19AIE203 – Data Structures and Algorithms 2

Team 7 End Sem Report

Implementation of DIGITAL PAYMENT SYSTEM INTERFACE

B Srivathsan	BL.EN.U4AIE20006
Samirit Saha	BL.EN.U4AIE20058
Kanisettypalli Karthik	BL.EN.U4AIE20025

Course Instructor :

Ms. Radha. D

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

TABLE OF CONTENTS

ABSTRACT:

In the given project, we will be making a graphical user interface (GUI) which will be implemented using the concepts of hashing and graphs, to store the details of the accounts and using QR codes which we shall scan and obtain the data and other similar data such as transaction IDs are also used for this purpose. In this project, we will be replicating the transaction from one account to another. It will be done using GUI, which similarly resembles the GUI of an ATM.

INTRODUCTION:

We are using Object Oriented Programming (OOP) concepts such as class, object, interface, single, multiple inheritance, method overloading and method overriding. These are the data structures which we are using to implement our project:

HashMap is used to store the key-value pair. We are using graphs to connect the banks and accounts in the bank. We use unique packages and libraries within Java to generate and read QR codes, which are created for each individual bank account for every bank. These graphs will check the link between the account number and the details of the bank account. We have created the code in such a way that we can also create multiple accounts for the same person in multiple banks.

GRAPH AND ITS' WORKING:

In our Java-based project, the **Graph** is a data structure that performs the function of storing a certain amount of data. The concept of the **graph** is based on the concept from mathematics which fulfills the need of the computer science field. Graph represents a network that connects multiple points to each other. Let us explore some important concepts pertaining to graph which are used in our project:

GRAPH

A **graph** is a data structure that stores connected data. In other words, a graph G (or g) is defined in the following manner: it is a set of vertices (V) and edges (E) that connects those vertices. Examples of graphs in computer science include social media networks, computer networks, Google Maps, etc.

Each graph consists of the following two components: **edges** and **vertices** (also called nodes). Each vertex and edge have a relation between them. A vertex represents the data while an edge represents the relation between them. Vertices are denoted by a circle with a label on them. Edges are denoted by a line that connects the nodes (vertices).

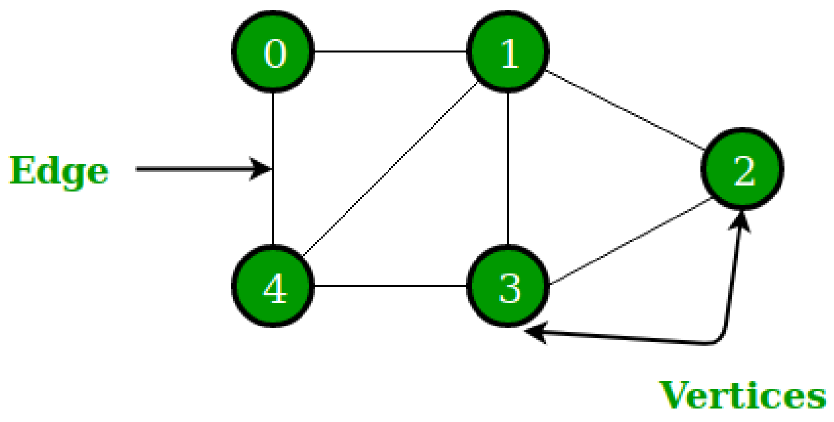


FIGURE 1: DIAGRAM OF A GRAPH WITH ITS' EDGES AND VERTICES

HASHMAP AND ITS' WORKING:

Java's **HashMap** class is a class which implements the Map interface. It allows us *to* store information as a *key and value pair*, where the keys should be unique. If a duplicate key is inserted, the element of the corresponding key will be replaced by it. The HashMap makes it easy to perform operations using the key index, which include operations like updating, deletion, etc. HashMap class is found in the java.util package.

HashMap in Java is like the legacy Hash table class; however, it is not synchronized. Null elements can be stored as well, however there should be only one null key. Since Java 5, it is denoted as HashMap <K,V>, where K stands for key and V for value. It inherits the Abstract Map class and implements the Map interface. A few important points regarding HashMap are:

- Java HashMap contains values based on the key.

- Java HashMap contains only unique keys.
- Java HashMap can have one null key and multiple null values.
- Java HashMap is non synchronized in nature.
- Java HashMap maintains no order.

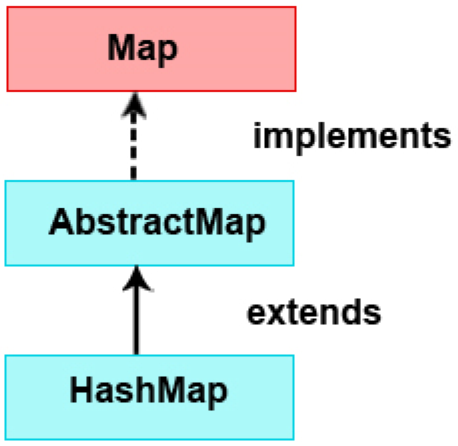


FIGURE 2: FLOWCHART OF THE RELATIONSHIP OF HASHMAP WITH MAP AND ABSTRACTMAP

JAVA’S QR CODE GENERATOR AND READER:

QR code is the shortened form for **Quick Response code**. It is an alternative to the bar code or a two-dimensional bar-code. It contains the matrix of small squares in which information is stored. Its’ working is the same as a bar-code. It can be read by an imaging device. Nowadays, it's being used a lot in various fields such as GEO location, on products, vehicle tracking, etc.

The QR code is made up of modules (black and white squares) which contain the encoded data. The modules are arranged in the row and columns which form the **Data Matrix**. The placing of data bits starts from the bottom right corner and then moves in an upward direction and changes the position while reach at the top.

The QR code is scanned through an imaging device such as a camera. After scanning, the QR code is processed by it using the **Reed-Solomon Error Correction Code** until the image is interpreted. After that, the data is extracted from the patterns that are stored in the horizontal and vertical components of the code.

To generate a QR code in Java, a third-party library named **ZXing (Zebra Crossing)**. ZXing is a popular API that allows us to generate and process the QR code. With the help of the library, we can easily generate and read the QR code. Before moving towards the Java program, we need to add the ZXing library to the project. It can be downloaded from the official site. After generating the QR code, the QR code image file can also be read using **ZXing library**.

The components of QR code are:

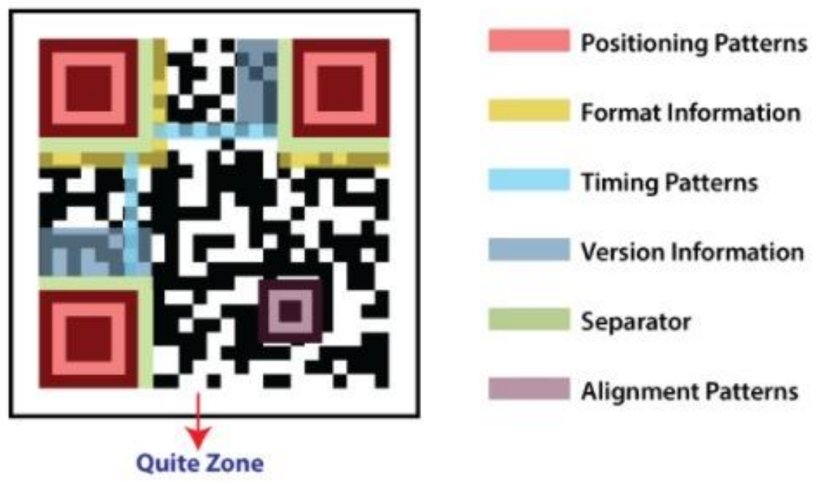


FIGURE 3: QR CODE AND ITS' COMPONENTS

ERROR CORRECTION LEVELS:

A QR code can be scanned even if it is damaged somewhat (up to 30%). The **Reed-Solomon Correction** algorithm makes it possible. It is to be noted that if we add error correction, it increases the number of the data block in the QR code. So, we need to adjust the level of error correction according to requirements. There are four levels of error correction

1.Low(L)

2.Medium(M)

3.Quartile(Q)

4.High(H)

7% is error correction of Low level

15% is error correction of Low level

25% is error correction of Low level

30% is error correction of Low level

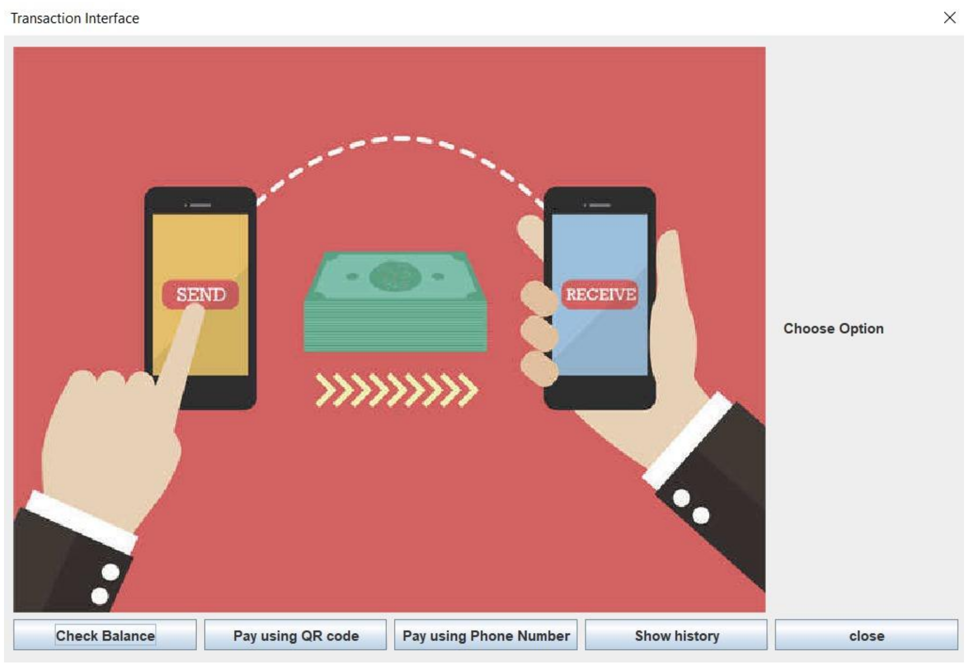
PACKAGES USED:

```
import java.util.*;
import java.io.File;
import java.io.IOException;
import javax.swing.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

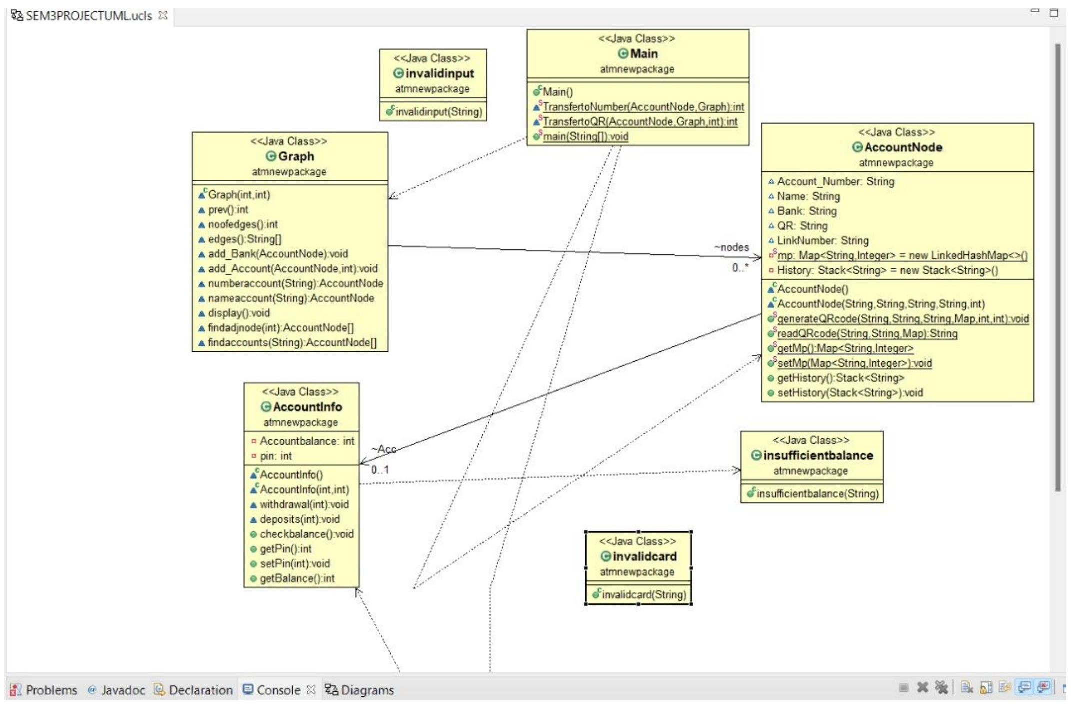
For Reading QR code:

```
import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.NotFoundException;
import com.google.zxing.WriterException;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;
import javax.imageio.ImageIO;
import com.google.zxing.BinaryBitmap;
import com.google.zxing.MultiFormatReader;
import com.google.zxing.Result;
import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
import com.google.zxing.common.HybridBinarizer;
```

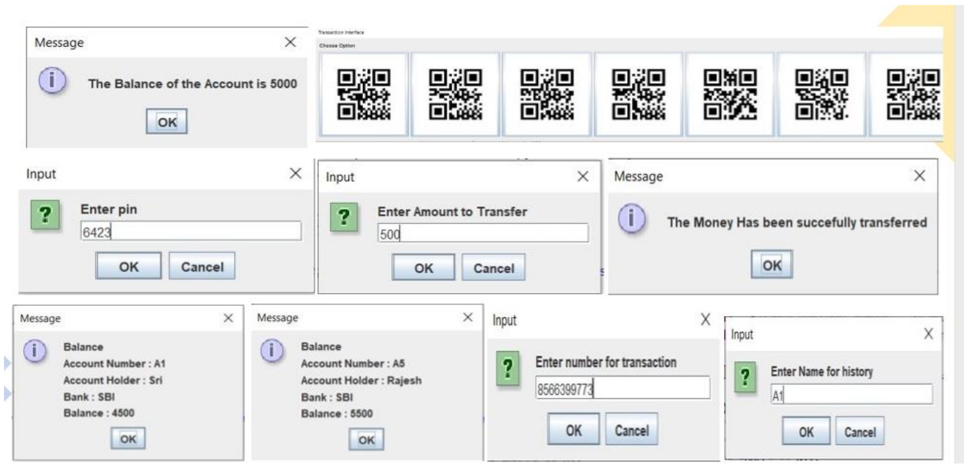
GRAPHICAL USER INTERFACE:



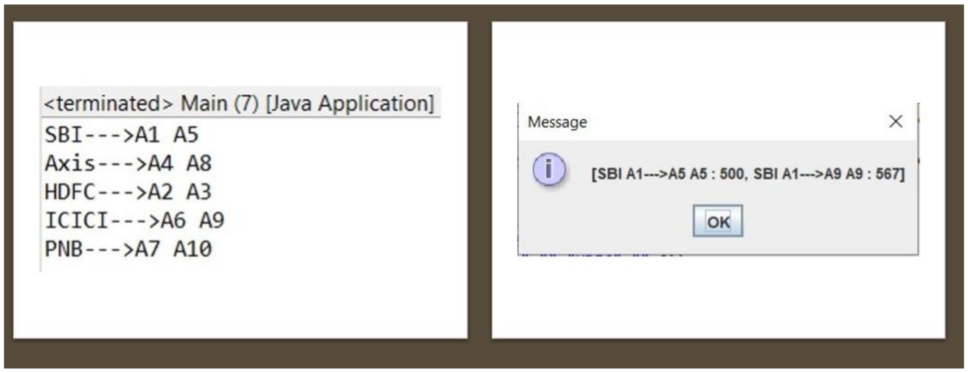
UML CLASS DIAGRAM:



RESULTS:



```
Message
The Balance of the Account is 5000
OK

Input
Enter pin
6423
OK  Cancel

Input
Enter Amount to Transfer
500
OK  Cancel

Message
The Money Has been succefully transferred
OK

Message
Balance
Account Number : A1
Account Holder : Sri
Bank : SBI
Balance : 4500
OK

Message
Balance
Account Number : A5
Account Holder : Rajesh
Bank : SBI
Balance : 5500
OK

Input
Enter number for transaction
8566399773
OK  Cancel

Input
Enter Name for history
A1
OK  Cancel
```



```
<terminated> Main (7) [Java Application]
SBI--->A1 A5
Axis--->A4 A8
HDFC--->A2 A3
ICICI--->A6 A9
PNB--->A7 A10
```

```
Message
[SBI A1--->A5 A5 : 500, SBI A1--->A9 A9 : 567]
OK
```

CODE:

```
package Transaction;


import java.util.*;
import java.io.File;
import java.io.IOException;
import javax.swing.*;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.NotFoundException;
import com.google.zxing.WriterException;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import javax.imageio.ImageIO;
import com.google.zxing.BinaryBitmap;
import com.google.zxing.MultiFormatReader;
import com.google.zxing.Result;
import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
import com.google.zxing.common.HybridBinarizer;



class AccountInfo {
private int Accountbalance;
   private int pin;
AccountInfo(){}
AccountInfo(int ACB,int p){
Accountbalance=ACB;
setPin(p);}

void withdrawal(int withdraw) {
        try {
            if (withdraw > Accountbalance)
                throw new insufficientbalance("insuffcient balance");
        } catch (insufficientbalance j) {
            System.out.println(j);
        }
        if (Accountbalance >= withdraw) {
            Accountbalance = Accountbalance - withdraw;
        } else {
              JOptionPane.showMessageDialog(null,JOptionPane.WARNING_MESSAGE,"Insufficient Balance",
0);}
        System.out.println("");}


    void deposits(int deposit) {
        Accountbalance = Accountbalance + deposit;
        JOptionPane.showMessageDialog(null,"The Money Has been succefully transferred");
        System.out.println("");}


    public void checkbalance() {
        System.out.println("Balance : " + Accountbalance);
        System.out.println("");}
```

```
public int getPin() {
return pin;}
public void setPin(int pin) {
this.pin = pin;}
public int getBalance() {
return Accountbalance;}}




class AccountNode{
String Account_Number;
String Name;
String Bank;
String QR;
AccountInfo Acc;
String LinkNumber;
private static Map<String, Integer> mp = new LinkedHashMap<>();
private Stack<String> History= new Stack<String>();
AccountNode(){
Account_Number="";
Name="";
Bank="";
Acc=new AccountInfo();}
AccountNode(String a,String N,String b,String LN,int p) throws WriterException, IOException,
NotFoundException {
Account_Number=a;
Name=N;
Bank=b;

LinkNumber=LN;

Acc=new AccountInfo(5000,p);

getMp().put(Account_Number,p);

QR="D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR codes\\"+Account_Number+".png";

Map<EncodeHintType, ErrorCorrectionLevel> hashMap = new HashMap<EncodeHintType,
ErrorCorrectionLevel>();

hashMap.put(EncodeHintType.ERROR_CORRECTION, ErrorCorrectionLevel.L);

generateQRcode(Account_Number,QR,"UTF-8", hashMap, 200, 200);      }


@SuppressWarnings("deprecation")

public static void generateQRcode(String data, String path, String charset,
@SuppressWarnings("rawtypes") Map map, int h, int w) throws WriterException, IOException  {

BitMatrix matrix = new MultiFormatWriter().encode(new String(data.getBytes(charset), charset),
BarcodeFormat.QR_CODE, w, h);

MatrixToImageWriter.writeToFile(matrix, path.substring(path.lastIndexOf('.') + 1), new File(path));  }

public static String readQRcode(String path, String charset, @SuppressWarnings("rawtypes") Map map)
throws FileNotFoundException, IOException, NotFoundException  {

BinaryBitmap binaryBitmap = new BinaryBitmap(new HybridBinarizer(new
BufferedImageLuminanceSource(ImageIO.read(new FileInputStream(path)))));

Result rslt = new MultiFormatReader().decode(binaryBitmap);

return rslt.getText();  }

public static Map<String, Integer> getMp() {

return mp;}

public static void setMp(Map<String, Integer> mp) {

AccountNode.mp = mp;}

public Stack<String> getHistory() {

return History;}

public void setHistory(Stack<String> history) {

History = history;}}
```

```
class Graph{
AccountNode nodes[][];
Graph(int M,int N){
nodes=new AccountNode[M][N];}
int prev() {
int A=0;
for (int i=0;i<nodes.length;i++) {
for (int j=0;j<nodes[i].length;j++) {
if (nodes[j][i]!=null){
if (i>A) {A=i;}
if (j>A) {A=j;}}}}
return A;}


int noofedges() {
int A=0;
for (int i=0;i<nodes.length;i++) {
for (int j=0;j<nodes[i].length;j++) {
if (nodes[j][i]!=null){A=A+1;}}}
return A;}


String [] edges() {
int count=0;
String edges[]=new String[noofedges()-1];
for (int i=0;i<prev()+1;i++) {
```

```
for (int j=0;j<prev()+1;j++) {
if (nodes[i][j]!=null ){
edges[count]=Integer.toString(j)+Integer.toString(i);
count=count+1;}}}
return edges;}
void add_Bank(AccountNode val) {
switch (val.Bank) {
   case "SBI":
            nodes[1][0]=val;
      break;
   case "Axis":
            nodes[2][0]=val;
      break;
   case "HDFC":
            nodes[3][0]=val;
      break;
   case "ICICI":
            nodes[4][0]=val;
      break;
   case "PNB":
            nodes[5][0]=val;
      break;}}


void add_Account(AccountNode val,int B) {
switch (val.Bank) {
   case "SBI":
            nodes[B][1]=val;
```

```
        break;
    case "Axis":
            nodes[B][2]=val;
        break;
    case "HDFC":
            nodes[B][3]=val;
        break;
    case "ICICI":
            nodes[B][4]=val;
        break;
    case "PNB":
            nodes[B][5]=val;
        break;}}
AccountNode numberaccount (String number) {
boolean found =false;
AccountNode a=null;
for (int i=0;i<nodes.length;i++) {
for (int j=0;j<nodes[i].length;j++) {
if (nodes[j][i]!=null){
if (nodes[j][i].LinkNumber==number) {
a=nodes[j][i];
found=true;
return a;}}}}
if (found==false) {System.out.println("No Account Linked with Number");
return a;}
else {return a;}}
```

```
AccountNode nameaccount (String accnumname) {
boolean found =false;
AccountNode a=null;
for (int i=0;i<nodes.length;i++) {
for (int j=0;j<nodes[i].length;j++) {
if (nodes[j][i]!=null){
if (nodes[j][i].Account_Number==accnumname) {
a=nodes[j][i];
found=true;
return a;}}}}
if (found==false) {System.out.println("No Account Linked with Number");
return a;}
else {return a;}}



void display() {
for (int i=1;i<6;i++) {
System.out.print(nodes[i][0].Bank+"--->");
for (int j=0;j<findadjnode(i).length;j++) {
System.out.print(findadjnode(i)[j].Account_Number+" ");}
System.out.println();}}



AccountNode[] findadjnode(int X) {
int count=0;
AccountNode adj[]=new AccountNode[noofedges()-1];
for (int i=0;i<prev()+1;i++) {
if (nodes[i][X]!=null){
```

```
adj[count]=nodes[i][X];
count=count+1;}}
AccountNode temp[]=new AccountNode[count];
for (int i = 0; i < temp.length; i++) {
    temp[i] = adj[i];}
return temp;}
AccountNode[] findaccounts(String L) {
int X=0;
switch (L) {
  case "SBI":
       X=1;
    break;
  case "Axis":
       X=2;
    break;
  case "HDFC":
       X=3;
    break;
  case "ICICI":
       X=4;
    break;
  case "PNB":
       X=5;
    break;}
int count=0;
AccountNode adj[]=new AccountNode[noofedges()-1];
for (int i=0;i<prev()+1;i++) {
if (nodes[i][X]!=null){
```

```
adj[count]=nodes[i][X];
count=count+1;}}
AccountNode temp[]=new AccountNode[count];
for (int i = 0; i < temp.length; i++) {
    temp[i] = adj[i];}
return temp;}}


public class Main {
@SuppressWarnings("resource")
static int TransfertoNumber(AccountNode A,Graph g){
Scanner s1=new Scanner(System.in);
String number="";
String num="";
num=JOptionPane.showInputDialog(null, "Enter number for transaction");
switch (num) {
  case "8759458364":
       number="8759458364";
    break;
  case "8279458312":
       number="8279458312";
    break;
  case "8995468312":
       number="8995468312";
    break;
  case "9688458312":
       number="9688458312";
    break;
```

```
   case "7465288663":
         number="7465288663";
     break;
   case "6368277669":
         number="6368277669";
     break;
   case "9884432318":
         number="9884432318";
     break;
   case "8566399773":
         number="8566399773";
     break;
   case "7679388771":
         number="7679388771";
     break;    }
AccountInfo b= g.numberaccount(number).Acc;
if (b==null) {
s1.close();
return 0;}
int p=Integer.parseInt(JOptionPane.showInputDialog(null, "Enter pin"));
if (p!=AccountNode.getMp().get(A.Account_Number)) {
JOptionPane.showMessageDialog(null,JOptionPane.WARNING_MESSAGE,"Incorrect Pin", 0);
return 0;}
int n=Integer.parseInt(JOptionPane.showInputDialog(null, "Enter Amount to Transfer"));
A.Acc.withdrawal(n);
b.deposits(n);
A.getHistory().add(A.Bank+" "+A.Account_Number+"---
>"+g.numberaccount(number).Account_Number+" "+g.numberaccount(number).Account_Number+" :
"+n);
```

```
g.numberaccount(number).getHistory().add(g.numberaccount(number).Bank+"
"+g.numberaccount(number).Account_Number+"<---"+A.Bank+" "+A.Account_Number+" : "+n);

int bal=A.Acc.getBalance();

JOptionPane.showMessageDialog(null, "Balance \nAccount Number : "+A.Account_Number+"\nAccount
Holder : "+A.Name+"\n Bank : "+A.Bank+" \nBalance : "+bal);

JOptionPane.showMessageDialog(null, "Balance \nAccount Number :
"+g.numberaccount(number).Account_Number+"\nAccount Holder :
"+g.numberaccount(number).Name+"\n Bank : "+g.numberaccount(number).Bank+" \nBalance :
"+b.getBalance());

return 1;}


@SuppressWarnings("resource")

static int TransfertoQR(AccountNode A,Graph g,int r) throws WriterException, IOException,
NotFoundException{

String path="";

switch (r) {

  case (0):

         path="A2";

    break;

  case (1):

         path="A3";

    break;

  case (2):

         path="A4";

    break;

  case (3):

         path="A5";

    break;

  case (4):

         path="A6";
```

```
        break;
    case (5):
            path="A7";
        break;
    case (6):
            path="A8";
        break;
    case (7):
            path="A9";
        break;
    case (8):
            path="A10";
        break;}
String inp="";
String InQR="D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR codes\\"+path+".png";
Map<EncodeHintType, ErrorCorrectionLevel> hintMap = new HashMap<EncodeHintType,
ErrorCorrectionLevel>();
hintMap.put(EncodeHintType.ERROR_CORRECTION, ErrorCorrectionLevel.L);
String name=AccountNode.readQRcode(InQR, "UTF-8", hintMap);
switch (name) {
    case "A2":
            inp="A2";
        break;
    case "A3":
            inp="A3";
        break;
    case "A4":
            inp="A4";
        break;
    case "A5":
            inp="A5";
        break;
    case "A6":
            inp="A6";
        break;
    case "A7":
            inp="A7";
        break;
    case "A8":
            inp="A8";
        break;
    case "A9":
            inp="A9";
        break;
    case "A10":
            inp="A10";
        break;}
AccountInfo b= g.nameaccount(inp).Acc;
if (b==null) {
return 0;}
int p=Integer.parseInt(JOptionPane.showInputDialog(null, "Enter pin"));
if (p!=AccountNode.getMp().get(A.Account_Number)) {
JOptionPane.showMessageDialog(null,JOptionPane.WARNING_MESSAGE,"Incorrect Pin", 0);
return 0;}
int n=Integer.parseInt(JOptionPane.showInputDialog(null, "Enter Amount to Transfer"));
A.Acc.withdrawal(n);
```

```
b.deposits(n);

A.getHistory().add(A.Bank+" "+A.Account_Number+"--->"+g.nameaccount(inp).Account_Number+" "+g.nameaccount(inp).Account_Number+" : "+n);

g.nameaccount(inp).getHistory().add(g.nameaccount(inp).Bank+" "+g.nameaccount(inp).Account_Number+"<---"+A.Bank+" "+A.Account_Number+" : "+n);

int bal=A.Acc.getBalance();

JOptionPane.showMessageDialog(null, "Balance \nAccount Number : "+A.Account_Number+"\nAccount Holder : "+A.Name+"\nBank : "+A.Bank+" \nBalance : "+bal);

JOptionPane.showMessageDialog(null, "Balance \nAccount Number : "+g.nameaccount(inp).Account_Number+"\nAccount Holder : "+g.nameaccount(inp).Name+"\nBank : "+g.nameaccount(inp).Bank+" \nBalance : "+b.getBalance());

return 1;}

  public static void main(String args[]) throws WriterException, IOException, NotFoundException {

        Graph g=new Graph(20,20);

        Scanner sc=new Scanner(System.in);

        g.add_Bank(new AccountNode("B1","","SBI","",0));

        g.add_Bank(new AccountNode("B2","","Axis","",0));

        g.add_Bank(new AccountNode("B3","","HDFC","",0));

        g.add_Bank(new AccountNode("B4","","ICICI","",0));

        g.add_Bank(new AccountNode("B5","","PNB","",0));

g.add_Account(new AccountNode("A1","Sri","SBI","8599458312",6423),g.findaccounts("SBI").length);

g.add_Account(new AccountNode("A2","Irs","HDFC","8759458364",7514),g.findaccounts("HDFC").length);

g.add_Account(new AccountNode("A3","Karthik","HDFC","8279458312",8923),g.findaccounts("HDFC").length);

g.add_Account(new AccountNode("A4","Samirit","Axis","8995468312",1234),g.findaccounts("Axis").length);

g.add_Account(new AccountNode("A5","Rajesh","SBI","9688458312",4321),g.findaccounts("SBI").length);

g.add_Account(new AccountNode("A6","Sonu","ICICI","7465288663",3467),g.findaccounts("ICICI").length);
```

```
g.add_Account(new
AccountNode("A7","Monu","PNB","6368277669",1754),g.findaccounts("PNB").length);

g.add_Account(new
AccountNode("A8","Gonu","Axis","9884432318",1262),g.findaccounts("Axis").length);

g.add_Account(new
AccountNode("A9","Lonu","ICICI","8566399773",2789),g.findaccounts("ICICI").length);

g.add_Account(new
AccountNode("A10","Donu","PNB","7679388771",5423),g.findaccounts("PNB").length);

g.display();

AccountNode own=g.nameaccount("A1");

while(true) {

ImageIcon icon=new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\Pay.png");

String[] buttons = { "Check Balance","Pay using QR code","Pay using Phone Number","Show
history","close" };

int returnValue = JOptionPane.showOptionDialog(null, "Choose Option","Transaction Interface",

       JOptionPane.NO_OPTION,JOptionPane.PLAIN_MESSAGE, icon, buttons,"default");


menu1:

switch (returnValue) {

case (0):

int bal=own.Acc.getBalance();

JOptionPane.showMessageDialog(null, "The Balance of the Account is "+bal);

   break menu1;

case (1):

   Object[] buttons1 = { new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A2.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A3.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A4.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A5.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A6.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A7.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A8.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A9.png"),new ImageIcon("D:\\Srivathsan\\eclipse-workspace\\Transaction\\QR
codes\\A10.png")};

int returnValue1 = JOptionPane.showOptionDialog(null, "Choose Option","Transaction Interface",

        JOptionPane.NO_OPTION,JOptionPane.PLAIN_MESSAGE,null,buttons1,"default");

TransfertoQR(own,g,returnValue1);

    break menu1;

case (2):

TransfertoNumber(own,g);

    break menu1;

case (3):

String his=JOptionPane.showInputDialog(null, "Enter Name for history");

String inp="";

switch (his) {

case "A1":

inp="A1";

break;

case "A2":

inp="A2";

break;

case "A3":

inp="A3";

break;

case "A4":

inp="A4";

break;

case "A5":

inp="A5";

break;
```

```
case "A6":
inp="A6";
break;
case "A7":
inp="A7";
break;
case "A8":
inp="A8";
break;
case "A9":
inp="A9";
break;
case "A10":
inp="A10";
break;}
JOptionPane.showMessageDialog(null,g.nameaccount(inp).getHistory());
   break menu1;
case (4):
sc.close();
   System.exit(0);
break ;  }}}}



@SuppressWarnings("serial")
class invalidinput extends Exception {
   public invalidinput(String s) {
      super(s);}}
```

```
@SuppressWarnings("serial")
class insufficientbalance extends Exception {
    public insufficientbalance(String p) {
        super(p);}}


@SuppressWarnings("serial")
class invalidcard extends Exception {
    public invalidcard(String x) {
        super(x);}}
```