “playermenu” |
| Test Data |  |
| Expected Test Result | System enters into “playeradd” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #13** | TEST DESCRIPTIONS |
| Objective | Test if user can access “playerdelete” right from “playermenu” |
| Test Data |  |
| Expected Test Result | System enters into “playerdelete” module |
| Actual Test Result | The program reacted expectedly. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #14** | TEST DESCRIPTIONS |
| Objective | Test if user is able search for non-existent player with the search module |
| Test Data | 1 |
| Expected Test Result | System will return error message, “No such Player Record Found.” |
| Actual Test Result | System return error message, “No such Player Record Found.” |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |
|  |  |

|  |  |
| --- | --- |
| **TC #15** | TEST DESCRIPTIONS |
| Objective | Test if user is able search existing player record using the search module |
| Test Data | 29 |
| Expected Test Result | System return, “Player Record Found.” |
| Actual Test Result | System return, “Player Record Found.” |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #16** | TEST DESCRIPTIONS |
| Objective | Test if user is able to add a record using the “*Add new Player record module*.” |
| Test Data | 45,45,45,45 |
| Expected Test Result | System return message, “Player Record Saved” |
| Actual Test Result | System return message, “Player Record Saved” |  |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #17** | TEST DESCRIPTIONS |
| Objective | Test if the newly added record is reflected into database as an added record |
| Test Data | 45, 45, 45, 45 |
| Expected Test Result | 45, 45, 45, 45 |
| Actual Test Result | 45, 45, 45, 45 |  |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #18** | TEST DESCRIPTIONS |
| Objective | Test if user is able to delete a record in database |
| Test Data | 45, 45, 45, 45 |
| Expected Test Result | System return message, “Player Record Deleted” |
| Actual Test Result | System return message, “Player Record Deleted” |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #19** | TEST DESCRIPTIONS |
| Objective | Test if deleted player is erased off from player’s database |
| Test Data | - |
| Expected Test Result | Record 45,45,45,45 removed |
| Actual Test Result | The program reacted expectedly with correct error message and prompting action. |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #20** | TEST DESCRIPTIONS |
| Objective | Test can user update existing player record |
| Test Data | “A2” change to “0” in Player ID. |
| Expected Test Result | System return message, “Player Record Updated” |
| Actual Test Result | System return message, “Player Record Updated” |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

|  |  |
| --- | --- |
| **TC #21** | TEST DESCRIPTIONS |
| Objective | Test if system can handle more than 30 characters input |
| Test Data | RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR |
| Expected Test Result | Record added |
| Actual Test Result | Record added |
| Screen Capture |  |
| Conclusion | The test is a success with expected result and outcome. |

# Implementation

## 4.1 Operating Instruction

### 4.1.1 System Requirements

SOFTWARE REQUIREMENTS

* Windows 98, 2000, XP, Windows Vista, 7, 8 or 8.1 OS
* Must have DOS Command Prompt

HARDWARE REQUIREMENTS

* Recommended 1GB RAM
* 30GB Hard Disk
* Keyboard
* Mouse
* Monitor (any brand)

### 4.1.2 User Manual

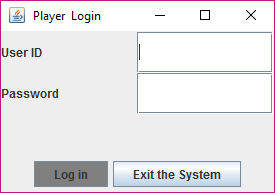
STEP 1

Open CMD, and enter the following



Enter ID as ‘123’

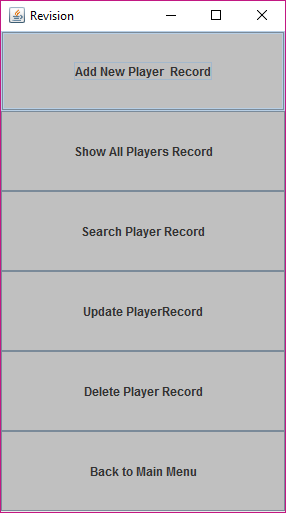
Enter password as ‘abc’ to enter menu



Lastly click Log in to proceed with system loggin.

STEP 2

Once logged in, you will see menu



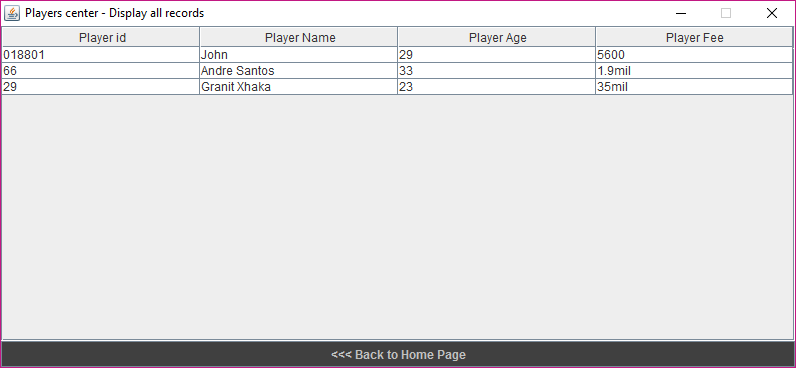
Click on Add New Player Record to add record

After which you must enter Player Age, Fees, Name and Player number to proceed.

STEP 3

Back to the menu

Click Show All Player Record to show all record.

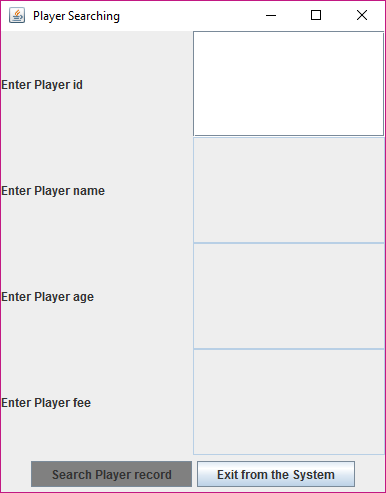


Click back to Home page to hit back at Menu again

STEP 4

Click Search Player Records to search a player

Enter a valid Player Id to search in player’s database.

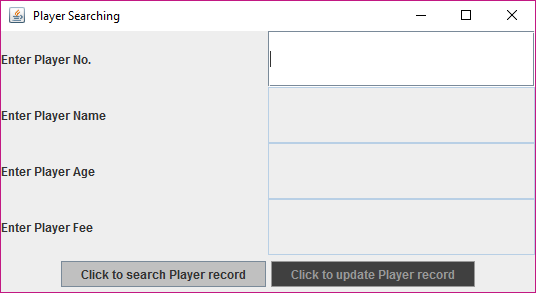


STEP 5

Click Exit from the System to exit.

Type  in CMD to enter directly into Player update module.

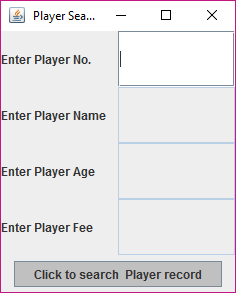
Enter a valid player Id to first search and before you can edit.



STEP 6

To reach Delete module, click Delete Player Record from Menu.

Enter a valid Player Id to delete a player off record.



# Conclusion

This program works as intended. It is not fancy and simple enough to understand. Provided no one taper the codes, this program will work fine.

## 5.1 Program Strength

Program is easy to use and is responsive. It is easy to train people to use it since the program is not complicated. Access can be limited and database can hold many records.

## 5.2 Program Weakness

This program has no visible weakness. However if anyone mess up the code, it can mess up the whole structure and effect continued usage.

## 5.3 Program Enhancement

Program is too simple and should have more modules.   
Program should be loaded with more records so as to simulate actual database situation.

APPENDIX I **Amended Project Proposal**

**INFORMATICS COMPUTER SCHOOL**

**IT209 – Java Programming Project Proposal**

**Employee Records Database System –**

**J2**

**Student Name: Josh Lim Guorong**

**Student Number: 018800022064**

**Contact Number: 93254531**

**Class Code: L4-02**

**Subject: IT201 Java Programming**

**Lecturer Name: Kiwy Tan**

**Center Code: 0188**

**Country: Singapore**

**Date of Submission: 11 June 2015**

INTRODUCTION OF COMPANY

ERDs is a 30 year old transportation company with 30 over employees in Singapore, and 300 over in South East Asia. Its primary success is based on security where a strict employee only access is allowed around its premises. Failure of which means ERDs could risk stolen goods, goods being opened and deem as deflective, damaging its business reputation, and loss of revenue. Hence ERDs has been very particular with its ‘Employee only’ access policy.

CURRENT SYSTEM AND EXISTING PROBLEMS

Recently due to rapid growth and demands in hiring. ERDs is forced to bypass its current manual input filing system and upgrade to a computerised employee records system codenamed “J2”. Current system is just too inefficient in practise and ineffective in its implementation. Waiting time is too long for employee verifications, there are times the security member just cannot find certain records. Prompting ERDs Directors to think of an in house employee records system for ERDs as the next logical upgrade to cater its ever growing list of employees.

It is time consuming to dig into files and find these paper records. Some paper record are written very long ago and even the ink is faded. Some records are gone since the last relocation back in 1998. Some records is destroyed during the great floods of 1992. And ERDs is keen to digitalise its records department so as to avoid these incidents happening again.

OBJECTIVES

This system has got to be able to keep up to 12 kinds of records. (1) Namely Name, (2) Gender, (3) Employee NRIC Number, (4) Employee Date of Birth, (5) Employee ID, (6) Job Title, (7) Contact Number, (8) Employment Status, (9) Education Level, (10) Work Department, (11) Contract End Date and (12) Line Manager. With a varied quantitative records, ERDs wants records of its current workforce as well as retains records of any leaving employees for security and business confidentially reasons.

The system got to be easy to use and easy to maintain. It must also be tested and deflects free, so as not to jeopardise in ERDs daily operations. It must be able to record up to maximum 30 characters and minimum 4 characters. The layout must also be readable. With a simple run search basing at ERDs Employee ID number to provide quick information like Contract End Date, Employment Status, and Line Manager Information, so as our gate keeping security forces can make informed decision. Each Employee ID number should be grouped according to job superiority. For example, numbers beginning with 99 are Company Directors, 88 are Managers, 77 are Senior Officials and 55 are Junior Officials. So as to better identify them at any point of entrances.

The ERDs Employee Records System must also be nimble enough to cater for any realistic edition by an authorised senior security member.

RECORD DATABASE

Main Menu

**Choice 1. Add New Employee record**

**Choice 2. Display Employee report**

**Choice 3. Search Employee detail**

**Choice 4. Update Employee detail**

**Choice 5. Delete Employment detail**

**Choice 6. Exit the system**

**Choice 1.**

**Add New Employee**

1. Name
2. Gender
3. NRIC Number
4. Date of Birth
5. Employee ID
6. Job Title
7. Contact Number
8. Employment Status
9. Education Level
10. Work Department
11. Contract End Date
12. Line Manager

**Choice 2.**

**Display Employee Record**

(Displays all employee details)

1. Employee Name
2. Employee ID
3. Job Title
4. Contact Number
5. Employment Status
6. Education Level
7. Work Department
8. Contract End Date
9. Line Manager

**Choice 3.**

**Search Employee Record**

(Search is based on employee id)

1. Employee Name
2. Employee ID
3. Job Title
4. Employment Status
5. Work Department
6. Contract End Date
7. Line Manager

**Choice 4.**

**Edit/Extend/Update Employment Record**

(Edit based on employee id)

1. Employee Name
2. Employee ID
3. Job Title
4. Employment Status
5. Education Level
6. Work Department
7. Contract End Date
8. Line Manager
9. Enter new job title
10. Enter new contact number
11. Enter new employment status
12. Enter new education level
13. Enter new work department
14. Enter new contract end date
15. Enter new line manager

**Choice ~~5~~.**

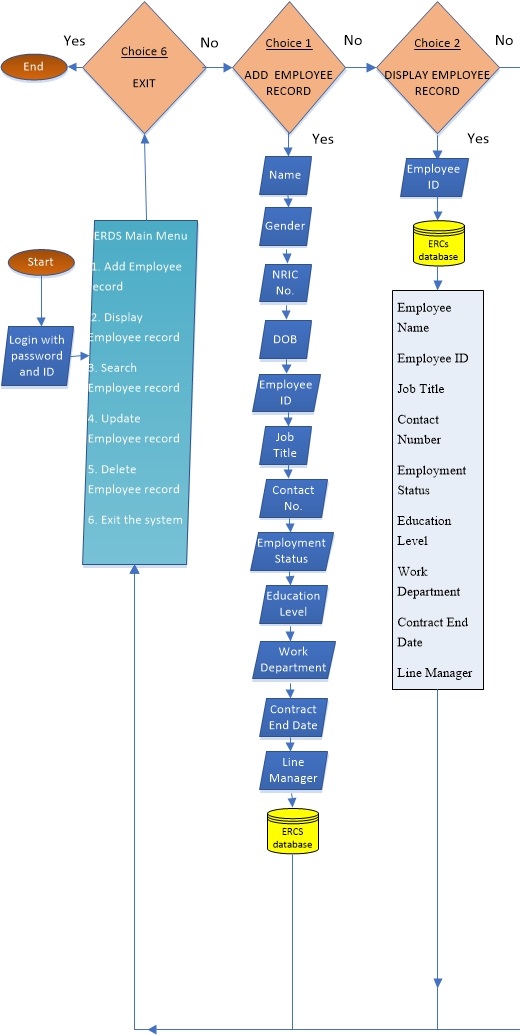
**Search/ Obtain Employment Status**

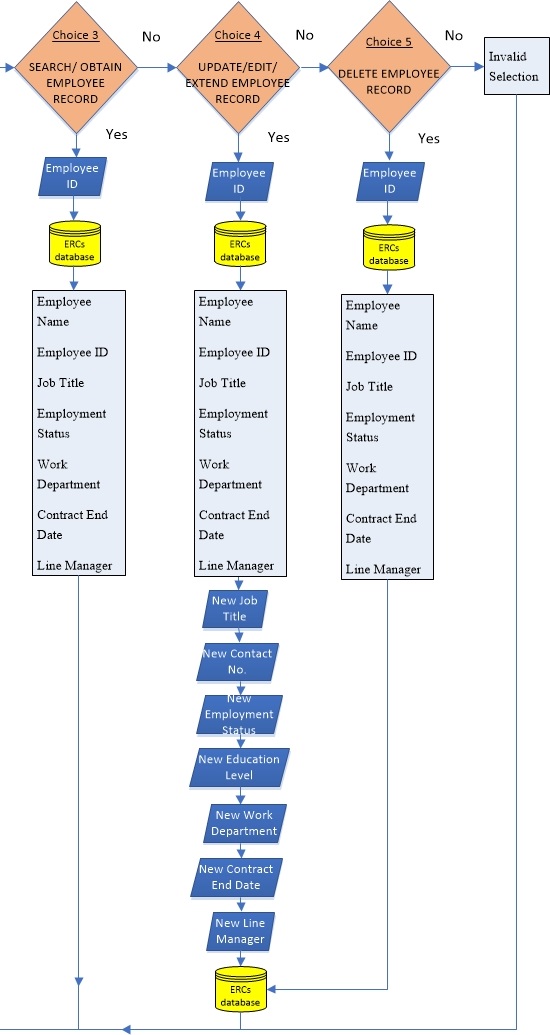
1. Employee Name
2. Employee ID
3. Job Title
4. Employment Status
5. Work Department
6. Contract End Date
7. Line Manager

**Choice 6. Exit**

1. Exit

**PROGRAM DESIGN**: FLOW CHART (continued next page)





INPUT SCREEN DESIGN

* **Add New Employee (option 1)**

Employee Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Gender: A

NRIC Number: 99999999

Employee DOB: 99/99/9999

Employee ID: 99999999

Job Title: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contact Number: 99999999

Employment Status: AAAAAAA

Education Level: AAAAAAA

Work Department: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contract End Date: 99/99/9999

Line Manger: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

* **Display Employee Records (option 2)**

Employee Name - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employee ID - 99999999

Job Title - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contact Number - 99999999

Employment Status - AAAAAAA

Education Level - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Work Department - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contract End Date - 99/99/9999

Line Manager - AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

* **Search/Obtain Employee Records (option 3)**

Enter employee id to be searched

AAAAAAAA

Employee Name AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employee ID 99999999

Job Title AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employment Status AAAAAAA

Work Department AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contract End Date 99/99/9999

Line Manager AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

* **Update/Edit/Extend Employee Records (option 4)**

Employee Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employee ID: 99999999

Job Title: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employment Status: AAAAA

Education Level: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Work Department: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contract End Date: 99/99/9999

Line Manager: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Enter new Job Title: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Enter new contact number: 99999999

Enter new employment status: AAAAAAA

Enter new education level: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Enter new work department: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Enter new contract end date: 99/99/9999

Enter new line manager: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Updating done ====================================================

* **Delete employee details (option 5)**

Enter employee id to be deleted

AAAAAAAA

Employee Name AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employee ID 99999999

Job Title AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Employment Status AAAA

Work Department AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Contract End Date 99/99/9999

Line Manager AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Deletion done ============================================

Contract End Date: 99-99-9999

SOFTWARE REQUIREMENTS

* Windows 98, 2000, XP, Windows Vista, 7, 8 or 8.1 OS
* Must have DOS Command Prompt

HARDWARE REQUIREMENTS

* Recommended 1GB RAM
* 30GB Hard Disk
* Keyboard
* Mouse
* Monitor (any brand)

CONCLUSION

The aim of the system is to enhance record filing system from manual input into computerised system where returns can be immediate and results are accurate and fast. Such system ensures records are permanent, can be easily backed up, and protect against loss of confidential employee records. Lastly, this program will be able to fit the needs of ERDs, checking against every requirement.

APPENDIX II **Proposal Acceptance Form**

**Project Title:** Employee Records Database System

**Software Requirement:**

• Windows 98, 2000, XP, Windows Vista, 7, 8 or 8.1 OS

• Must have DOS Command Prompt

**Hardware Requirement:**

• Recommended 1GB RAM

• 30GB Hard Disk

• Keyboard

• Mouse

• Monitor (any brand)

**Abstract:**A company particular with access security wants computerised employee records database system over its manual filing to keep up with demands and growing security needs.

**Project Status:** Reject Pending Refinement Accept

**Date Received: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date of Approval: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Approved By: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

APPENDIX III **Reference Section**

1 Title: Java All-In-One 4th Edition

Author: Doug Lowe

Publisher: John Wiley & Sons, Inc.

ISBN: 978-1-118-40803-2 (pbk)

2 Title: Java Programming

Author: Joyce Farrel

Publisher: Course Technology

ISBN: 0-7600-1070-b

3 Title: Java Programming

Author: Joyce Farrel

Publisher: Course Technology

ISBN: 0-7600-1070-b

4 Title: Programming Simplified

Retrieved 16 June 2015 from

URL address: <http://www.programmingsimplified.com/java-source-codes>

5 Title: Free Java Guide

Retrieved 18 June 2015 from

URL address: <http://www.freejavaguide.com/javasource3.htm>

6 Title: Java Programming Tutorial

Retrieved 20 June 2015 from

URL address: <http://www.ntu.edu.sg/home/ehchua/programming/java/J2_Basics.html>

7 Title: How do I parse a text string into date and time?

Retrieved 22 June 2015 from

URL address: <http://kodejava.org/>

8 Title: How can I convert .class files to .java files?

Retrieved 21 June 2015 from

URL address: <http://stackoverflow.com/questions/5451470/how-can-i-convert-class-files-to-java-files>

9 Title: How to use clear screen in Java

Retrieved 21 June 2015 from

URL address: ht<tp://stackoverflow.com/questions/19142548/how-to-use-clear-sc>reen-in-java