

Actotal.java

```
1 package test;
2 import java.io.Serializable;
4 public class Actotal extends JFrame implements Serializable {
5     private int total;
6     private int account;
7     private int depo;
8     private int withdrawl;
9     private String date;
10    private String time;
11    public String getDate() {
12        return date;
13    }
14    public void setDate(String date) {
15        this.date = date;
16    }
17    public String getTime() {
18        return time;
19    }
20    public void setTime(String time) {
21        this.time = time;
22    }
23    public int getTotal() {
24        return total;
25    }
26    public void setTotal(int total) {
27        this.total = total;
28    }
29    public int getAccount() {
30        return account;
31    }
32    public void setAccount(int account) {
33        this.account = account;
34    }
35    public int getDepo() {
36        return depo;
37    }
38    public void setDepo(int depo) {
39        this.depo = depo;
40    }
41    public int getWithdrawl() {
42        return withdrawl;
43    }
44    public void setWithdrawl(int withdrawl) {
45        this.withdrawl = withdrawl;
46    }
47 }
48
```

Actotalcreate.java

```

1 package test;
2 import java.util.Calendar;
6 public class Actotalcreate extends JFrame {
7     private static final long serialVersionUID = -948627250006298669L;
8     private String sysdate, systime;
9     Actotal sd=new Actotal();
10    Actotalcreate(int a,int b,int d,int t) {
11        Calendar cal = Calendar.getInstance();
12        String cday = ""+cal.get(Calendar.DATE);
13        int x =cal.get(Calendar.MONTH);
14        String cmonth = ""+(x+1);
15        String cyear = ""+cal.get(Calendar.YEAR);
16        sysdate = cday+"/"+cmonth+"/"+cyear;
17        String chr = ""+cal.get(Calendar.HOUR_OF_DAY);
18        String cmin = ""+cal.get(Calendar.MINUTE);
19        String csec =""+cal.get(Calendar.SECOND);
20        systime = chr+":"+cmin+": "+csec;
21        sd.setAccount(a);
22        sd.setWithdrawl(b);
23        sd.setDepo(d);
24        sd.setTotal(t);
25        sd.setDate(sysdate);
26        sd.setTime(systime);
27        ArrayList<Actotal> list4;
28        try {
29            FileInputStream fin=new FileInputStream("actotal.dat");
30            ObjectInputStream oin=new ObjectInputStream(fin);
31            list4=(ArrayList<Actotal>)oin.readObject();
32        }catch(Exception e){
33            list4=new ArrayList<Actotal>();
34        }
35        list4.add(sd);
36        try {
37            FileOutputStream fout=new FileOutputStream("actotal.dat");
38            ObjectOutputStream oout=new ObjectOutputStream(fout);
39            oout.writeObject(list4);
40        }catch(Exception e){}
41    }
42 }
43

```

AddInformation1.java

```
1 package test;
2 import java.io.*;
4 public class AddInformation1 {
5     ArrayList<Create> list4;
6     public AddInformation1(Create rg1) {
7         try {
8             FileInputStream fin=new FileInputStream("Regis.dat");
9             ObjectInputStream oin=new ObjectInputStream(fin);
10            list4=(ArrayList<Create>)oin.readObject();
11        } catch (Exception e) {
12            list4=new ArrayList<Create>();
13        }
14        list4.add(rg1);
15        try {
16            FileOutputStream fout=new FileOutputStream("Regis.dat");
17            ObjectOutputStream oout=new ObjectOutputStream(fout);
18            oout.writeObject(list4);
19        } catch (Exception e){}
20    }
21 }
22
```

```

1 package test;
2 import java.awt.event.*;
3 class Admin extends JFrame implements ActionListener {
4     private JLabel l1,l2;
5     private JButton b1,b2,b3,b4,b5,b6,b7,b8;
6     private JButton b11;
7     private JButton b12;
8     private JButton b13;
9     public Admin() {
10         super("Welcome Administrator");
11         Container c=getContentPane();
12         c.setLayout(new GridLayout(8,2));
13         Font f1=new Font("Times New Roman",Font.BOLD,20);
14         l1=new JLabel("ADMINISTRATOR");
15         l1.setFont(f1);
16         l1.setForeground(Color.RED);
17         JPanel bpanel=new JPanel();
18         bpanel.add(l1);
19         bpanel.setBackground(new Color(0,0,64));
20         l2=new JLabel("");
21         l2.setFont(f1);
22         l2.setForeground(Color.RED);
23         JPanel cpanel=new JPanel();
24         cpanel.add(l2);
25         cpanel.setBackground(new Color(0,0,64));
26         b1=new JButton("ADD AN OPERATOR");
27         b1.addActionListener(this);
28         JPanel apanel=new JPanel();
29         apanel.add(b1);
30         apanel.setBackground(new Color(0,0,64));
31         b2=new JButton("SEARCH AN OPERATOR");
32         b2.addActionListener(this);
33         JPanel dpanel=new JPanel();
34         dpanel.add(b2);
35         dpanel.setBackground(new Color(0,0,64));
36         b3=new JButton("DELETE AN OPERATOR");
37         b3.addActionListener(this);
38         JPanel epanel=new JPanel();
39         epanel.add(b3);
40         epanel.setBackground(new Color(0,0,64));
41         b4=new JButton("SHOW ALL OPERATORS");
42         b4.addActionListener(this);
43         JPanel fpanel=new JPanel();
44         fpanel.add(b4);
45         fpanel.setBackground(new Color(0,0,64));
46         b5=new JButton("SHOW ALL USERS");
47         b5.addActionListener(this);
48         JPanel gpanel=new JPanel();
49         gpanel.add(b5);
50         gpanel.setBackground(new Color(0,0,64));
51         b6=new JButton("SEARCH AN USER");
52         b6.addActionListener(this);
53         JPanel hpanel=new JPanel();
54         hpanel.add(b6);
55         hpanel.setBackground(new Color(0,0,64));
56         b7=new JButton("DELETE AN USER");
57         b7.addActionListener(this);
58         JPanel opanel=new JPanel();
59         opanel.add(b7);
60         opanel.setBackground(new Color(0,0,64));
61         b8=new JButton("LOG OUT");
62         b8.addActionListener(this);

```

```

65     JPanel qpanel=new JPanel();
66     qpanel.add(b8);
67     qpanel.setBackground(new Color(0,0,64));
68     b11=new JButton("LOCKER DETAILS OF USER");
69     b11.addActionListener(this);
70     JPanel qoopanel=new JPanel();
71     qoopanel.add(b11);
72     qoopanel.setBackground(new Color(0,0,64));
73     b12=new JButton("FIXED DEPOSIT DETAILS");
74     b12.addActionListener(this);
75     JPanel qipanel=new JPanel();
76     qipanel.add(b12);
77     qipanel.setBackground(new Color(0,0,64));
78     b13=new JButton("LOCKER APPROVE");
79     b13.addActionListener(this);
80     JPanel q2panel=new JPanel();
81     q2panel.add(b13);
82     q2panel.setBackground(new Color(0,0,64));
83     c.add(bpanel);c.add(cpanel);
84     c.add(apanel);c.add(dpanel);
85     c.add(epanel);c.add(fpanel);
86     c.add(gpanel);c.add(hpanel);
87     c.add(qoopanel);c.add(qipanel);
88     c.add(q2panel);c.add(opanel);
89     c.add(qpanel);
90     setSize(600,525);
91     setLocation(200,200);
92     setResizable(false);
93     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
94     setVisible(true);
95 }
96 public void actionPerformed(ActionEvent e) {
97     if(e.getSource()==b1) {
98         new OperatorReg("");
99         setVisible(false);
100    }
101    if(e.getSource()==b2) {
102        new OperatorSearch();
103        setVisible(false);
104    }
105    if(e.getSource()==b3) {
106        new OperatorDelete();
107        setVisible(false);
108    }
109    if(e.getSource()==b4)
110        new OperatorDisplay();
111    if(e.getSource()==b5)
112        new DiaplayAll();
113    if(e.getSource()==b6) {
114        new UserSearch();
115        setVisible(false);
116    }
117    if(e.getSource()==b7) {
118        new AdminUserRemove();
119        setVisible(false);
120    }
121    if(e.getSource()==b8) {
122        int con=JOptionPane.showConfirmDialog(this, "Are You Sure to cancel?");
123        if(con==JOptionPane.YES_OPTION) {
124            new SecondWindow();
125            setVisible(false);
126        }

```

AdminLogin.java

```
127     }  
128 }  
129 }  
130 public class AdminLogin {  
131     public static void main(String[] args) {}  
132 }  
133
```

AdminUserRemove.java

```

1 package test;
2 import java.awt.event.*;
6 public class AdminUserRemove extends JFrame implements ActionListener {
7     private JLabel l1,l2,l3;
8     private JTextField t1;
9     private JButton submit,back;
10    private boolean flagAccNum = false;
11    public AdminUserRemove() {
12        super("Remove an user");
13        Container c=getContentPane();
14        c.setLayout(new GridLayout(3,2));
15        Font f1=new Font("Times New Roman",Font.BOLD,20);
16        l1=new JLabel("USER");
17        l1.setFont(f1);
18        l1.setForeground(Color.RED);
19        JPanel fpanel=new JPanel();
20        fpanel.add(l1);
21        fpanel.setBackground(new Color(0,0,64));
22        l2=new JLabel("Delete");
23        l2.setFont(f1);
24        l2.setForeground(Color.RED);
25        JPanel apanel=new JPanel();
26        apanel.add(l2);
27        apanel.setBackground(new Color(0,0,64));
28        l3=new JLabel("Give USER A/C no");
29        l3.setFont(f1);
30        l3.setForeground(Color.RED);
31        JPanel bpanel=new JPanel();
32        bpanel.add(l3);
33        bpanel.setBackground(new Color(0,0,64));
34        t1=new JTextField();
35        submit=new JButton("Delete");
36        submit.addActionListener(this);
37        JPanel cpanel=new JPanel();
38        cpanel.add(submit);
39        cpanel.setBackground(new Color(0,0,64));
40        back=new JButton("BACK");
41        back.addActionListener(this);
42        JPanel dpanel=new JPanel();
43        dpanel.add(back);
44        dpanel.setBackground(new Color(0,0,64));
45        c.add(fpanel);c.add(apanel);
46        c.add(bpanel);c.add(t1);
47        c.add(cpanel);c.add(dpanel);
48        setSize(550,300);
49        setLocation(200,200);
50        setResizable(false);
51        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
52        setVisible(true);
53    }
54    public void actionPerformed(ActionEvent e) {
55        if(e.getSource() == submit) {
56            String vaccNum = t1.getText();
57            String accNumpattern = "[0-9]{4}" ;
58            Scanner scan = new Scanner(vaccNum) ;
59            String matched = scan.findInLine(accNumpattern) ;
60            if ( matched == null ) {
61                JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
62                t1.setText("");
63            }

```

```
64         else
65             flagAccNum = true;
66         if(flagAccNum == true) {
67             String sname = t1.getText().trim();
68             new UserRemoveDisplay(sname);
69             t1.setText("");
70         }
71         flagAccNum = false;
72     }
73     if(e.getSource() == back) {
74         int con = JOptionPane.showConfirmDialog(this, "Are You Sure to cancel?");
75         if(con == JOptionPane.YES_OPTION) {
76             new Admin();
77             setVisible(false);
78         }
79     }
80 }
81 }
82 }
```


ChangePassword.java

```

1 package test;
2 import java.awt.*;
3
4 public class ChangePassword extends JFrame implements ActionListener {
5     private JLabel lHeading, lFirstName, lCurrentPassword, lNewPassword;
6     private JTextField tFirstName;
7     private JPasswordField pCurrentPassword, pNewPassword;
8     private JButton bBack, bChange;
9     ArrayList<OperatorRegData> list;
10    private boolean flagFirstName = false;
11    private boolean flagNewPassword = false;
12    private boolean flagCurrentPassword = false;
13    private boolean flagFirstNameMatch = false;
14    private boolean flagCurrentPasswordMatch = false;
15    private JLabel lHeading2;
16    public ChangePassword() {
17        super("Change Password");
18        Container c=getContentPane();
19        c.setLayout(new GridLayout(5,2));
20        Font f1=new Font("Chiller",Font.BOLD,22);
21        Font f2=new Font("Times New Roman",Font.BOLD,20);
22        lHeading=new JLabel("CHANGE PASSWORD");
23        lHeading.setFont(f1);
24        lHeading.setForeground(Color.RED);
25        JPanel apanel=new JPanel();
26        apanel.add(lHeading);
27        apanel.setBackground(new Color(0,0,64));
28        lHeading2=new JLabel("");
29        lHeading2.setFont(f1);
30        lHeading2.setForeground(Color.RED);
31        JPanel abpanel=new JPanel();
32        abpanel.add(lHeading2);
33        abpanel.setBackground(new Color(0,0,64));
34        lFirstName=new JLabel("Enter First Name");
35        lFirstName.setFont(f2);
36        lFirstName.setForeground(Color.GRAY);
37        JPanel bpanel=new JPanel();
38        bpanel.add(lFirstName);
39        bpanel.setBackground(new Color(0,0,64));
40        lCurrentPassword=new JLabel("Enter Current Password");
41        lCurrentPassword.setFont(f2);
42        lCurrentPassword.setForeground(Color.GRAY);
43        JPanel cpanel=new JPanel();
44        cpanel.add(lCurrentPassword);
45        cpanel.setBackground(new Color(0,0,64));
46        lNewPassword=new JLabel("Enter New Password");
47        lNewPassword.setFont(f2);
48        lNewPassword.setForeground(Color.GRAY);
49        JPanel dpanel=new JPanel();
50        dpanel.add(lNewPassword);
51        dpanel.setBackground(new Color(0,0,64));
52        tFirstName=new JTextField();
53        pCurrentPassword=new JPasswordField();
54        pNewPassword=new JPasswordField();
55        bChange=new JButton("Change");
56        bChange.addActionListener(this);
57        JPanel epanel=new JPanel();
58        epanel.add(bChange);
59        epanel.setBackground(new Color(0,0,64));
60        bBack=new JButton("Back");
61        bBack.addActionListener(this);
62        JPanel fpanel=new JPanel();
63        fpanel.add(bBack);

```

ChangePassword.java

```

67         fpanel.setBackground(new Color(0,0,64));
68         c.add(amodel);c.add(abpanel);
69         c.add(bpanel);c.add(tFirstName);
70         c.add(cpanel);c.add(pCurrentPassword);
71         c.add(dpanel);c.add(pNewPassword);
72         c.add(emodel);c.add(fpanel);
73         setSize(500,500);
74         setLocation(100,100);
75         setResizable(false);
76         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
77         setVisible(true);
78     }
79     public void actionPerformed(ActionEvent e) {
80         if(e.getSource() == bChange) {
81             String vFirstName = tFirstName.getText();
82             String vCurrentPassword = pCurrentPassword.getText();
83             String vNewPassword = pNewPassword.getText();
84             String firstNamepattern = "[a-zA-Z]" ;
85             Scanner scan1 = new Scanner( vFirstName );
86             String matched1 = scan1.findInLine( firstNamepattern );
87             if ( matched1 == null ) {
88                 JOptionPane.showMessageDialog(this, "INVALID FIRST NAME\nIt should
contain only alphabets.", "Error", JOptionPane.ERROR_MESSAGE);
89                 tFirstName.setText("");
90             }
91             else
92                 flagFirstName = true;
93             String currentPasswordpattern = "[0-9]{4}";
94             Scanner scan2 = new Scanner( vCurrentPassword );
95             String matched2 = scan2.findInLine( currentPasswordpattern );
96             if ( matched2 == null ) {
97                 JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
98                 pCurrentPassword.setText("");
99             }
100             else
101                 flagCurrentPassword = true;
102             String newPasswordpattern = "[0-9]{4}" ;
103             Scanner scan3 = new Scanner( vNewPassword );
104             String matched3 = scan3.findInLine( currentPasswordpattern );
105             if ( matched3 == null ) {
106                 JOptionPane.showMessageDialog(this, "INVALID NEW PASSWORD\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
107                 pNewPassword.setText("");
108             }
109             else
110                 flagNewPassword = true;
111             if(flagFirstName == true) {
112                 if(flagCurrentPassword == true) {
113                     if(flagNewPassword == true) {
114                         try {
115                             FileInputStream fin=new FileInputStream("Reg.dat");
116                             ObjectInputStream oin=new ObjectInputStream(fin);
117                             list=(ArrayList<OperatorRegData>)oin.readObject();
118                         }catch(Exception e1) {
119                             JOptionPane.showMessageDialog(this, "No file found in
database", "Error", JOptionPane.ERROR_MESSAGE);
120                             list=new ArrayList<OperatorRegData>();
121                         }
122                         for(int i=0;i<list.size();i++) {

```

ChangePassword.java

```

123         if(list.get(i).getName().equals(tFirstName.getText()))
124         {
125             flagFirstNameMatch = true;
126             if(list.get(i).getPassword().equals(pCurrentPassword
127             d.getText())) {
128                 flagCurrentPasswordMatch = true;
129                 list.get(i).setPassword(pNewPassword.getText());
130             }
131         }
132     try {
133         FileOutputStream fout=new FileOutputStream("Reg.dat");
134         ObjectOutputStream oout=new ObjectOutputStream(fout);
135         oout.writeObject(list);
136     }catch(Exception e1){}
137     if((flagFirstNameMatch == true) &&
138     (flagCurrentPasswordMatch == true)) {
139         JOptionPane.showMessageDialog(this, "Password has been
140     successfully changed");
141         tFirstName.setText("");
142         pCurrentPassword.setText("");
143         pNewPassword.setText("");
144     }
145     if((flagFirstNameMatch == true) &&
146     (flagCurrentPasswordMatch == false)) {
147         JOptionPane.showMessageDialog(this, "Password
148     incorrect", "Error", JOptionPane.ERROR_MESSAGE);
149         pCurrentPassword.setText("");
150     }
151     if((flagFirstNameMatch == false) &&
152     (flagCurrentPasswordMatch == false)) {
153         JOptionPane.showMessageDialog(this, "Operator does not
154     exist.", "Error", JOptionPane.ERROR_MESSAGE);
155         tFirstName.setText("");
156         pCurrentPassword.setText("");
157         pNewPassword.setText("");
158     }
159 }
160 }
161 flagFirstName = false;
162 flagCurrentPassword = false;
163 flagNewPassword = false;
164 flagFirstNameMatch = false;
165 flagCurrentPasswordMatch = false;
166 }
167 if(e.getSource()==bBack) {
168     int rply = JOptionPane.showConfirmDialog(this, "Are you sure to
169     quit?");
170     if(rply == JOptionPane.YES_OPTION) {
171         new Userwindow("");
172         setVisible(false);
173     }
174 }
175 }
176 }

```

```
1 package test;
2 import java.io.Serializable;
3 public class Create implements Serializable {
4     private String ac;
5     private String name;
6     private String name1;
7     private String address;
8     private String email;
9     private String nationality;
10    private String acctype;
11    private String city;
12    private String gender;
13    private String dob;
14    private String date;
15    private String time;
16    private String Identity;
17    private String profession;
18    private String initialamnt;
19    public String getAc() {
20        return ac;
21    }
22    public void setAc(String ac) {
23        this.ac = ac;
24    }
25    public String getInitialamnt() {
26        return initialamnt;
27    }
28    public void setInitialamnt(String initialamnt) {
29        this.initialamnt = initialamnt;
30    }
31    public String getNationality() {
32        return nationality;
33    }
34    public void setNationality(String nationality) {
35        this.nationality = nationality;
36    }
37    public String getAcctype() {
38        return acctype;
39    }
40    public void setAcctype(String acctype) {
41        this.acctype = acctype;
42    }
43    public String getIdentity() {
44        return Identity;
45    }
46    public void setIdentity(String identity) {
47        Identity = identity;
48    }
49    public String getProfession() {
50        return profession;
51    }
52    public void setProfession(String profession) {
53        this.profession = profession;
54    }
55    public String getName1() {
56        return name1;
57    }
58    public void setName1(String name1) {
59        this.name1 = name1;
60    }
61    public String getEmail() {
62        return email;
```

```
63     }
64     public void setEmail(String email) {
65         this.email = email;
66     }
67     public String getName() {
68         return name;
69     }
70     public void setName(String name) {
71         this.name = name;
72     }
73     public String getAddress() {
74         return address;
75     }
76     public void setAddress(String address) {
77         this.address = address;
78     }
79     public String getCity() {
80         return city;
81     }
82     public void setCity(String city) {
83         this.city = city;
84     }
85     public String getGender() {
86         return gender;
87     }
88     public void setGender(String gender) {
89         this.gender = gender;
90     }
91     public String getDob() {
92         return dob;
93     }
94     public void setDob(String dob) {
95         this.dob = dob;
96     }
97     public String getDate() {
98         return date;
99     }
100    public void setDate(String date) {
101        this.date = date;
102    }
103    public String getTime() {
104        return time;
105    }
106    public void setTime(String time) {
107        this.time = time;
108    }
109 }
```

```

1 package test;
2 import java.awt.*;
3
4 public class CreateFrame extends JFrame implements ActionListener {
5     private JLabel
6         10,101,11,12,13,14,140,15,16,17,18,19,110,111,112,113,114,116,117,118,119,120;
7     private JTextField t1,tid1,tid2,tid3,tid4,tid5,tid6,tid7,tid8,tid10;
8     private JComboBox city,day,month,year,annualincome,identity,profession,acno;
9     private JRadioButton rmale,rfemale,rsav,rothers,married,unmarried;
10    private JButton bsubmit,back;
11    private String sysdate,systime;
12    private int count = 0;
13    private String ini;
14    private boolean flagName = false;
15    private boolean flagName1 = false;
16    private boolean flagAddress = false;
17    private boolean flagNationality = false;
18    private ArrayList<Create> alist;
19    private boolean flagAccountNumber = false;
20    private boolean flagInitialAmount = false;
21    private boolean flagLandline = false;
22    private boolean flagMobile = false;
23    private boolean flagAge = false;
24    private boolean flagProof = false;
25    CreateFrame() {
26        super("User registration form");
27        Container c=getContentPane();
28        c.setLayout(new GridLayout(21,2));
29        tid1=new JTextField();
30        tid2=new JTextField();
31        tid3=new JTextField();
32        tid4=new JTextField();
33        tid5=new JTextField();
34        tid6=new JTextField();
35        tid7=new JTextField();
36        tid8=new JTextField();
37        tid10=new JTextField();
38        String cvalue[]={ "<50000", "50000-100000", "100000-600000", ">600000"};
39        annualincome=new JComboBox(cvalue);
40        String dlvalue[]={ "Government Service", "Business", "Private
41        Sector", "Student", "Others"};
42        profession=new JComboBox(dlvalue);
43        String evalue[]={ "Pan Card", "Voter Card", "Ration Card", "Driving License"};
44        identity=new JComboBox(evalue);
45        rsav=new JRadioButton("Savings");
46        rothers=new JRadioButton("Savings");
47        ButtonGroup rlgroupp=new ButtonGroup();
48        rlgroupp.add(rsav);
49        JPanel kpanel=new JPanel();
50        kpanel.add(rsav);
51        rmale=new JRadioButton("Male");
52        rfemale=new JRadioButton("Female");
53        ButtonGroup rgroup=new ButtonGroup();
54        rgroup.add(rmale);
55        rgroup.add(rfemale);
56        JPanel gpanel=new JPanel();
57        married=new JRadioButton("Married");
58        unmarried=new JRadioButton("Single");
59        ButtonGroup r2group=new ButtonGroup();
60        r2group.add(married);
61        r2group.add(unmarried);
62        JPanel gopanel=new JPanel();
63        gopanel.add(married);

```

```

65     gpanel.add(unmarried);
66     gpanel.add(rmale);
67     gpanel.add(rfemale);
68     t1=new JTextField();
69     String dvalue[]=new String[31];
70     for(int i=0;i<=30;i++)
71         dvalue[i]=String.valueOf(i+1);
72     day=new JComboBox(dvalue);
73     String mvalue[]=new String[12];
74     for(int i=0;i<=11;i++)
75         mvalue[i]=String.valueOf(i+1);
76     month=new JComboBox(mvalue);
77     String yvalue[]=new String[25];
78     int cnt=0;
79     for(int i=1989;i<=2013;i++) {
80         yvalue[cnt]=String.valueOf(i);
81         cnt++;
82     }
83     year=new JComboBox(yvalue);
84     JPanel cpanel=new JPanel();
85     cpanel.add(day);
86     cpanel.add(month);
87     cpanel.add(year);
88     bsubmit=new JButton("Register");
89     bsubmit.addActionListener(this);
90     Font f1=new Font("Times New Roman",Font.BOLD,14);
91     l01=new JLabel("ACCOUNT OPENING");
92     l01.setFont(f1);
93     l01.setForeground(Color.RED);
94     JPanel fpanel=new JPanel();
95     fpanel.add(l01);
96     fpanel.setBackground(new Color(0,0,64));
97     l0=new JLabel("FORM");
98     l0.setFont(f1);
99     l0.setForeground(Color.RED);
100    JPanel apanel=new JPanel();
101    apanel.add(l0);
102    apanel.setBackground(new Color(0,0,64));
103    l1=new JLabel("Enter First Name");
104    l1.setFont(f1);
105    l1.setForeground(Color.RED);
106    JPanel bpanel=new JPanel();
107    bpanel.add(l1);
108    bpanel.setBackground(new Color(0,0,64));
109    l2=new JLabel("Enter Last Name");
110    l2.setFont(f1);
111    l2.setForeground(Color.RED);
112    JPanel copanel=new JPanel();
113    copanel.add(l2);
114    copanel.setBackground(new Color(0,0,64));
115    l3=new JLabel("Enter Address");
116    l3.setFont(f1);
117    l3.setForeground(Color.RED);
118    JPanel dpanel=new JPanel();
119    dpanel.add(l3);
120    dpanel.setBackground(new Color(0,0,64));
121    l4=new JLabel("Enter Phone Number(Landline)");
122    l4.setFont(f1);
123    l4.setForeground(Color.RED);
124    JPanel epanel=new JPanel();
125    epanel.add(l4);
126    epanel.setBackground(new Color(0,0,64));

```

CreateFrame.java

```

127      140=new JLabel("Enter Phone Number(Mobile)");
128      140.setFont(f1);
129      140.setForeground(Color.RED);
130      JPanel vpanel=new JPanel();
131      vpanel.add(140);
132      vpanel.setBackground(new Color(0,0,64));
133      15=new JLabel("Select Sex");
134      15.setFont(f1);
135      15.setForeground(Color.RED);
136      JPanel qpanel=new JPanel();
137      qpanel.add(15);
138      qpanel.setBackground(new Color(0,0,64));
139      16=new JLabel("Natinality");
140      16.setFont(f1);
141      16.setForeground(Color.RED);
142      JPanel wpanel=new JPanel();
143      wpanel.add(16);
144      wpanel.setBackground(new Color(0,0,64));
145      17=new JLabel("Select Age");
146      17.setFont(f1);
147      17.setForeground(Color.RED);
148      JPanel opanel=new JPanel();
149      opanel.add(17);
150      opanel.setBackground(new Color(0,0,64));
151      18=new JLabel("Select A/C Type");
152      18.setFont(f1);
153      18.setForeground(Color.RED);
154      JPanel zpanel=new JPanel();
155      zpanel.add(18);
156      zpanel.setBackground(new Color(0,0,64));
157      19=new JLabel("Date of Birth");
158      19.setFont(f1);
159      19.setForeground(Color.RED);
160      JPanel xpanel=new JPanel();
161      xpanel.add(19);
162      xpanel.setBackground(new Color(0,0,64));
163      110=new JLabel("Enter Annual Income");
164      110.setFont(f1);
165      110.setForeground(Color.RED);
166      JPanel vxpanel=new JPanel();
167      vxpanel.add(110);
168      vxpanel.setBackground(new Color(0,0,64));
169      111=new JLabel("Enter Profession");
170      111.setFont(f1);
171      111.setForeground(Color.RED);
172      JPanel vppanel=new JPanel();
173      vppanel.add(111);
174      vppanel.setBackground(new Color(0,0,64));
175      112=new JLabel("Enter Marital Status");
176      112.setFont(f1);
177      112.setForeground(Color.RED);
178      JPanel npanel=new JPanel();
179      npanel.add(112);
180      npanel.setBackground(new Color(0,0,64));
181      113=new JLabel("Enter Identity Proof");
182      113.setFont(f1);
183      113.setForeground(Color.RED);
184      JPanel mpanel=new JPanel();
185      mpanel.add(113);
186      mpanel.setBackground(new Color(0,0,64));
187      114=new JLabel("Enter Identity proof no:");
188      114.setFont(f1);

```


CreateFrame.java

```

189         l14.setForeground(Color.RED);
190         JPanel jpanel=new JPanel();
191         jpanel.add(l14);
192         jpanel.setBackground(new Color(0,0,64));
193         Calendar cal = Calendar.getInstance();
194         String cday = ""+cal.get(Calendar.DATE);
195         int x =cal.get(Calendar.MONTH);
196         String cmonth = ""+(x+1);
197         String cyear = ""+cal.get(Calendar.YEAR);
198         sysdate = cday+"/"+cmonth+"/"+cyear;
199         l16=new JLabel(sysdate);
200         String chr = ""+cal.get(Calendar.HOUR_OF_DAY);
201         String cmin = ""+cal.get(Calendar.MINUTE);
202         String csec = ""+cal.get(Calendar.SECOND);
203         systime = chr+":"+cmin+": "+csec;
204         l17=new JLabel(systime);
205         l18=new JLabel("Current Time");
206         l18.setFont(f1);
207         l18.setForeground(Color.RED);
208         JPanel vopanel=new JPanel();
209         vopanel.add(l18);
210         vopanel.setBackground(new Color(0,0,64));
211         l19=new JLabel("Current Date");
212         l19.setFont(f1);
213         l19.setForeground(Color.RED);
214         JPanel vipanel=new JPanel();
215         vipanel.add(l19);
216         vipanel.setBackground(new Color(0,0,64));
217         l20=new JLabel("Give a 4-digit Account number");
218         l20.setFont(f1);
219         l20.setForeground(Color.RED);
220         JPanel poppanel=new JPanel();
221         poppanel.add(l20);
222         poppanel.setBackground(new Color(0,0,64));
223         l20=new JLabel("Initial Amount");
224         l20.setFont(f1);
225         l20.setForeground(Color.RED);
226         JPanel kop=new JPanel();
227         kop.add(l20);
228         kop.setBackground(new Color(0,0,64));
229         bsubmit=new JButton("SUBMIT");
230         bsubmit.addActionListener(this);
231         JPanel dopanel=new JPanel();
232         dopanel.add(bsubmit);
233         dopanel.setBackground(new Color(0,0,64));
234         back=new JButton("BACK");
235         back.addActionListener(this);
236         JPanel dipanel=new JPanel();
237         dipanel.add(back);
238         dipanel.setBackground(new Color(0,0,64));
239         c.add(fpanel);c.add(apanel);
240         c.add(bpanel);c.add(tid1);
241         c.add(copanel);c.add(tid2);
242         c.add(dpanel);c.add(tid3);
243         c.add(epanel);c.add(tid4);
244         c.add(vpanel);c.add(tid5);
245         c.add(qpanel);c.add(gpanel);
246         c.add(wpanel);c.add(tid6);
247         c.add(opanel);c.add(tid7);
248         c.add(zpanel);c.add(kpanel);
249         c.add(xpanel);c.add(cpanel);
250         c.add(vxpanel);c.add(annualincome);

```

CreateFrame.java

```

251     c.add(vppanel);c.add(profession);
252     c.add(npanel);c.add(gopanel);
253     c.add(mpanel);c.add(identity);
254     c.add(jpanel);c.add(tid8);
255     c.add(poppanel);c.add(t1);
256     c.add(kop);c.add(tid10);
257     c.add(vipanel);c.add(l16);
258     c.add(vopanel);c.add(l17);
259     c.add(dopanel);c.add(dipanel);
260     setSize(600,625);
261     setLocation(200,200);
262     setResizable(false);
263     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
264     setVisible(true);
265 }
266 public void actionPerformed(ActionEvent e) {
267     String name;
268     String name1;
269     String address;
270     String nationality;
271     String acc;
272     String gn;
273     String dob;
274     String date;
275     String time;
276     String Iden;
277     String prof;
278     String inti;
279     String acl;
280     String land;
281     String mob;
282     String age;
283     String proof;
284     Create reg;
285     if(e.getSource()==bsubmit) {
286         name=tid1.getText();
287         name1=tid2.getText();
288         address=tid3.getText();
289         nationality=tid6.getText();
290         acl=t1.getText();
291         ini = tid10.getText();
292         land = tid4.getText();
293         mob = tid5.getText();
294         age = tid7.getText();
295         proof = tid8.getText();
296         String namepattern = "[A-Za-z]";
297         Scanner scan = new Scanner( name );
298         String matched = scan.findInLine( namepattern );
299         if ( matched == null ) {
300             JOptionPane.showMessageDialog(this, "INVALID FIRST NAME\nIt
should contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
301             tid1.setText("");
302         }
303         else
304             flagName = true;
305         String namepattern1 = "[A-Za-z]";
306         Scanner scan1 = new Scanner( name1 );
307         String matched1 = scan1.findInLine( namepattern1 );
308         if ( matched1 == null ) {
309             JOptionPane.showMessageDialog(this, "INVALID LAST NAME\nIt
should contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
310             tid2.setText("");

```

```

311     }
312     else
313         flagName1 = true;
314     String namepattern2 = "[A-Za-z0-9]{1}";
315     Scanner scan2 = new Scanner( address );
316     String matched2 = scan2.findInLine( namepattern2 );
317     if ( matched2 == null ) {
318         JOptionPane.showMessageDialog(this, "INVALID ADDRESS\nIT SHOLUD
NOT BLANK.", "Error", JOptionPane.ERROR_MESSAGE);
319         tid3.setText("");
320     }
321     else
322         flagAddress = true;
323     String landlinepattern = "[0-9]{8}";
324     Scanner scan6 = new Scanner( land );
325     String matched6 = scan6.findInLine( landlinepattern );
326     if ( matched6 == null ) {
327         JOptionPane.showMessageDialog(this, "INVALID LANDLINE NUMBER\nIt
should contain only digits with minimum length of 8 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
328         tid4.setText("");
329     }
330     else
331         flagLandline = true;
332     String mobpattern = "[0-9]{10}";
333     Scanner scan7 = new Scanner( mob );
334     String matched7 = scan7.findInLine( mobpattern );
335     if ( matched7 == null ) {
336         JOptionPane.showMessageDialog(this, "INVALID MOBILE NUMBER\nIt
should contain only digits with minimum length of 10 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
337         tid5.setText("");
338     }
339     else
340         flagMobile = true;
341     String namepattern3 = "[A-Za-z]";
342     Scanner scan3 = new Scanner( nationality );
343     String matched3 = scan3.findInLine( namepattern3 );
344     if ( matched3 == null ) {
345         JOptionPane.showMessageDialog(this, "INVALID NATIONALITY\nIt
should contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
346         tid6.setText("");
347     }
348     else
349         flagNationality = true;
350     String agepattern = "[0-9]{2}";
351     Scanner scan8 = new Scanner( age );
352     String matched8 = scan8.findInLine( agepattern );
353     if ( matched8 == null ) {
354         JOptionPane.showMessageDialog(this, "INVALID AGE\nAge must be
minimum 10 years & it should contain only digit.", "Error",
JOptionPane.ERROR_MESSAGE);
355         tid7.setText("");
356     }
357     else
358         flagAge = true;
359     String proofpattern2 = "[A-Za-z0-9]{5}";
360     Scanner scan9 = new Scanner( proof );
361     String matched9 = scan9.findInLine( proofpattern2 );
362     if ( matched9 == null ) {
363         JOptionPane.showMessageDialog(this, "INVALID IDENTITY PROOF
NUMBER\nIT SHOLUD CONTAIN MINIMUM 5 ALPHABET OR DIGIT.", "Error",

```

```

JOptionPane.ERROR_MESSAGE);
364         tid8.setText("");
365     }
366     else
367         flagProof = true;
368     String accpattern = "[0-9]{4}" ;
369     Scanner scan4 = new Scanner( ac1 ) ;
370     String matched4 = scan4.findInLine( accpattern ) ;
371     if ( matched4 == null ) {
372         JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
373         t1.setText("");
374     }
375     else
376         flagAccountNumber = true;
377     String inipattern = "[0-9]{3}" ;
378     Scanner scan5 = new Scanner( ini ) ;
379     String matched5 = scan5.findInLine( inipattern ) ;
380     if ( matched5 == null ) {
381         JOptionPane.showMessageDialog(this, "INVALID INITIAL AMOUNT\nIt
should contain only digits with minimum amount of INR 500.", "Error",
JOptionPane.ERROR_MESSAGE);
382         tid10.setText("");
383     }
384     else
385         flagInitialAmount = true;
386     if(flagName == true) {
387         if(flagName1 == true) {
388             if(flagAddress == true) {
389                 if(flagNationality == true) {
390                     if(flagAccountNumber == true) {
391                         if(flagInitialAmount == true) {
392                             if(flagLandline == true) {
393                                 if(flagMobile == true) {
394                                     if(flagAge == true) {
395                                         if(flagProof == true) {
396                                             try {
397                             FileInputStream fin=new
FileInputStream("Regis.dat");
398                             ObjectInputStream oin=new
ObjectInputStream(fin);
399                             alist=(ArrayList<Create>)oin.readObject();
400                             }catch(Exception ex) {
401                                 alist=new
ArrayList<Create>();
402                             }
403                             for(Create element : alist) {
404                                 if(element.getAc().equals(t1.getText()))
count++;
405                             }
406                             if(count > 0) {
407                                 JOptionPane.showMessageDialog(this, "Account numner is already registered.\nTry
another one", "Error", JOptionPane.ERROR_MESSAGE);
409                                 t1.setText("");
410                             }
411                             else {
412                                 if(Integer.parseInt(ini)
>= 500) {

```

CreateFrame.java

```

413
414
415
416
    address=tid3.getText();
417
    nationality=tid6.getText();
418
419
    (String)identity.getSelectedItemAt();
420
    (String)profession.getSelectedItemAt();
421
422
423
424
    if(rothers.isSelected())
425
426
427
428
429
    if(rfemale.isSelected())
430
431
    (String)day.getSelectedItemAt();
432
    (String)month.getSelectedItemAt();
433
    (String)year.getSelectedItemAt();
434
    y;
435
436
437
438
        reg.setAddress(address);
439
        reg.setNationality(nationality);
440
441
442
443
444
        reg.setProfession(prof);
445
446
        reg.setInitialamnt(inti);
447
        reg.setDate(l16.getText());
448
        reg.setTime(l17.getText());
449
        con=JOptionPane.showConfirmDialog(this, "Are You Sure to Register?");
450
        if(con==JOptionPane.YES_OPTION) {
451
            AddInformation1(reg);
452
            Actotalcreate(Integer.parseInt(t1.getText()),0,Integer.parseInt(tid10.getText()),Integer.parseInt(tid10.getText()));
453

```

```

    inti=tid10.getText();
    name=tid1.getText();
    namel=tid2.getText();

    ac1=t1.getText();
    Iden=

    prof=

    acc="";
    if(rsav.isSelected())
        acc="Savings";
    else
        acc="Others";
    gn="";
    if(rmale.isSelected())
        gn="Male";
    else
        gn="Female";
    String d=

    String m=

    String y=

    dob=d + "-" + m + "-" +

    reg=new Create();
    reg.setName(name);
    reg.setNamel(namel);

    reg.setAcctype(acc);
    reg.setGender(gn);
    reg.setDob(dob);
    reg.setIdentity(Iden);

    reg.setAc(ac1);

    int

```

```

TotalUpdate(Integer.parseInt(t1.getText()), Integer.parseInt(tid10.getText()));
454
        JOptionPane.showMessageDialog(this, "Successfully
registered");
455
        new
        Individualpassbook(ac1);
456
        tid10.setText("");
457
        tid1.setText("");
458
        tid2.setText("");
459
        tid3.setText("");
460
        tid6.setText("");
461
        t1.setText("");
462
        tid4.setText("");
463
        tid5.setText("");
464
        tid7.setText("");
465
        tid8.setText("");
466
    }
467
    }
468
    else {
469
        JOptionPane.showMessageDialog(this, "Minimum initial amount should be 500",
"Error", JOptionPane.ERROR_MESSAGE);
470
        tid10.setText("");
471
    }
472
    }
473
    }
474
    }
475
    }
476
    }
477
    }
478
    }
479
    }
480
    }
481
    }
482
    }
483
    flagName = false;
484
    flagName1 = false;
485
    flagAddress = false;
486
    flagNationality = false;
487
    flagAccountNumber = false;
488
    flagInitialAmount = false;
489
    count =0;
490
    flagLandline = false;
491
    flagMobile = false;
492
    flagAge = false;
493
    flagProof = false;
494
}
495
if(e.getSource()==back) {
496
    int rply = JOptionPane.showConfirmDialog(this, "Are you sure to
quit?");
497
    if(rply == JOptionPane.YES_OPTION) {
498
        new Userwindow("");
499
        setVisible(false);
500
    }
501
}
502
}
503
}
504

```

```

1 package test;
2 import java.io.Serializable;
4 public class Depo extends JFrame implements Serializable {
5     private String savingsacc;
6     private String fixeddepoacc;
7     private String savingsaccbal;
8     private String fixedamnt;
9     private String terms;
10    private String rate;
11    private String date;
12    private String time;
13    public String getDate() {
14        return date;
15    }
16    public void setDate(String date) {
17        this.date = date;
18    }
19    public String getTime() {
20        return time;
21    }
22    public void setTime(String time) {
23        this.time = time;
24    }
25    public String getSavingsacc() {
26        return savingsacc;
27    }
28    public void setSavingsacc(String savingsacc) {
29        this.savingsacc = savingsacc;
30    }
31    public String getFixeddepoacc() {
32        return fixeddepoacc;
33    }
34    public void setFixeddepoacc(String fixeddepoacc) {
35        this.fixeddepoacc = fixeddepoacc;
36    }
37    public String getSavingsaccbal() {
38        return savingsaccbal;
39    }
40    public void setSavingsaccbal(String savingsaccbal) {
41        this.savingsaccbal = savingsaccbal;
42    }
43    public String getFixedamnt() {
44        return fixedamnt;
45    }
46    public void setFixedamnt(String fixedamnt) {
47        this.fixedamnt = fixedamnt;
48    }
49    public String getTerms() {
50        return terms;
51    }
52    public void setTerms(String terms) {
53        this.terms = terms;
54    }
55    public String getRate() {
56        return rate;
57    }
58    public void setRate(String rate) {
59        this.rate = rate;
60    }
61 }
62

```

```

1 package test;
2 import java.util.*;
3
4
5 public class DepoCreate extends JFrame {
6     private String sysdate, systime;
7     DepoCreate(String a,String b,String c,String d,String e,String f) {
8         Calendar cal = Calendar.getInstance();
9         String cday = ""+cal.get(Calendar.DATE);
10        int x =cal.get(Calendar.MONTH);
11        String cmonth = ""+(x+1);
12        String cyear = ""+cal.get(Calendar.YEAR);
13        sysdate = cday+"/"+cmonth+"/"+cyear;
14        String chr = ""+cal.get(Calendar.HOUR_OF_DAY);
15        String cmin = ""+cal.get(Calendar.MINUTE);
16        String csec = ""+cal.get(Calendar.SECOND);
17        systime = chr+":"+cmin+": "+csec;
18        Depo sdl=new Depo();
19        sdl.setSavingsacc(a);
20        sdl.setFixeddepoacc(b);
21        sdl.setSavingsaccbal(c);
22        sdl.setFixedamnt(d);
23        sdl.setTerms(e);
24        sdl.setRate(f);
25        sdl.setDate(sysdate);
26        sdl.setTime(systime);
27        ArrayList<Depo> list4;
28        try {
29            FileInputStream fin=new FileInputStream("fixed.dat");
30            ObjectInputStream oin=new ObjectInputStream(fin);
31            list4=(ArrayList<Depo>)oin.readObject();
32        }catch(Exception e1) {
33            list4=new ArrayList<Depo>();
34        }
35        list4.add(sdl);
36        try {
37            FileOutputStream fout=new FileOutputStream("fixed.dat");
38            ObjectOutputStream oout=new ObjectOutputStream(fout);
39            oout.writeObject(list4);
40        }catch(Exception e1){}
41    }
42 }

```


Deposit.java

```

1 package test;
2 import java.awt.event.*;
3
4 public class Deposit extends JFrame implements ActionListener {
5     private JLabel l1,l2,l3;
6     private JTextField t1;
7     private JButton submit,cancel;
8     public Deposit() {
9         Container c=getContentPane();
10        c.setLayout(new GridLayout(3,2));
11        Font f1=new Font("Times New Roman",Font.BOLD,20);
12        l1=new JLabel("DEPOSIT");
13        l1.setFont(f1);
14        l1.setForeground(Color.GRAY);
15        JPanel fpanel=new JPanel();
16        fpanel.add(l1);
17        fpanel.setBackground(new Color(0,0,64));
18        l1=new JLabel("SLIP");
19        l1.setFont(f1);
20        l1.setForeground(Color.GRAY);
21        JPanel apanel=new JPanel();
22        apanel.add(l1);
23        apanel.setBackground(new Color(0,0,64));
24        l1=new JLabel("Enter ACCOUNT NO:");
25        l1.setFont(f1);
26        l1.setForeground(Color.GRAY);
27        JPanel bpanel=new JPanel();
28        bpanel.add(l1);
29        bpanel.setBackground(new Color(0,0,64));
30        t1=new JTextField();
31        submit=new JButton("SUBMIT");
32        submit.addActionListener(this);
33        JPanel dpanel=new JPanel();
34        dpanel.add(submit);
35        dpanel.setBackground(new Color(0,0,64));
36        cancel=new JButton("CANCEL");
37        cancel.addActionListener(this);
38        JPanel epanel=new JPanel();
39        epanel.add(cancel);
40        epanel.setBackground(new Color(0,0,64));
41        c.add(fpanel);c.add(apanel);
42        c.add(bpanel);c.add(t1);
43        c.add(dpanel);c.add(epanel);
44        setSize(500,400);
45        setLocation(200,200);
46        setResizable(false);
47        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48        setVisible(true);
49    }
50    public void actionPerformed(ActionEvent e) {
51        if(e.getSource()==submit) {
52            new Deposit1();
53            setVisible(false);
54        }
55        if(e.getSource()==cancel) {
56            new Userwindow("");
57            setVisible(false);
58        }
59    }
60 }
61 }

```

Deposit1.java

```

1 package test;
2 import java.awt.event.*;
3
4 public class Deposit1 extends JFrame implements ActionListener {
5     private JLabel l1,l2,l3,l4,l5;
6     private JTextField t1,t2,t3;
7     private JButton proceed,back;
8     public Deposit1() {
9         Container c=getContentPane();
10        c.setLayout(new GridLayout(5,2));
11        Font f1=new Font("Times New Roman",Font.BOLD,20);
12        l1=new JLabel("WITHDRAWAL");
13        l1.setFont(f1);
14        l1.setForeground(Color.WHITE);
15        JPanel fpanel=new JPanel();
16        fpanel.add(l1);
17        fpanel.setBackground(new Color(0,0,64));
18        l2=new JLabel("DETAILS");
19        l2.setFont(f1);
20        l2.setForeground(Color.WHITE);
21        JPanel apanel=new JPanel();
22        apanel.add(l2);
23        apanel.setBackground(new Color(0,0,64));
24        l3=new JLabel("Account Holder Name");
25        l3.setFont(f1);
26        l3.setForeground(Color.GRAY);
27        JPanel bpanel=new JPanel();
28        bpanel.add(l3);
29        bpanel.setBackground(new Color(0,0,64));
30        l4=new JLabel("BALANCE IN ACCOUNT");
31        l4.setFont(f1);
32        l4.setForeground(Color.GRAY);
33        JPanel cpanel=new JPanel();
34        cpanel.add(l4);
35        cpanel.setBackground(new Color(0,0,64));
36        l5=new JLabel("Amount To Be Deposited");
37        l5.setFont(f1);
38        l5.setForeground(Color.GRAY);
39        JPanel dpanel=new JPanel();
40        dpanel.add(l5);
41        dpanel.setBackground(new Color(0,0,64));
42        t1=new JTextField();
43        t2=new JTextField();
44        t3=new JTextField();
45        proceed=new JButton("PROCEED");
46        proceed.addActionListener(this);
47        JPanel epanel=new JPanel();
48        epanel.add(proceed);
49        epanel.setBackground(new Color(0,0,64));
50        back=new JButton("BACK");
51        back.addActionListener(this);
52        JPanel gpanel=new JPanel();
53        gpanel.add(back);
54        gpanel.setBackground(new Color(0,0,64));
55        c.add(fpanel);c.add(apanel);
56        c.add(bpanel);c.add(t1);
57        c.add(cpanel);c.add(t2);
58        c.add(dpanel);c.add(t3);
59        c.add(epanel);c.add(gpanel);
60        setSize(500,400);
61        setLocation(200,200);
62        setResizable(false);
63        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64    }

```

Deposit1.java

```
65     setVisible(true);
66 }
67 public void actionPerformed(ActionEvent e) {
68     if(e.getSource() == proceed) {
69         int con = JOptionPane.showConfirmDialog(this, "Are You Sure to Deposit?");
70         if(con == JOptionPane.YES_OPTION) {
71             setVisible(false);
72         }
73     }
74     if(e.getSource() == back) {
75         new Deposit();
76         setVisible(false);
77     }
78 }
79 }
```

DiaplayAll.java

```

1 package test;
2 import javax.swing.*;
6 public class DiaplayAll extends JFrame {
7     public DiaplayAll() {
8         super("DISPLAY All USERS INFORMATION");
9         String heading[]={"First Name","Last name","Address","Nationality","Account
Type","Date Of Birth","Identity","Profession","Reg Date","Reg Time","A/c
no","Balance"};
10        String data[][];
11        ArrayList<Create> list;
12        try {
13            FileInputStream fin=new FileInputStream("Regis.dat");
14            ObjectInputStream oin=new ObjectInputStream(fin);
15            list=(ArrayList<Create>)oin.readObject();
16            data = new String[list.size()][heading.length+1];
17            int r=0;
18            for(Create re : list) {
19                data[r][0]=re.getName();
20                data[r][1]=re.getName1();
21                data[r][2]=re.getAddress();
22                data[r][3]=re.getNationality();
23                data[r][4]=re.getAcctype();
24                data[r][5]=re.getDob();
25                data[r][6]=re.getIdentity();
26                data[r][7]=re.getProfession();
27                data[r][8]=re.getDate();
28                data[r][9]=re.getTime();
29                data[r][10]=re.getAc();
30                data[r][11]=re.getInitialamnt();
31                r++;
32            }
33            Container con=getContentPane();
34            con.setLayout(new BorderLayout());
35            JTable datatable=new JTable(data, heading);
36            JScrollPane jsp=new JScrollPane(datatable);
37            con.add(new JLabel("All Registration Details"),BorderLayout.NORTH);
38            con.add(jsp,BorderLayout.CENTER);
39            setSize(650, 300);
40            setLocation(200, 200);
41            setVisible(true);
42        } catch (Exception e) {
43            JOptionPane.showMessageDialog(this, "No file found in data base",
"Error", JOptionPane.ERROR_MESSAGE);
44        }
45    }
46 }
47

```

Dispwithdraw.java

```
1 package test;
2 import javax.swing.*;
3
4 public class Dispwithdraw extends JFrame {
5     String heading[]={ "First Name", "Last name", "Address", "Nationality", "Account
        Type", "Date Of Birth", "Identity", "Profession", "Reg Date", "Reg Time", "A/c
        no", "Initial amnt" };
6     public Dispwithdraw(String b[][]) {
7         Container con=getContentPane();
8         con.setLayout(new BorderLayout());
9         JTable datatable=new JTable(b, heading);
10        JScrollPane jsp=new JScrollPane(datatable);
11        con.add(new JLabel("Registration Details"), BorderLayout.NORTH);
12        con.add(jsp, BorderLayout.CENTER);
13        setSize(650, 650);
14        setLocation(600, 200);
15        setVisible(true);
16    }
17 }
```

Individualpassbook.java

```

1 package test;
2 import javax.swing.*;
6 public class Individualpassbook extends JFrame {
7     String heading[]={ "Accno", "Withdraw", "Deposit", "Total", "Date", "Time"};
8     String data[][];
9     ArrayList<Actotal> list;
10    public Individualpassbook(String name){
11        super("Search for " + name);
12        int acc=Integer.parseInt(name);
13        try {
14            FileInputStream fin=new FileInputStream("actotal.dat");
15            ObjectInputStream oin=new ObjectInputStream(fin);
16            list=(ArrayList<Actotal>)oin.readObject();
17        }catch(Exception e1) {
18            JOptionPane.showMessageDialog(this, "No file found in data base",
19            "Error", JOptionPane.ERROR_MESSAGE);
20        }
21        int row=0;
22        for(Actotal re : list) {
23            if(re.getAccount()==(acc)){
24                row++;
25            }
26        }
27        data = new String[row][heading.length+1];
28        int r=0;
29        for(Actotal re : list) {
30            if(re.getAccount()==(acc)) {
31                data[r][0]=String.valueOf(re.getAccount());
32                data[r][1]=String.valueOf(re.getWithdrawl());
33                data[r][2]=String.valueOf(re.getDepo());
34                data[r][3]=String.valueOf(re.getTotal());
35                data[r][4]=re.getDate();
36                data[r][5]=re.getTime();
37                r++;
38            }
39        }
40        Container con=getContentPane();
41        con.setLayout(new BorderLayout());
42        JTable datatable=new JTable(data,heading);
43        JScrollPane jsp=new JScrollPane(datatable);
44        con.add(new JLabel("PASSBOOK DETAILS"),BorderLayout.NORTH);
45        con.add(jsp,BorderLayout.CENTER);
46        setSize(650, 300);
47        setLocation(200,200);
48        setVisible(true);
49 }

```

```

1 package test;
2 import java.awt.event.*;
3
4 class FirstWindow extends JFrame implements ActionListener {
5     private JLabel l0,l1,l2,l3;
6     private JButton admin,user;
7     public FirstWindow(String title) {
8         super("WELCOME TO UNITED BANK OF PARTNERS");
9         Container c=getContentPane();
10        c.setLayout(new GridLayout(3,2));
11        Font f1=new Font("Times New Roman",Font.BOLD,25);
12        l0=new JLabel("WELCOME TO UNITED");
13        l0.setFont(f1);
14        l0.setForeground(Color.WHITE);
15        JPanel fpanel=new JPanel();
16        fpanel.add(l0);
17        fpanel.setBackground(new Color(0,0,64));
18        l1=new JLabel("BANK OF PARTNERS");
19        l1.setFont(f1);
20        l1.setForeground(Color.WHITE);
21        JPanel spanel=new JPanel();
22        spanel.add(l1);
23        spanel.setBackground(new Color(0,0,64));
24        l3=new JLabel("AS:");
25        l3.setFont(f1);
26        l3.setForeground(Color.WHITE);
27        JPanel xpanel=new JPanel();
28        xpanel.add(l3);
29        xpanel.setBackground(new Color(0,0,64));
30        Font f2=new Font("Times New Roman",Font.BOLD,25);
31        l2=new JLabel("LOGIN");
32        l2.setFont(f2);
33        l2.setForeground(Color.WHITE);
34        JPanel ppanel=new JPanel();
35        ppanel.add(l2);
36        ppanel.setBackground(new Color(0,0,64));
37        admin=new JButton("ADMINISTRATOR");
38        JPanel gpanel=new JPanel();
39        gpanel.add(admin);
40        gpanel.setBackground(new Color(0,0,64));
41        admin.addActionListener(this);
42        gpanel.add(new JLabel(""));
43        user=new JButton("OPERATOR");
44        JPanel qpanel=new JPanel();
45        qpanel.add(user);
46        qpanel.setBackground(new Color(0,0,64));
47        user.addActionListener(this);
48        qpanel.add(user);
49        c.add(fpanel);c.add(spanel);
50        c.add(ppanel);c.add(xpanel);
51        c.add(gpanel);c.add(qpanel);
52        setSize(600,450);
53        setLocation(200,200);
54        setResizable(false);
55        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
56        setVisible(true);
57    }
58    public void actionPerformed(ActionEvent e) {
59        if(e.getSource()==admin) {
60            new SecondWindow();
61            setVisible(false);
62        }
63        if(e.getSource()==user) {

```

Main.java

```
65         new ThirdWindow() ;
66         setVisible(false);
67     }
68 }
69 }
70 public class Main {
71     public static void main(String[] args) {
72         new FirstWindow("WELCOME TO UNITED BANK OF PARTNERS");
73     }
74 }
75
```



```

1 package test;
2 import java.awt.event.*;
6
7 class SecondWindow extends JFrame implements ActionListener {
8     private JLabel l0,l1,l2,l3;
9     private JTextField t1;
10    private JPasswordField p1;
11    private JButton login, cancel;
12    private String sysDate, sysTime;
13    private boolean flagID = false;
14    private boolean flagPass = false;
15    public SecondWindow() {
16        super("Login as Administrator");
17        Container c=getContentPane();
18        c.setLayout(new GridLayout(4,2));
19        Font f1=new Font("Times New Roman",Font.BOLD,20);
20        l1=new JLabel("ADMINISTRATOR");
21        l1.setFont(f1);
22        l1.setForeground(Color.RED);
23        JPanel fpanel=new JPanel();
24        fpanel.add(l1);
25        fpanel.setBackground(new Color(0,0,64));
26        l0=new JLabel("LOGIN");
27        l0.setFont(f1);
28        l0.setForeground(Color.RED);
29        JPanel epanel=new JPanel();
30        epanel.add(l0);
31        epanel.setBackground(new Color(0,0,64));
32        l2=new JLabel("USERNAME");
33        l2.setFont(f1);
34        l2.setForeground(Color.RED);
35        JPanel apanel=new JPanel();
36        apanel.add(l2);
37        apanel.setBackground(new Color(0,0,64));
38        l3=new JLabel("PASSWORD");
39        l3.setFont(f1);
40        l3.setForeground(Color.RED);
41        JPanel bpanel=new JPanel();
42        bpanel.add(l3);
43        bpanel.setBackground(new Color(0,0,64));
44        t1=new JTextField();
45        p1=new JPasswordField();
46        login=new JButton("LOGIN");
47        login.addActionListener(this);
48        JPanel cpanel=new JPanel();
49        cpanel.add(login);
50        cpanel.setBackground(new Color(0,0,64));
51        cancel=new JButton("CANCEL");
52        cancel.addActionListener(this);
53        JPanel dpanel=new JPanel();
54        dpanel.add(cancel);
55        dpanel.setBackground(new Color(0,0,64));
56        Calendar cal = Calendar.getInstance();
57        String cday = ""+cal.get(Calendar.DATE);
58        int x =cal.get(Calendar.MONTH);
59        String cmonth = ""+(x+1);
60        String cyear = ""+cal.get(Calendar.YEAR);
61        sysDate = cday+"/"+cmonth+"/"+cyear;
62        String chr = ""+cal.get(Calendar.HOUR_OF_DAY);
63        String cmin = ""+cal.get(Calendar.MINUTE);
64        String csec = ""+cal.get(Calendar.SECOND);
65        sysTime = chr+":"+cmin+": "+csec;

```

```

66         c.add(fpanel);c.add(epanel);
67         c.add(apanel);c.add(t1);
68         c.add(bpanel);c.add(p1);
69         c.add(cpanel);c.add(dpanel);
70         setSize(500,425);
71         setLocation(200,200);
72         setResizable(false);
73         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
74         setVisible(true);
75     }
76     public void actionPerformed(ActionEvent e) {
77         if(e.getSource()==login) {
78             String n = t1.getText().trim();
79             String ps = p1.getText().trim();
80             String namepattern = "[A-Za-z]";
81             Scanner scan = new Scanner( n );
82             String matched = scan.findInLine( namepattern );
83             if ( matched == null ) {
84                 JOptionPane.showMessageDialog(this, "INVALID USER NAME\nIt should
contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
85                 t1.setText("");
86             }
87             else {
88                 flagID = true;
89             }
90             String passwordpattern = "[0-9]{4}";
91             Scanner scan2 = new Scanner( ps );
92             String matched2 = scan2.findInLine( passwordpattern );
93             if ( matched2 == null ) {
94                 JOptionPane.showMessageDialog(this, "INVALID PASSWORD\nIt should
contain only digits with minimum password length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
95                 p1.setText("");
96             }
97             else
98                 flagPass = true;
99             if(flagID == true) {
100                 if(flagPass == true) {
101                     if(t1.getText().equals("admin") && p1.getText().equals("1234"))
102 {
103                         JOptionPane.showMessageDialog(this, "Welcome
Administrator");
104                         new Admin();
105                         setVisible(false);
106                     }
107                     else {
108                         JOptionPane.showMessageDialog(this, "Wrong
username/password...\nTry again", "Error", JOptionPane.ERROR_MESSAGE);
109                         t1.setText("");
110                         p1.setText("");
111                     }
112                 }
113                 flagID = false;
114                 flagPass = false;
115             }
116             if(e.getSource()==cancel) {
117                 int con=JOptionPane.showConfirmDialog(this, "Are you sure to cancel?");
118                 if(con==JOptionPane.YES_OPTION) {
119                     new FirstWindow("");
120                     setVisible(false);
121                 }

```

Main1.java

```
122     }  
123 }  
124 }  
125 public class Main1 {  
126     public static void main(String[] args) {}  
127 }  
128
```

OperatorAddInfo.java

```
1 package test;
2 import java.io.*;
4 public class OperatorAddInfo {
5     ArrayList<OperatorRegData> list4;
6     public OperatorAddInfo(OperatorRegData rg1) {
7         try {
8             FileInputStream fin=new FileInputStream("Reg.dat");
9             ObjectInputStream oin=new ObjectInputStream(fin);
10            list4=(ArrayList<OperatorRegData>)oin.readObject();
11        } catch (Exception e) {
12            list4=new ArrayList<OperatorRegData>();
13        }
14        list4.add(rg1);
15        try {
16            FileOutputStream fout=new FileOutputStream("Reg.dat");
17            ObjectOutputStream oout=new ObjectOutputStream(fout);
18            oout.writeObject(list4);
19        } catch (Exception e){}
20    }
21 }
22
```

OperatorDelete.java

```

1 package test;
2 import java.awt.event.*;
6 public class OperatorDelete extends JFrame implements ActionListener {
7     private JLabel l1,l2,l3;
8     private JTextField t1;
9     private JButton submit,back;
10    private boolean flag = false;
11    public OperatorDelete() {
12        super("Operator Delete");
13        Container c=getContentPane();
14        c.setLayout(new GridLayout(3,2));
15        Font f1=new Font("Times New Roman",Font.BOLD,20);
16        l1=new JLabel("OPERATOR");
17        l1.setFont(f1);
18        l1.setForeground(Color.RED);
19        JPanel fpanel=new JPanel();
20        fpanel.add(l1);
21        fpanel.setBackground(new Color(0,0,64));
22        l2=new JLabel("Delete");
23        l2.setFont(f1);
24        l2.setForeground(Color.RED);
25        JPanel apanel=new JPanel();
26        apanel.add(l2);
27        apanel.setBackground(new Color(0,0,64));
28        l3=new JLabel("Give OPERATOR First name");
29        l3.setFont(f1);
30        l3.setForeground(Color.RED);
31        JPanel bpanel=new JPanel();
32        bpanel.add(l3);
33        bpanel.setBackground(new Color(0,0,64));
34        t1=new JTextField();
35        submit=new JButton("Delete");
36        submit.addActionListener(this);
37        JPanel cpanel=new JPanel();
38        cpanel.add(submit);
39        cpanel.setBackground(new Color(0,0,64));
40        back=new JButton("BACK");
41        back.addActionListener(this);
42        JPanel dpanel=new JPanel();
43        dpanel.add(back);
44        dpanel.setBackground(new Color(0,0,64));
45        c.add(fpanel);c.add(apanel);
46        c.add(bpanel);c.add(t1);
47        c.add(cpanel);c.add(dpanel);
48        setSize(550,300);
49        setLocation(200,200);
50        setResizable(false);
51        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
52        setVisible(true);
53    }
54    public void actionPerformed(ActionEvent e) {
55        if(e.getSource() == submit) {
56            String n = t1.getText().trim();
57            String namepattern = "[A-Za-z]";
58            Scanner scan = new Scanner(n);
59            String matched = scan.findInLine(namepattern);
60            if (matched == null) {
61                JOptionPane.showMessageDialog(this, "INVALID OPERATOR NAME\nIt
should starts with alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
62                t1.setText("");
63            }
64            else

```

OperatorDelete.java

```
65         flag = true;
66         if(flag == true) {
67             String sname = t1.getText().trim();
68             new OperatorDeleteDisplay(sname);
69             t1.setText("");
70         }
71         flag = false;
72     }
73     if(e.getSource()==back) {
74         int con=JOptionPane.showConfirmDialog(this, "Are You Sure you want to
cancel?");
75         if(con==JOptionPane.YES_OPTION) {
76             new Admin();
77             setVisible(false);
78         }
79     }
80 }
81 }
82 }
```

OperatorDeleteDisplay.java

```
1 package test;
2 import java.io.*;
3
4
5 public class OperatorDeleteDisplay extends JFrame {
6     ArrayList<OperatorRegData> list1;
7     public OperatorDeleteDisplay(String name) {
8         try {
9             FileInputStream fin=new FileInputStream("Reg.dat");
10            ObjectInputStream oin=new ObjectInputStream(fin);
11            list1=(ArrayList<OperatorRegData>)oin.readObject();
12            int r=0;
13            for(int i=0;i<list1.size();i) {
14                if(list1.get(i).getName().equals(name)) {
15                    list1.remove(i);
16                    r++;
17                }
18                else
19                    i++;
20            }
21            if(r==0)
22                JOptionPane.showMessageDialog(this, "Match not Found");
23            else {
24                try {
25                    FileOutputStream fout=new FileOutputStream("Reg.dat");
26                    ObjectOutputStream oout=new ObjectOutputStream(fout);
27                    oout.writeObject(list1);
28                }catch(Exception e){}
29                JOptionPane.showMessageDialog(this, "Successfully deleted");
30            }
31        }
32        catch(Exception e) {
33            JOptionPane.showMessageDialog(this, "No file found in data base",
34            "Error", JOptionPane.ERROR_MESSAGE);
35        }
36 }
37
38
```

OperatorDisplay.java

```

1 package test;
2 import javax.swing.*;
6 public class OperatorDisplay extends JFrame {
7     public OperatorDisplay() {
8         super("DISPLAY All OPERATORS INFORMATION");
9         String heading[]={"First Name", "Last
name", "Password", "Address", "City", "Gender", "Date Of Birth", "Reg Date", "Reg Time"};
10        String data[][];
11        ArrayList<OperatorRegData> list;
12        try {
13            FileInputStream fin=new FileInputStream("Reg.dat");
14            ObjectInputStream oin=new ObjectInputStream(fin);
15            list=(ArrayList<OperatorRegData>)oin.readObject();
16            data = new String[list.size()][heading.length+1];
17            int r=0;
18            for(OperatorRegData re : list) {
19                data[r][0]=re.getName();
20                data[r][1]=re.getName1();
21                data[r][2]=re.getPassword();
22                data[r][3]=re.getAddress();
23                data[r][4]=re.getCity();
24                data[r][5]=re.getGender();
25                data[r][6]=re.getDob();
26                data[r][7]=re.getDate();
27                data[r][8]=re.getTime();
28                r++;
29            }
30            Container con=getContentPane();
31            con.setLayout(new BorderLayout());
32            JTable datatable=new JTable(data, heading);
33            JScrollPane jsp=new JScrollPane(datatable);
34            con.add(new JLabel("All Registration Details"), BorderLayout.NORTH);
35            con.add(jsp, BorderLayout.CENTER);
36            setSize(650, 300);
37            setLocation(200, 200);
38            setVisible(true);
39        } catch (Exception e) {
40            JOptionPane.showMessageDialog(this, "No file found in data base",
"Error", JOptionPane.ERROR_MESSAGE);
41        }
42    }
43 }

```


OperatorReg.java

```

1 package test;
2 import java.awt.*;
3
4 public class OperatorReg extends JFrame implements ActionListener {
5     private JLabel l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l1;
6     private JTextField tid,tod;
7     private JPasswordField tpass;
8     private JTextArea tadd;
9     private JComboBox city,day,month,year,annualincome;
10    private JCheckBox cmo,cmu,cco;
11    private JRadioButton rmale,rfemale;
12    private JButton bsubmit,back;
13    private String sysdate,systime;
14    private boolean flagName=false, flagName1 = false, flagPass=false, flagAdd =
15    false;
16    private ArrayList<OperatorRegData> alist = new ArrayList<OperatorRegData>();
17    public OperatorReg(String title) {
18        super("Operator registration");
19        Container c=getContentPane();
20        c.setLayout(new GridLayout(12,2));
21        tid=new JTextField();
22        tod=new JTextField();
23        tpass=new JPasswordField();
24        tadd=new JTextArea(5,20);
25        JScrollPane tapan=new JScrollPane(tadd);
26        String cvalue[]={ "<50000", "50000-100000", "100000-600000", ">600000" };
27        annualincome=new JComboBox(cvalue);
28        String
29        ctvalue[]={ "Kolkata", "Delhi", "Mumbai", "Chennai", "Hydrabad", "Pune", "Bangalore" };
30        city=new JComboBox(ctvalue);
31        rmale=new JRadioButton("Male");
32        rfemale=new JRadioButton("Female");
33        ButtonGroup rgroup=new ButtonGroup();
34        rgroup.add(rmale);
35        rgroup.add(rfemale);
36        JPanel gpanel=new JPanel();
37        gpanel.add(rmale);
38        gpanel.add(rfemale);
39        String dvalue[]=new String[31];
40        for(int i=0;i<=30;i++)
41            dvalue[i]=String.valueOf(i+1);
42        day=new JComboBox(dvalue);
43        String mvalue[]=new String[12];
44        for(int i=0;i<=11;i++)
45            mvalue[i]=String.valueOf(i+1);
46        month=new JComboBox(mvalue);
47        String yvalue[]=new String[25];
48        int cnt=0;
49        for(int i=1989;i<=2013;i++) {
50            yvalue[cnt]=String.valueOf(i);
51            cnt++;
52        }
53        year=new JComboBox(yvalue);
54        JPanel cpanel=new JPanel();
55        cpanel.add(day);
56        cpanel.add(month);
57        cpanel.add(year);
58        bsubmit=new JButton("Register");
59        bsubmit.addActionListener(this);
60        back=new JButton("BACK");
61        back.addActionListener(this);
62        Font f=new Font("TIMES NEW ROMAN",Font.BOLD,27);
63        l1=new JLabel("OPERATOR");

```

```

64     l1.setFont(f);
65     l1.setForeground(Color.BLUE);
66     JPanel apanel=new JPanel();
67     apanel.add(l1);
68     apanel.setBackground(new Color(0,0,64));
69     l0=new JLabel("REGISTRATION");
70     l0.setFont(f);
71     l0.setForeground(Color.BLUE);
72     JPanel bpanel=new JPanel();
73     bpanel.add(l0);
74     bpanel.setBackground(new Color(0,0,64));
75     Font f1=new Font("comic sans ms",Font.BOLD,14);
76     l2=new JLabel("Enter First Name");
77     l2.setFont(f1);
78     l2.setForeground(Color.RED);
79     JPanel dpanel=new JPanel();
80     dpanel.add(l2);
81     dpanel.setBackground(new Color(0,0,64));
82     l=new JLabel("Enter Last Name");
83     l.setFont(f1);
84     l.setForeground(Color.RED);
85     JPanel qpanel=new JPanel();
86     qpanel.add(l);
87     qpanel.setBackground(new Color(0,0,64));
88     l3=new JLabel("Enter Password");
89     l3.setFont(f1);
90     l3.setForeground(Color.RED);
91     JPanel epanel=new JPanel();
92     epanel.add(l3);
93     epanel.setBackground(new Color(0,0,64));
94     l4=new JLabel("Enter Address");
95     l4.setFont(f1);
96     l4.setForeground(Color.RED);
97     JPanel fpanel=new JPanel();
98     fpanel.add(l4);
99     fpanel.setBackground(new Color(0,0,64));
100    l5=new JLabel("Select City");
101    l5.setFont(f1);
102    l5.setForeground(Color.RED);
103    JPanel hpanel=new JPanel();
104    hpanel.add(l5);
105    hpanel.setBackground(new Color(0,0,64));
106    l6=new JLabel("Select Gender");
107    l6.setFont(f1);
108    l6.setForeground(Color.RED);
109    JPanel ipanel=new JPanel();
110    ipanel.add(l6);
111    ipanel.setBackground(new Color(0,0,64));
112    l7=new JLabel("Select DOB");
113    l7.setFont(f1);
114    l7.setForeground(Color.RED);
115    JPanel jpanel=new JPanel();
116    jpanel.add(l7);
117    jpanel.setBackground(new Color(0,0,64));
118    l8=new JLabel("Salary");
119    l8.setFont(f1);
120    l8.setForeground(Color.RED);
121    JPanel kpanel=new JPanel();
122    kpanel.add(l8);
123    kpanel.setBackground(new Color(0,0,64));
124    l9=new JLabel("Current Date");
125    l9.setFont(f1);

```

```

126         l9.setForeground(Color.RED);
127         JPanel npanel=new JPanel();
128         npanel.add(l9);
129         npanel.setBackground(new Color(0,0,64));
130         Calendar cal = Calendar.getInstance();
131         String cday = ""+cal.get(Calendar.DATE);
132         int x =cal.get(Calendar.MONTH);
133         String cmonth = ""+(x+1);
134         String cyear = ""+cal.get(Calendar.YEAR);
135         sysdate = cday+"/"+cmonth+"/"+cyear;
136         l11=new JLabel(sysdate);
137         String chr = ""+cal.get(Calendar.HOUR_OF_DAY);
138         String cmin = ""+cal.get(Calendar.MINUTE);
139         String csec = ""+cal.get(Calendar.SECOND);
140         systime = chr+":"+cmin+": "+csec;
141         l12=new JLabel(systime);
142         l10=new JLabel("Current Time");
143         l10.setFont(f1);
144         l10.setForeground(Color.RED);
145         JPanel opanel=new JPanel();
146         opanel.add(l10);
147         opanel.setBackground(new Color(0,0,64));
148         c.add(apanel);c.add(bpanel);
149         c.add(dpanel);c.add(tid);
150         c.add(qpanel);c.add(tod);
151         c.add(epanel);c.add(tpass);
152         c.add(fpanel);c.add(tapan);
153         c.add(hpanel);c.add(city);
154         c.add(ipanel);c.add(gpanel);
155         c.add(jpanel);c.add(cpanel);
156         c.add(kpanel);c.add(annualincome);
157         c.add(npanel);c.add(l11);
158         c.add(opanel);c.add(l12);
159         c.add(bsubmit);c.add(back);
160         setSize(500, 425);
161         setLocation(600,200);
162         setResizable(false);
163         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
164         setVisible(true);
165     }
166     public void actionPerformed(ActionEvent e) {
167         String name,name1,pass,add,ct,gn,dob,ins;
168         OperatorRegData reg;
169         if(e.getSource() == bsubmit) {
170             String n = tid.getText().trim();
171             String n1 = tod.getText().trim();
172             String ps = tpass.getText().trim()tpass.trim();
173             String ad = tadd.getText().trim();
174             String namepattern = "[A-Za-z]";
175             Scanner scan = new Scanner( n );
176             String matched = scan.findInLine( namepattern );
177             if ( matched == null ) {
178                 JOptionPane.showMessageDialog(this, "INVALID FIRST NAME\nIt
should contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
179                 tid.setText("");
180             }
181             else
182                 flagName = true;
183             String namepattern1 = "[A-Za-z]";
184             Scanner scan1 = new Scanner( n1 );
185             String matched1 = scan1.findInLine( namepattern1 );
186             if ( matched1 == null ) {

```

```

187         JOptionPane.showMessageDialog(this, "INVALID LAST NAME\nIt should
contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
188         tod.setText("");
189     }
190     else
191         flagName1 = true;
192     String passwordpattern = "[0-9]{4}";
193     Scanner scan2 = new Scanner( ps );
194     String matched2 = scan2.findInLine( passwordpattern );
195     if ( matched2 == null ) {
196         JOptionPane.showMessageDialog(this, "INVALID PASSWORD\nIt should
contain only digits with minimum password length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
197         tpass.setText("");
198     }
199     else
200         flagPass = true;
201     String addpattern = "[A-Za-z0-9]{1}";
202     Scanner scan3 = new Scanner( ad );
203     String matched3 = scan3.findInLine( addpattern );
204     if ( matched3 == null ) {
205         JOptionPane.showMessageDialog(this, "INVALID ADDRESS\nIt should not
blank.", "Error", JOptionPane.ERROR_MESSAGE);
206         tadd.setText("");
207     }
208     else
209         flagAdd = true;
210     name = tid.getText().trim();
211     name1 = tod.getText().trim();
212     pass = tpass.getText().trim();
213     add = tadd.getText().trim();
214     ct = (String) city.getSelectedItem();
215     gn = "";
216     if (rmale.isSelected())
217         gn = "Male";
218     else if (rfemale.isSelected())
219         gn = "Female";
220     String d = (String) day.getSelectedItem();
221     String m = (String) month.getSelectedItem();
222     String y = (String) year.getSelectedItem();
223     dob = d + "-" + m + "-" + y;
224     if (flagName == true) {
225         if (flagName1 == true) {
226             if (flagPass == true) {
227                 if (flagAdd == true) {
228                     reg = new OperatorRegData();
229                     reg.setName(name);
230                     reg.setName1(name1);
231                     reg.setPassword(pass);
232                     reg.setAddress(add);
233                     reg.setCity(ct);
234                     reg.setGender(gn);
235                     reg.setDob(dob);
236                     reg.setDate(l11.getText());
237                     reg.setTime(l12.getText());
238                     alist.add(reg);
239                     int con = JOptionPane.showConfirmDialog(this, "Are You
Sure to Register?");
240                     if (con == JOptionPane.YES_OPTION) {
241                         new OperatorAddInfo(reg);
242                         JOptionPane.showMessageDialog(this, "Successfully
registered");

```

OperatorReg.java

```

243         tid.setText("");
244         tod.setText("");
245         tadd.setText("");
246         tpass.setText("");
247     }
248 }
249 }
250 }
251 }
252     flagName = false;
253     flagName1 = false;
254     flagPass = false;
255     flagAdd = false;
256 }
257 if(e.getSource()==back) {
258     int con=JOptionPane.showConfirmDialog(this, "Are You Sure to Quit?");
259     if(con==JOptionPane.YES_OPTION) {
260         new Admin();
261         setVisible(false);
262     }
263 }
264 }
265 }

```

```
1 package test;
2 import java.io.Serializable;
3 public class OperatorRegData implements Serializable {
4     private String name;
5     private String name1;
6     private String password;
7     private String address;
8     private String city;
9     private String gender;
10    private String salary;
11    private String dob;
12    private String date;
13    private String time;
14    public String getName1() {
15        return name1;
16    }
17    public void setName1(String name1) {
18        this.name1 = name1;
19    }
20    public String getSalary() {
21        return salary;
22    }
23    public void setSalary(String salary) {
24        this.salary = salary;
25    }
26    public String getName() {
27        return name;
28    }
29    public void setName(String name) {
30        this.name = name;
31    }
32    public String getPassword() {
33        return password;
34    }
35    public void setPassword(String password) {
36        this.password = password;
37    }
38    public String getAddress() {
39        return address;
40    }
41    public void setAddress(String address) {
42        this.address = address;
43    }
44    public String getCity() {
45        return city;
46    }
47    public void setCity(String city) {
48        this.city = city;
49    }
50    public String getGender() {
51        return gender;
52    }
53    public void setGender(String gender) {
54        this.gender = gender;
55    }
56    public String getDob() {
57        return dob;
58    }
59    public void setDob(String dob) {
60        this.dob = dob;
61    }
62    public String getDate() {
```

```
63         return date;
64     }
65     public void setDate(String date) {
66         this.date = date;
67     }
68     public String getTime() {
69         return time;
70     }
71     public void setTime(String time) {
72         this.time = time;
73     }
74 }
```

OperatorSearch.java

```

1 package test;
2 import java.awt.event.*;
7 public class OperatorSearch extends JFrame implements ActionListener {
8     private JLabel l1,l2,l3;
9     private JTextField t1;
10    private JButton submit,back;
11    private boolean flag = false;
12    public OperatorSearch() {
13        super("Operator Search");
14        Container c=getContentPane();
15        c.setLayout(new GridLayout(3,2));
16        Font fl=new Font("Times New Roman",Font.BOLD,20);
17        l1=new JLabel("OPERATOR");
18        l1.setFont(fl);
19        l1.setForeground(Color.RED);
20        JPanel fpanel=new JPanel();
21        fpanel.add(l1);
22        fpanel.setBackground(new Color(0,0,64));
23        l2=new JLabel("SEARCH");
24        l2.setFont(fl);
25        l2.setForeground(Color.RED);
26        JPanel apanel=new JPanel();
27        apanel.add(l2);
28        apanel.setBackground(new Color(0,0,64));
29        l3=new JLabel("Give OPERATOR First name");
30        l3.setFont(fl);
31        l3.setForeground(Color.RED);
32        JPanel bpanel=new JPanel();
33        bpanel.add(l3);
34        bpanel.setBackground(new Color(0,0,64));
35        t1=new JTextField();
36        submit=new JButton("SUBMIT");
37        submit.addActionListener(this);
38        JPanel cpanel=new JPanel();
39        cpanel.add(submit);
40        cpanel.setBackground(new Color(0,0,64));
41        back=new JButton("BACK");
42        back.addActionListener(this);
43        JPanel dpanel=new JPanel();
44        dpanel.add(back);
45        dpanel.setBackground(new Color(0,0,64));
46        c.add(fpanel);c.add(apanel);
47        c.add(bpanel);c.add(t1);
48        c.add(cpanel);c.add(dpanel);
49        setSize(550,300);
50        setLocation(200,200);
51        setResizable(false);
52        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
53        setVisible(true);
54    }
55    public void actionPerformed(ActionEvent e) {
56        if(e.getSource() == submit) {
57            String n = t1.getText().trim();
58            String namepattern = "[A-Za-z]";
59            Scanner scan = new Scanner(n);
60            String matched = scan.findInLine(namepattern);
61            if (matched == null) {
62                JOptionPane.showMessageDialog(this, "INVALID OPERATOR NAME\nIt
should starts with alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
63                t1.setText("");
64            }
65            else

```


OperatorSearch.java

```
66         flag = true;
67         if(flag == true) {
68             String sname = t1.getText().trim();
69             new OperatorSearchDisplay(sname);
70             t1.setText("");
71         }
72         flag = false;
73     }
74     if(e.getSource() == back) {
75         int con = JOptionPane.showConfirmDialog(this, "Are you sure to cancel?");
76         if(con == JOptionPane.YES_OPTION) {
77             new Admin();
78             setVisible(false);
79         }
80     }
81 }
82 }
```

OperatorSearchDisplay.java

```

1 package test;
2 import java.awt.*;
3
4 public class OperatorSearchDisplay extends JFrame {
5     String data[][];
6     String heading[]={ "First Name", "Last
7     name", "Password", "Address", "City", "Gender", "Date Of Birth", "Reg Date", "Reg Time"};
8     ArrayList<OperatorRegData> list;
9     private boolean flagIn;
10    public OperatorSearchDisplay(String name) {
11        super("Search for user account number" + name);
12        try {
13            FileInputStream fin=new FileInputStream("Reg.dat");
14            ObjectInputStream oin=new ObjectInputStream(fin);
15            list=(ArrayList<OperatorRegData>)oin.readObject();
16            int row=0;
17            for(OperatorRegData re : list) {
18                if(re.getName().equals(name)) {
19                    row++;
20                }
21            }
22            data = new String[row][heading.length+1];
23            int r=0;
24            for(OperatorRegData re : list) {
25                if(re.getName().equals(name)) {
26                    data[r][0]=re.getName();
27                    data[r][1]=re.getName1();
28                    data[r][2]=re.getPassword();
29                    data[r][3]=re.getAddress();
30                    data[r][4]=re.getCity();
31                    data[r][5]=re.getGender();
32                    data[r][6]=re.getDob();
33                    data[r][7]=re.getDate();
34                    data[r][8]=re.getTime();
35                    flagIn = true;
36                }
37            }
38            if(flagIn == true) {
39                Container con=getContentPane();
40                con.setLayout(new BorderLayout());
41                JTable datatable=new JTable(data, heading);
42                JScrollPane jsp=new JScrollPane(datatable);
43                con.add(new JLabel("Registration Details for " +
44                name),BorderLayout.NORTH);
45                con.add(jsp,BorderLayout.CENTER);
46                setSize(650, 300);
47                setLocation(200, 200);
48                setVisible(true);
49            }
50            else
51                JOptionPane.showMessageDialog(this, "Match not Found");
52            flagIn = false;
53        } catch (Exception e) {
54            JOptionPane.showMessageDialog(this, "No file found in data base",
55            "Error", JOptionPane.ERROR_MESSAGE);
56        }
57    }
58 }

```

OperatorUserSearch.java

```

1 package test;
2 import java.awt.event.*;
6 public class OperatorUserSearch extends JFrame implements ActionListener {
7     private JLabel l1,l2,l3;
8     private JTextField t1;
9     private JButton submit,back;
10    private boolean flag = false;
11    public OperatorUserSearch() {
12        super("User Search");
13        Container c=getContentPane();
14        c.setLayout(new GridLayout(3,2));
15        Font f1=new Font("Times New Roman",Font.BOLD,20);
16        l1=new JLabel("CUSTOMER");
17        l1.setFont(f1);
18        l1.setForeground(Color.RED);
19        JPanel fpanel=new JPanel();
20        fpanel.add(l1);
21        fpanel.setBackground(new Color(0,0,64));
22        l2=new JLabel("SEARCH");
23        l2.setFont(f1);
24        l2.setForeground(Color.RED);
25        JPanel apanel=new JPanel();
26        apanel.add(l2);
27        apanel.setBackground(new Color(0,0,64));
28        l3=new JLabel("Give User A/C no");
29        l3.setFont(f1);
30        l3.setForeground(Color.RED);
31        JPanel bpanel=new JPanel();
32        bpanel.add(l3);
33        bpanel.setBackground(new Color(0,0,64));
34        t1=new JTextField();
35        submit=new JButton("SUBMIT");
36        submit.addActionListener(this);
37        JPanel cpanel=new JPanel();
38        cpanel.add(submit);
39        cpanel.setBackground(new Color(0,0,64));
40        back=new JButton("BACK");
41        back.addActionListener(this);
42        JPanel dpanel=new JPanel();
43        dpanel.add(back);
44        dpanel.setBackground(new Color(0,0,64));
45        c.add(fpanel);c.add(apanel);
46        c.add(bpanel);c.add(t1);
47        c.add(cpanel);c.add(dpanel);
48        setSize(550,300);
49        setLocation(200,200);
50        setResizable(false);
51        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
52        setVisible(true);
53    }
54    public void actionPerformed(ActionEvent e) {
55        if(e.getSource() == submit) {
56            String n = t1.getText().trim();
57            String namepattern = "[0-9]{4}";
58            Scanner scan = new Scanner(n);
59            String matched = scan.findInLine(namepattern);
60            if (matched == null) {
61                JOptionPane.showMessageDialog(this, "INVALID OPERATOR NAME\nIt
should contain only digit.", "Error", JOptionPane.ERROR_MESSAGE);
62                t1.setText("");
63            }
64            else

```

OperatorUserSearch.java

```
65         flag = true;
66         if(flag == true) {
67             String sname = t1.getText().trim();
68             new UserSearchDisplay(sname);
69             t1.setText("");
70         }
71         flag = false;
72     }
73     if(e.getSource()==back) {
74         int con=JOptionPane.showConfirmDialog(this, "Are you sure to cancel?");
75         if(con==JOptionPane.YES_OPTION) {
76             new Userwindow("");
77             setVisible(false);
78         }
79     }
80 }
81 }
```

```

1 package test;
2 import javax.swing.*;
3
4 public class PassBook extends JFrame implements ActionListener {
5     private JLabel l1;
6     private JTextField tid;
7     private JButton bdisplay, cancel;
8     private boolean flagAccNum = false;
9     private ArrayList<Actotal> list4;
10    private boolean flagAccMatch = false;
11    public PassBook() {
12        super("Indivisual Passbook Display");
13        Container c=getContentPane();
14        c.setLayout(new GridLayout(2,2));
15        tid=new JTextField();
16        bdisplay=new JButton("SUBMIT");
17        bdisplay.addActionListener(this);
18        JPanel apanel=new JPanel();
19        apanel.add(bdisplay);
20        apanel.setBackground(new Color(0,0,64));
21        cancel=new JButton("CANCEL");
22        cancel.addActionListener(this);
23        JPanel bpanel=new JPanel();
24        bpanel.add(cancel);
25        bpanel.setBackground(new Color(0,0,64));
26        Font fl=new Font("comic sans ms",Font.BOLD,14);
27        l1=new JLabel("Enter A/C no");
28        l1.setFont(fl);
29        l1.setForeground(Color.RED);
30        JPanel cpanel=new JPanel();
31        cpanel.add(l1);
32        cpanel.setBackground(new Color(0,0,64));
33        c.add(cpanel);c.add(tid);
34        c.add(apanel);c.add(bpanel);
35        setSize(400,175);
36        setLocation(600,200);
37        setResizable(false);
38        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39        setVisible(true);
40    }
41    public void actionPerformed(ActionEvent e) {
42        if(e.getSource()==bdisplay) {
43            String vAccNum = tid.getText();
44            String accNumpattern = "[0-9]{4}" ;
45            Scanner scanl = new Scanner( vAccNum );
46            String matched1 = scanl.findInLine( accNumpattern );
47            if ( matched1 == null ) {
48                JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
49                should contain only digits with minimum length of 4 digit.", "Error",
50                JOptionPane.ERROR_MESSAGE);
51                tid.setText("");
52            }
53            else
54                flagAccNum = true;
55            try {
56                FileInputStream fin=new FileInputStream("actotal.dat");
57                ObjectInputStream oin=new ObjectInputStream(fin);
58                list4=(ArrayList<Actotal>)oin.readObject();
59            }catch(Exception ex) {
60                JOptionPane.showMessageDialog(this, "No file found in data base",
61                "Error", JOptionPane.ERROR_MESSAGE);
62            }
63            if(flagAccNum == true) {

```

```

65         for(Actotal re : list4) {
66             if(String.valueOf(re.getAccount()).equals(tid.getText())) {
67                 new Individualpassbook(tid.getText());
68                 tid.setText("");
69                 flagAccMatch = true;
70             }
71         }
72         if(flagAccMatch == false) {
73             JOptionPane.showMessageDialog(this, "Account number does not
74 exist.", "Error", JOptionPane.ERROR_MESSAGE);
75             tid.setText("");
76             flagAccMatch = false;
77         }
78         flagAccNum = false;
79     }
80     if(e.getSource()==cancel) {
81         int con=JOptionPane.showConfirmDialog(this, "Are you sure to cancel?");
82         if(con==JOptionPane.YES_OPTION) {
83             new Userwindow("");
84             setVisible(false);
85         }
86     }
87 }
88 }

```

Passdisp.java

```

1 package test;
2 import javax.swing.*;
6 public class Passdisp extends JFrame {
7     public Passdisp(String accNumber) {
8         super("Passbook");
9         String heading[]={"Accno","Withdraw","Deposit","Total","Date","Time"};
10        String data[][];//=new String[1000][6];
11        ArrayList<Actotal> list=new ArrayList<Actotal>();
12        try {
13            FileInputStream fin=new FileInputStream("actotal.dat");
14            ObjectInputStream oin=new ObjectInputStream(fin);
15            list=(ArrayList<Actotal>)oin.readObject();
16        } catch (Exception e) {
17            JOptionPane.showMessageDialog(this, "No file found in data base",
18            "Error", JOptionPane.ERROR_MESSAGE);
19        }
20        int row=0;
21        for(Actotal re : list) {
22            if(String.valueOf(re.getAccount()).equals(accNumber))
23                row++;
24        }
25        data = new String[row][heading.length+1];
26        int r=0,i=0;
27        for(Actotal re : list) {
28            if(String.valueOf(re.getAccount()).equals(accNumber)) {
29                data[r][0]=String.valueOf(re.getAccount());
30                data[r][1]=String.valueOf(re.getWithdrawl());
31                data[r][2]=String.valueOf(re.getDepo());
32                data[r][3]=String.valueOf(re.getTotal());
33                data[r][4]=re.getDate();
34                data[r][5]=re.getTime();
35                r++;
36            }
37        }
38        Container con=getContentPane();
39        con.setLayout(new BorderLayout());
40        JTable datatable=new JTable(data,heading);
41        JScrollPane jsp=new JScrollPane(datatable);
42        con.add(new JLabel("All Registration Details"),BorderLayout.NORTH);
43        con.add(jsp,BorderLayout.CENTER);
44        setSize(650, 300);
45        setLocation(200, 200);
46        setVisible(true);
47 }

```

SavingsUpdate.java

```
1 package test;
2 import java.io.*;
4 public class SavingsUpdate {
5     public SavingsUpdate(int acc,int total) {
6         ArrayList<Depo> list4;
7         String ac,amt;
8         ac = String.valueOf(acc);
9         amt = String.valueOf(total);
10        try {
11            FileInputStream fin=new FileInputStream("fixed.dat");
12            ObjectInputStream oin=new ObjectInputStream(fin);
13            list4=(ArrayList<Depo>)oin.readObject();
14        } catch (Exception e) {
15            list4=new ArrayList<Depo>();
16        }
17        for(int i = 0 ;i<list4.size();i++) {
18            if(list4.get(i).getSavingsacc().equals(ac)) {
19                list4.get(i).setSavingsaccbal(amt);
20            }
21        }
22        try {
23            FileOutputStream fout=new FileOutputStream("fixed.dat");
24            ObjectOutputStream oout=new ObjectOutputStream(fout);
25            oout.writeObject(list4);
26        } catch (Exception e){}
27    }
28 }
29
```


TotalUpdate.java

```
1 package test;
2 import java.io.*;
4 public class TotalUpdate {
5     public TotalUpdate(int acc,int total) {
6         ArrayList<Create> list4;
7         String ac,amt;
8         ac = String.valueOf(acc);
9         amt = String.valueOf(total);
10        try {
11            FileInputStream fin=new FileInputStream("Regis.dat");
12            ObjectInputStream oin=new ObjectInputStream(fin);
13            list4=(ArrayList<Create>)oin.readObject();
14        } catch (Exception e) {
15            list4=new ArrayList<Create>();
16        }
17        for(int i = 0 ;i<list4.size();i++) {
18            if(list4.get(i).getAc().equals(ac)) {
19                list4.get(i).setInitialamt(amt);
20            }
21        }
22        try {
23            FileOutputStream fout=new FileOutputStream("Regis.dat");
24            ObjectOutputStream oout=new ObjectOutputStream(fout);
25            oout.writeObject(list4);
26        } catch (Exception e){}
27    }
28 }
```

Transfer.java

```

1 package test;
2 import java.awt.*;
3
4 public class Transfer extends JFrame implements ActionListener {
5     private JLabel l1,l2,l3;
6     private JTextField tid1,tid2,tid3;
7     private JButton bsubmit,bBack;
8     private int r=0,sourceamt,destamt,tran,t,acc,d=0,w=0;
9     ArrayList<Create> list1;
10    private boolean flagSource = false;
11    private boolean flagDestination = false;
12    private boolean flagAmount = false;
13    private boolean flagSourceMatch = false;
14    private boolean flagDestinationMatch = false;
15    private boolean flagSourceAmountMatch = false;
16    private boolean flagSourceInsufficientBalance = false;
17    public Transfer() {
18        super("Money Transfer");
19        tid1=new JTextField();
20        tid2=new JTextField();
21        tid3=new JTextField();
22        Container c=getContentPane();
23        c.setLayout(new GridLayout(4,2));
24        bsubmit=new JButton("Submit");
25        bsubmit.addActionListener(this);
26        JPanel epanel=new JPanel();
27        epanel.add(bsubmit);
28        epanel.setBackground(new Color(0,0,64));
29        bBack = new JButton("Back");
30        bBack.addActionListener(this);
31        JPanel fpanel=new JPanel();
32        fpanel.add(bBack);
33        fpanel.setBackground(new Color(0,0,64));
34        Font f1=new Font("comic sans ms",Font.BOLD,14);
35        l1=new JLabel("Enter Source A/C no");
36        l1.setFont(f1);
37        l1.setForeground(Color.RED);
38        JPanel gpanel=new JPanel();
39        gpanel.add(l1);
40        gpanel.setBackground(new Color(0,0,64));
41        l2=new JLabel("Enter Destination A/C no");
42        l2.setFont(f1);
43        l2.setForeground(Color.RED);
44        JPanel apanel=new JPanel();
45        apanel.add(l2);
46        apanel.setBackground(new Color(0,0,64));
47        l3=new JLabel("Enter Amount to be Transferred");
48        l3.setFont(f1);
49        l3.setForeground(Color.RED);
50        JPanel bpanel=new JPanel();
51        bpanel.add(l3);
52        bpanel.setBackground(new Color(0,0,64));
53        c.add(gpanel);c.add(tid1);
54        c.add(apanel);c.add(tid2);
55        c.add(bpanel);c.add(tid3);
56        c.add(epanel);c.add(fpanel);
57        setSize(500, 425);
58        setLocation(600,200);
59        setResizable(false);
60        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
61        setVisible(true);
62    }
63    public void actionPerformed(ActionEvent e) {

```

Transfer.java

```

67     String source,dest;
68     source=tid1.getText();
69     dest=tid2.getText();
70     if(e.getSource()==bsubmit) {
71         String vSource = tid1.getText();
72         String vDestination = tid2.getText();
73         String vAmount = tid3.getText();
74         String sourcepattern = "[0-9]{4}" ;
75         Scanner scan = new Scanner( vSource) ;
76         String matched = scan.findInLine( sourcepattern ) ;
77         if ( matched == null ) {
78             JOptionPane.showMessageDialog(this, "INVALID SOURCE ACCOUNT
NUMBER\nIt should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
79             tid1.setText("");
80         }
81         else
82             flagSource = true;
83         String destinationpattern = "[0-9]{4}" ;
84         Scanner scan1 = new Scanner( vDestination) ;
85         String matched1 = scan1.findInLine( destinationpattern ) ;
86         if ( matched1 == null ) {
87             JOptionPane.showMessageDialog(this, "INVALID DESTINATION ACCOUNT
NUMBER\nIt should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
88             tid2.setText("");
89         }
90         else
91             flagDestination = true;
92         String amountwpattern = "[0-9]{3}" ;
93         Scanner scan2 = new Scanner( vAmount ) ;
94         String matched2 = scan2.findInLine( amountwpattern ) ;
95         if ( matched2 == null ) {
96             JOptionPane.showMessageDialog(this, "INVALID AMOUNT\nIt should
contain only digits with minimum amount of INR 100.", "Error",
JOptionPane.ERROR_MESSAGE);
97             tid3.setText("");
98         }
99         else
100             flagAmount = true;
101         if(flagSource == true) {
102             if(flagDestination == true) {
103                 if(flagAmount == true) {
104                     try {
105                         FileInputStream fin=new FileInputStream("Regis.dat");
106                         ObjectInputStream oin=new ObjectInputStream(fin);
107                         list1=(ArrayList<Create>)oin.readObject();
108                     }
109                     catch(Exception e1){}
110                     for(Create re : list1) {
111                         if(re.getAc().equals(source)) {
112                             flagSourceMatch = true;
113                             for(Create re2 : list1) {
114                                 if(re2.getAc().equals(dest)) {
115                                     flagDestinationMatch = true;
116                                     sourceamnt=Integer.parseInt(re.getInitialam
nt());
117                                     destamnt=Integer.parseInt(re2.getInitialamn
t());
118                                     if(sourceamnt>500) {
119                                         flagSourceAmountMatch = true;
120                                         tran=Integer.parseInt(tid3.getText());

```

Transfer.java

```

121                                     t=sourceamnt-tran;
122                                     if((t>0)&&(t>=500)) {
123                                         flagSourceInsufficientBalance =
true;
124                                         acc=Integer.parseInt(tid1.getText())
);
125                                         new TotalUpdate(acc,t);
126                                         new Acttotalcreate(acc,tran,d,t);
127                                         t=destamnt+tran;
128                                         acc=Integer.parseInt(tid2.getText())
);
129                                         new TotalUpdate(acc,t);
130                                         new Acttotalcreate(acc,w,tran,t);
131                                         r++;
132                                     }
133                                 }
134                             }
135                         }
136                     }
137                 }
138                 if(r==1) {
139                     JOptionPane.showMessageDialog(this, "MONEY TRANSFERED
SUCCESSFULLY");
140                     tid1.setText("");
141                     tid2.setText("");
142                     tid3.setText("");
143                     JOptionPane.showMessageDialog(this, "Showing both
account holders passbook.");
144                     new Individualpassbook(source);
145                     new Individualpassbook(dest);
146                 }
147                 if(flagSourceMatch == false) {
148                     JOptionPane.showMessageDialog(this, "Source account
does not match", "Source account error", JOptionPane.ERROR_MESSAGE);
149                     tid1.setText("");
150                 }
151                 if(flagSourceMatch == true && flagDestinationMatch == true
&& flagSourceAmountMatch == false) {
152                     JOptionPane.showMessageDialog(this, "Source account
does not have minimum balance of INR 500", "Insufficient balance error",
JOptionPane.ERROR_MESSAGE);
153                     tid1.setText("");
154                     tid2.setText("");
155                     tid3.setText("");
156                 }
157                 else if(flagSourceMatch == true && flagDestinationMatch ==
true && flagSourceAmountMatch == true && flagSourceInsufficientBalance == false) {
158                     JOptionPane.showMessageDialog(this, "Insufficient
balance of source account to transfer money.", "nsufficient balance error",
JOptionPane.ERROR_MESSAGE);
159                     tid3.setText("");
160                 }
161                 if(flagSourceMatch == true && flagDestinationMatch ==
false) {
162                     JOptionPane.showMessageDialog(this, "Destination
account does not match", "Destination account error", JOptionPane.ERROR_MESSAGE);
163                     tid2.setText("");
164                 }
165             }
166         }
167     }
168     flagSource = false;

```

Transfer.java

```
169         flagDestination = false;
170         flagAmount = false;
171         flagSourceMatch = false;
172         flagDestinationMatch = false;
173         flagSourceAmountMatch = false;
174         flagSourceInsufficientBalance = false;
175         r=0;
176     }
177     if(e.getSource() == bBack) {
178         int rply = JOptionPane.showConfirmDialog(this, "Are you sure to
quit?");
179         if(rply == JOptionPane.YES_OPTION) {
180             new Userwindow("");
181             setVisible(false);
182         }
183     }
184 }
185 }
```

```

1 package test;
2 import java.awt.event.*;
3 class ThirdWindow extends JFrame implements ActionListener {
4     private JLabel l0,l1,l2,l3;
5     private JTextField t1;
6     private JPasswordField p1;
7     private JButton login,back;
8     private boolean flagID = false;
9     private boolean flagPass = false;
10    private ArrayList<OperatorRegData> alistTemp;
11    private boolean flag = false;
12    public ThirdWindow() {
13        super("Log in as operator");
14        Container c=getContentPane();
15        c.setLayout(new GridLayout(4,2));
16        Font f1=new Font("Times New Roman",Font.BOLD,28);
17        l0=new JLabel("OPERATOR");
18        l0.setFont(f1);
19        l0.setForeground(Color.RED);
20        JPanel fpanel=new JPanel();
21        fpanel.add(l0);
22        fpanel.setBackground(new Color(0,0,64));
23        Font f2=new Font("Times New Roman",Font.BOLD,20);
24        l1=new JLabel("LOGIN");
25        l1.setFont(f1);
26        l1.setForeground(Color.RED);
27        JPanel fxpanel=new JPanel();
28        fxpanel.add(l1);
29        fxpanel.setBackground(new Color(0,0,64));
30        l2=new JLabel("USERNAME");
31        l2.setFont(f2);
32        l2.setForeground(Color.RED);
33        l3=new JLabel("PASSWORD");
34        l3.setFont(f2);
35        l3.setForeground(Color.RED);
36        t1=new JTextField();
37        p1=new JPasswordField();
38        JPanel xpanel=new JPanel();
39        xpanel.add(l2);
40        xpanel.setBackground(new Color(0,0,64));
41        JPanel xppanel=new JPanel();
42        xppanel.add(t1);
43        xppanel.setBackground(new Color(0,0,64));
44        JPanel lpanel=new JPanel();
45        lpanel.add(l3);
46        lpanel.setBackground(new Color(0,0,64));
47        JPanel xlpanel=new JPanel();
48        xlpanel.add(p1);
49        xlpanel.setBackground(new Color(0,0,64));
50        login=new JButton("LOGIN");
51        JPanel gpanel=new JPanel();
52        gpanel.add(login);
53        gpanel.setBackground(new Color(0,0,64));
54        login.addActionListener(this);
55        back=new JButton("BACK");
56        JPanel apanel=new JPanel();
57        apanel.add(back);
58        apanel.setBackground(new Color(0,0,64));
59        back.addActionListener(this);
60        c.add(fpanel);c.add(fxpanel);
61        c.add(xpanel);c.add(t1);
62        c.add(lpanel);c.add(p1);

```

UserMain1.java

```

67      c.add(gpanel);c.add(apanel);
68      setSize(500,425);
69      setLocation(200,200);
70      setResizable(false);
71      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
72      setVisible(true);
73  }
74  public void actionPerformed(ActionEvent e)
75  {
76      if(e.getSource()==login) {
77          String n = t1.getText().trim();
78          String ps = p1.getText().trim();
79          String namepattern = "[A-Za-z]*";
80          Scanner scan = new Scanner( n );
81          String matched = scan.findInLine( namepattern );
82          if ( matched == null ) {
83              JOptionPane.showMessageDialog(this, "INVALID USER NAME\nIt should
contain only alphabet.", "Error", JOptionPane.ERROR_MESSAGE);
84              t1.setText("");
85          }
86          else
87              flagID = true;
88          String passwordpattern = "[0-9]{4}";
89          Scanner scan2 = new Scanner( ps );
90          String matched2 = scan2.findInLine( passwordpattern );
91          if ( matched2 == null ) {
92              JOptionPane.showMessageDialog(this, "INVALID PASSWORD\nIt should
contain only digits with minimum password length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
93              p1.setText("");
94          }
95          else
96              flagPass = true;
97          if(flagID == true) {
98              if(flagPass == true) {
99                  String enteredID = t1.getText().trim();
100                 String enteredPassword = p1.getText().trim();
101                 try {
102                     FileInputStream fin = new FileInputStream("Reg.dat");
103                     ObjectInputStream oin = new ObjectInputStream(fin);
104                     alistTemp = (ArrayList<OperatorRegData>)oin.readObject();
105                 }catch(Exception ex) {
106                     JOptionPane.showMessageDialog(this, "No file found in
database.", "Error", JOptionPane.ERROR_MESSAGE);
107                 }
108                 for(OperatorRegData element : alistTemp) {
109                     if(element.getName().equals(enteredID)) {
110                         if(element.getPassword().equals(enteredPassword))
111                             flag = true;
112                     }
113                 }
114                 if(flag == true) {
115                     JOptionPane.showMessageDialog(this, "Successfully logged
in");
116                     new Userwindow(t1.getText());
117                     setVisible(false);
118                 }
119                 else {
120                     JOptionPane.showMessageDialog(this, "Incorrect UserID or
Password\nTry again.", "Error", JOptionPane.ERROR_MESSAGE);
121                     t1.setText("");
122                     p1.setText("");

```

```
123         }
124     }
125 }
126     flagID = false;
127     flagPass = false;
128 }
129     if(e.getSource()==back) {
130         int con=JOptionPane.showConfirmDialog(this, "Are You Sure to go
back?");
131         if(con==JOptionPane.YES_OPTION) {
132             new FirstWindow("");
133             setVisible(false);
134         }
135     }
136 }
137 }
138 public class UserMain1 {
139     public static void main(String[] args) {}
140 }
```


UserRemove.java

```

1 package test;
2 import java.awt.event.*;
7 public class UserRemove extends JFrame implements ActionListener {
8     private JLabel l1,l2,l3;
9     private JTextField t1;
10    private JButton submit,back;
11    private boolean flagAccNum = false;
12    public UserRemove() {
13        super("Remove an user");
14        Container c=getContentPane();
15        c.setLayout(new GridLayout(3,2));
16        Font f1=new Font("Times New Roman",Font.BOLD,20);
17        l1=new JLabel("USER");
18        l1.setFont(f1);
19        l1.setForeground(Color.RED);
20        JPanel fpanel=new JPanel();
21        fpanel.add(l1);
22        fpanel.setBackground(new Color(0,0,64));
23        l2=new JLabel("Delete");
24        l2.setFont(f1);
25        l2.setForeground(Color.RED);
26        JPanel apanel=new JPanel();
27        apanel.add(l2);
28        apanel.setBackground(new Color(0,0,64));
29        l3=new JLabel("Give USER A/C no");
30        l3.setFont(f1);
31        l3.setForeground(Color.RED);
32        JPanel bpanel=new JPanel();
33        bpanel.add(l3);
34        bpanel.setBackground(new Color(0,0,64));
35        t1=new JTextField();
36        submit=new JButton("Delete");
37        submit.addActionListener(this);
38        JPanel cpanel=new JPanel();
39        cpanel.add(submit);
40        cpanel.setBackground(new Color(0,0,64));
41        back=new JButton("BACK");
42        back.addActionListener(this);
43        JPanel dpanel=new JPanel();
44        dpanel.add(back);
45        dpanel.setBackground(new Color(0,0,64));
46        c.add(fpanel);c.add(apanel);
47        c.add(bpanel);c.add(t1);
48        c.add(cpanel);c.add(dpanel);
49        setSize(550,300);
50        setLocation(200,200);
51        setResizable(false);
52        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
53        setVisible(true);
54    }
55    public void actionPerformed(ActionEvent e) {
56        if(e.getSource() == submit) {
57            String vaccNum = t1.getText();
58            String accNumpattern = "[0-9]{4}" ;
59            Scanner scan = new Scanner(vaccNum) ;
60            String matched = scan.findInLine(accNumpattern) ;
61            if ( matched == null ) {
62                JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
63                t1.setText("");
64            }

```

```
65         else
66             flagAccNum = true;
67         if(flagAccNum == true) {
68             String sname = t1.getText().trim();
69             new UserRemoveDisplay(sname);
70             t1.setText("");
71         }
72         flagAccNum = false;
73     }
74     if(e.getSource() == back) {
75         int con = JOptionPane.showConfirmDialog(this, "Are You Sure to cancel?");
76         if(con == JOptionPane.YES_OPTION) {
77             new Userwindow("");
78             setVisible(false);
79         }
80     }
81 }
82 }
```

UserRemoveDisplay.java

```

1 package test;
2 import java.io.*;
3
4
5 public class UserRemoveDisplay extends JFrame {
6     ArrayList<Actotal> list1;
7     ArrayList<Create> list2;
8     int p;
9     public UserRemoveDisplay(String name) {
10         p=Integer.parseInt(name);
11         try {
12             FileInputStream fin=new FileInputStream("Regis.dat");
13             ObjectInputStream oin=new ObjectInputStream(fin);
14             list2=(ArrayList<Create>)oin.readObject();
15         } catch (Exception e) {
16             list2 = new ArrayList<Create>();
17         }
18         int r=0,c=0;
19         for(int i=0;i<list2.size();i++) {
20             if(list2.get(i).getAc().equals(name)) {
21                 r++;
22                 String rl=String.valueOf(r);
23                 list2.remove(i);
24             }
25         }
26         try {
27             FileOutputStream fout=new FileOutputStream("Regis.dat");
28             ObjectOutputStream oout=new ObjectOutputStream(fout);
29             oout.writeObject(list2);
30         } catch (Exception e) {}
31         try {
32             FileInputStream fin=new FileInputStream("actotal.dat");
33             ObjectInputStream oin=new ObjectInputStream(fin);
34             list1=(ArrayList<Actotal>)oin.readObject();
35         } catch (Exception e) {
36             list1 = new ArrayList<Actotal>();
37         }
38         for(int i=0;i<list1.size();i++) {
39             if(list1.get(i).getAccount()==p) {
40                 c++;
41                 String cl=String.valueOf(c);
42                 list1.remove(i);
43             }
44             else
45                 i++;
46         }
47         try {
48             FileOutputStream fout=new FileOutputStream("actotal.dat");
49             ObjectOutputStream oout=new ObjectOutputStream(fout);
50             oout.writeObject(list1);
51         } catch (Exception e) {}
52         if(r==1)
53             JOptionPane.showMessageDialog(this, "USER RECORD REMOVED");
54         else
55             JOptionPane.showMessageDialog(this, "No Data Found");
56     }
57 }

```

UserSearch.java

```

1 package test;
2 import java.awt.event.*;
6 public class UserSearch extends JFrame implements ActionListener {
7     private JLabel l1,l2,l3;
8     private JTextField t1;
9     private JButton submit,back;
10    private boolean flag = false;
11    public UserSearch() {
12        super("User Search");
13        Container c=getContentPane();
14        c.setLayout(new GridLayout(3,2));
15        Font f1=new Font("Times New Roman",Font.BOLD,20);
16        l1=new JLabel("CUSTOMER");
17        l1.setFont(f1);
18        l1.setForeground(Color.RED);
19        JPanel fpanel=new JPanel();
20        fpanel.add(l1);
21        fpanel.setBackground(new Color(0,0,64));
22        l2=new JLabel("SEARCH");
23        l2.setFont(f1);
24        l2.setForeground(Color.RED);
25        JPanel apanel=new JPanel();
26        apanel.add(l2);
27        apanel.setBackground(new Color(0,0,64));
28        l3=new JLabel("Give User A/C no");
29        l3.setFont(f1);
30        l3.setForeground(Color.RED);
31        JPanel bpanel=new JPanel();
32        bpanel.add(l3);
33        bpanel.setBackground(new Color(0,0,64));
34        t1=new JTextField();
35        submit=new JButton("SUBMIT");
36        submit.addActionListener(this);
37        JPanel cpanel=new JPanel();
38        cpanel.add(submit);
39        cpanel.setBackground(new Color(0,0,64));
40        back=new JButton("BACK");
41        back.addActionListener(this);
42        JPanel dpanel=new JPanel();
43        dpanel.add(back);
44        dpanel.setBackground(new Color(0,0,64));
45        c.add(fpanel);c.add(apanel);
46        c.add(bpanel);c.add(t1);
47        c.add(cpanel);c.add(dpanel);
48        setSize(550,300);
49        setLocation(200,200);
50        setResizable(false);
51        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
52        setVisible(true);
53    }
54    public void actionPerformed(ActionEvent e) {
55        if(e.getSource() == submit) {
56            String n = t1.getText().trim();
57            String namepattern = "[0-9]{4}";
58            Scanner scan = new Scanner(n);
59            String matched = scan.findInLine(namepattern);
60            if (matched == null) {
61                JOptionPane.showMessageDialog(this, "INVALID USER ACCOUNT NUMBER\nIt
should contain only digit with minimum of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
62                t1.setText("");
63            }

```

```
64         else
65             flag = true;
66         if(flag == true) {
67             String sname = t1.getText().trim();
68             new UserSearchDisplay(sname);
69             t1.setText("");
70         }
71         flag = false;
72     }
73     if(e.getSource() == back) {
74         int con = JOptionPane.showConfirmDialog(this, "Are you sure to cancel?");
75         if(con == JOptionPane.YES_OPTION) {
76             new Admin();
77             setVisible(false);
78         }
79     }
80 }
81 }
```

UserSearchDisplay.java

```

1 package test;
2 import java.awt.*;
6 public class UserSearchDisplay extends JFrame {
7     String data[][];
8     String heading[]={ "First Name", "Last name", "Address", "Nationality", "Account
    Type", "Date Of Birth", "Identity", "Profession", "Reg Date", "Reg Time", "A/c
    no", "Balance"};
9     ArrayList<Create> list;
10    public UserSearchDisplay(String name) {
11        super("Search for user account number " + name);
12        try {
13            FileInputStream fin=new FileInputStream("Regis.dat");
14            ObjectInputStream oin=new ObjectInputStream(fin);
15            list=(ArrayList<Create>)oin.readObject();
16            int row=0;
17            for(Create re : list) {
18                if(re.getAc().equals(name))
19                    row++;
20            }
21            data = new String[row][heading.length+1];
22            int r=0;
23            for(Create re : list) {
24                if(re.getAc().equals(name)) {
25                    data[r][0]=re.getName();
26                    data[r][1]=re.getName1();
27                    data[r][2]=re.getAddress();
28                    data[r][3]=re.getNationality();
29                    data[r][4]=re.getAcctype();
30                    data[r][5]=re.getDob();
31                    data[r][6]=re.getIdentity();
32                    data[r][7]=re.getProfession();
33                    data[r][8]=re.getDate();
34                    data[r][9]=re.getTime();
35                    data[r][10]=re.getAc();
36                    data[r][11]=re.getInitialamnt();
37                    r++;
38                }
39            }
40            if (r>0) {
41                Container con=getContentPane();
42                con.setLayout(new BorderLayout());
43                JTable datatable=new JTable(data, heading);
44                JScrollPane jsp=new JScrollPane(datatable);
45                con.add(new JLabel("Registration Details"),BorderLayout.NORTH);
46                con.add(jsp,BorderLayout.CENTER);
47                setSize(650, 300);
48                setLocation(200, 200);
49                setVisible(true);
50            }
51            else
52                JOptionPane.showMessageDialog(this, "Match not Found");
53        } catch (Exception e) {
54            JOptionPane.showMessageDialog(this, "No file found in data base",
    "Error", JOptionPane.ERROR_MESSAGE);
55        }
56    }
57 }

```

UserWindow.java

```

1 package test;
2 import java.awt.*;
3 class Userwindow extends JFrame implements ActionListener {
4     private JLabel l1,l2;
5     private JButton
        bcreate,bremove,bdetails,bwithdraw,btransfer,bpass,bexit,bChangePassword;
6     Userwindow(String currentOperator) {
7         super("Welcome " + currentOperator);
8         Container c=getContentPane();
9         c.setLayout(new GridLayout(5,2));
10        Font f1=new Font("Times New Roman",Font.BOLD,25);
11        l1=new JLabel("OPERATOR");
12        l1.setFont(f1);
13        l1.setForeground(Color.WHITE);
14        JPanel fpanel=new JPanel();
15        fpanel.add(l1);
16        fpanel.setBackground(new Color(0,0,64));
17        l2=new JLabel("FUNCTIONS");
18        l2.setFont(f1);
19        l2.setForeground(Color.WHITE);
20        JPanel ppanel=new JPanel();
21        ppanel.add(l2);
22        ppanel.setBackground(new Color(0,0,64));
23        bcreate=new JButton("CREATE NEW ACCOUNT");
24        bcreate.addActionListener(this);
25        JPanel apanel=new JPanel();
26        apanel.add(bcreate);
27        apanel.setBackground(new Color(0,0,64));
28        bremove=new JButton("REMOVE ACCOUNT");
29        bremove.addActionListener(this);
30        JPanel bpanel=new JPanel();
31        bpanel.add(bremove);
32        bpanel.setBackground(new Color(0,0,64));
33        bdetails=new JButton("VIEW DETAILS OF AN ACCOUNT");
34        bdetails.addActionListener(this);
35        JPanel cpanel=new JPanel();
36        cpanel.add(bdetails);
37        cpanel.setBackground(new Color(0,0,64));
38        bwithdraw=new JButton("WITHDRAWAL/DEPOSIT");
39        bwithdraw.addActionListener(this);
40        JPanel dpanel=new JPanel();
41        dpanel.add(bwithdraw);
42        dpanel.setBackground(new Color(0,0,64));
43        btransfer=new JButton("TRANSFER MONEY");
44        btransfer.addActionListener(this);
45        JPanel epanel=new JPanel();
46        epanel.add(btransfer);
47        epanel.setBackground(new Color(0,0,64));
48        bpass=new JButton("VIEW ACCOUNT PASS BOOK");
49        bpass.addActionListener(this);
50        JPanel gpanel=new JPanel();
51        gpanel.add(bpass);
52        gpanel.setBackground(new Color(0,0,64));
53        bexit=new JButton("LOG OUT");
54        bexit.addActionListener(this);
55        JPanel hpanel=new JPanel();
56        hpanel.add(bexit);
57        hpanel.setBackground(new Color(0,0,64));
58        bChangePassword=new JButton("CHANGE PASSWORD");
59        bChangePassword.addActionListener(this);
60        JPanel iapanel=new JPanel();
61        iapanel.add(bChangePassword);

```

```

64     aapanel.setBackground(new Color(0,0,64));
65     c.add(fpanel);c.add(ppanel);
66     c.add(apanel);c.add(dpanel);
67     c.add(bpanel);c.add(epanel);
68     c.add(qpanel);c.add(cpanel);
69     c.add(aapanel);c.add(gpanel);
70     setSize(500,400);
71     setLocation(200,200);
72     setResizable(false);
73     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
74     setVisible(true);
75 }
76 public void actionPerformed(ActionEvent e) {
77     if(e.getSource()==bcreate) {
78         new CreateFrame();
79         setVisible(false);
80     }
81     if(e.getSource()==bwithdraw) {
82         new Withdraw();
83         setVisible(false);
84     }
85     if(e.getSource()==bremove) {
86         new UserRemove();
87         setVisible(false);
88     }
89     if(e.getSource()==btransfer) {
90         new Transfer();
91         setVisible(false);
92     }
93     if(e.getSource()==bpass) {
94         new PassBook();
95         setVisible(false);
96     }
97     if(e.getSource()==bdetails) {
98         new OperatorUserSearch();
99         setVisible(false);
100    }
101    if(e.getSource()==bexit) {
102        int con=JOptionPane.showConfirmDialog(this, "Are you sure to
cancel?");
103        if(con==JOptionPane.YES_OPTION) {
104            new ThirdWindow();
105            setVisible(false);
106        }
107    }
108    if(e.getSource()==bChangePassword) {
109        new ChangePassword();
110        setVisible(false);
111    }
112 }
113 }

```


Withdraw.java

```

1 package test;
2 import java.awt.event.*;
3
4 public class Withdraw extends JFrame implements ActionListener {
5     private JLabel l1,l2,l3,l4;
6     private JTextField t1,t2;
7     private JButton withdraw, cancel, deposit;
8     private int x, amnt, t, w, account, d, r=0;
9     ArrayList<Create> list1;
10    private boolean flagWithdraw = false;
11    private boolean flagAccNum = false;
12    private boolean flagYes = false;
13    private boolean flagAccMatch = false;
14    private boolean flagNo = false;
15    public Withdraw() {
16        super("DEPOSIT/WITHDRAWAL SLIP");
17        Container c=getContentPane();
18        c.setLayout(new GridLayout(4,2));
19        Font f1=new Font("Times New Roman",Font.BOLD,20);
20        l1=new JLabel("Enter ACCOUNT NO:");
21        l1.setFont(f1);
22        l1.setForeground(Color.GRAY);
23        JPanel bpanel=new JPanel();
24        bpanel.add(l1);
25        bpanel.setBackground(new Color(0,0,64));
26        l2=new JLabel("ENTER AMOUNT:");
27        l2.setFont(f1);
28        l2.setForeground(Color.GRAY);
29        JPanel blpanel=new JPanel();
30        blpanel.add(l2);
31        blpanel.setBackground(new Color(0,0,64));
32        l3=new JLabel("DEPOSIT:");
33        l3.setFont(f1);
34        l3.setForeground(Color.GRAY);
35        JPanel b2panel=new JPanel();
36        b2panel.add(l3);
37        b2panel.setBackground(new Color(0,0,64));
38        t1=new JTextField();
39        t2=new JTextField();
40        withdraw=new JButton("WITHDRAW");
41        withdraw.addActionListener(this);
42        JPanel dpanel=new JPanel();
43        dpanel.add(withdraw);
44        dpanel.setBackground(new Color(0,0,64));
45        deposit=new JButton("DEPOSIT");
46        deposit.addActionListener(this);
47        JPanel dlpanel=new JPanel();
48        dlpanel.add(deposit);
49        dlpanel.setBackground(new Color(0,0,64));
50        cancel=new JButton("CANCEL");
51        cancel.addActionListener(this);
52        JPanel epanel=new JPanel();
53        epanel.add(cancel);
54        epanel.setBackground(new Color(0,0,64));
55        l4=new JLabel("");
56        l4.setFont(f1);
57        l4.setForeground(Color.GRAY);
58        JPanel b3panel=new JPanel();
59        b3panel.add(l4);
60        b3panel.setBackground(new Color(0,0,64));
61        c.add(bpanel);c.add(t1);
62        c.add(blpanel);c.add(t2);
63        c.add(dpanel);c.add(dlpanel);

```

Withdraw.java

```

68      c.add(epanel);c.add(b3panel);
69      setSize(500,500);
70      setLocation(100,100);
71      setResizable(false);
72      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
73      setVisible(true);
74  }
75  public void actionPerformed(ActionEvent e) {
76      String search=t1.getText();//search ----->account number
77      if(e.getSource()==withdraw) {
78          String vaccNum = t1.getText();
79          String vwithdrw = t2.getText();
80          String accNumpattern = "^[0-9]{4}" ;
81          Scanner scanl = new Scanner( vaccNum ) ;
82          String matchedl = scanl.findInLine( accNumpattern ) ;
83          if ( matchedl == null ) {
84              JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
85              t1.setText("");
86          }
87          else
88              flagAccNum = true;
89          String withdrawpattern = "^[0-9]{3}" ;
90          Scanner scan = new Scanner( vwithdrw ) ;
91          String matched = scan.findInLine( withdrawpattern ) ;
92          if ( matched == null ) {
93              JOptionPane.showMessageDialog(this, "INVALID WITHDRAW AMOUNT\nIt
should contain only digits with minimum withdraw amount of INR 100.", "Error",
JOptionPane.ERROR_MESSAGE);
94              t2.setText("");
95          }
96          else
97              flagWithdraw = true;
98          if(flagAccNum == true) {
99              if(flagWithdraw == true) {
100                  try {
101                      FileInputStream fin=new FileInputStream("Regis.dat");
102                      ObjectInputStream oin=new ObjectInputStream(fin);
103                      list1=(ArrayList<Create>)oin.readObject();
104                      for(Create re : list1) {
105                          if(re.getAc().equals(search)) {
106                              flagAccMatch = true;
107                              w=Integer.parseInt(t2.getText());
108                              amnt=Integer.parseInt(re.getInitialamnt());
109                              if(amnt>=500) {
110                                  flagNo = true;
111                                  t = amnt - w;
112                                  if(t >= 500) {
113                                      account=Integer.parseInt(t1.getText());
114                                      d=0;
115                                      flagYes = true;
116                                  }
117                              }
118                          }
119                      }
120                      if((flagAccMatch == true) && (flagNo == true) && (flagYes
== true)) {
121                          new Acttotalcreate(account,w,d,t);
122                          new TotalUpdate(account,t);
123                          JOptionPane.showMessageDialog(this, "Withdraw done
successfully");

```

Withdraw.java

```

124         new Passdisp(t1.getText());
125         t1.setText("");
126         t2.setText("");
127     }
128     if((flagAccMatch == true) && (flagNo == true) && (flagYes
== false)) {
129         JOptionPane.showMessageDialog(this, "Insufficient
balance.\nAccount balance must be 500 after withdraw.", "Error",
JOptionPane.ERROR_MESSAGE);
130         t1.setText("");
131         t2.setText("");
132     }
133     if((flagAccMatch == true) && (flagNo == false) && (flagYes
== false)) {
134         JOptionPane.showMessageDialog(this, "Account number has
not the minimum balance of INR 500.", "Error", JOptionPane.ERROR_MESSAGE);
135         t1.setText("");
136         t2.setText("");
137     }
138     if((flagAccMatch == false) && (flagNo == false) && (flagYes
== false)) {
139         JOptionPane.showMessageDialog(this, "Account number
does not exists", "Error", JOptionPane.ERROR_MESSAGE);
140         t1.setText("");
141         t2.setText("");
142     }
143     flagYes = false;
144     flagAccMatch = false;
145     flagNo = false;
146 }
147 catch(Exception e1) {
148     JOptionPane.showMessageDialog(this, "No file found in data
base", "Error", JOptionPane.ERROR_MESSAGE);
149     t1.setText("");
150     t2.setText("");
151 }
152 }
153 }
154 flagAccNum =false;
155 flagWithdraw = false;
156 }
157 if(e.getSource()==cancel) {
158     int rply = JOptionPane.showConfirmDialog(this, "Are you sure to
quit?");
159     if(rply == JOptionPane.YES_OPTION) {
160         new Userwindow("");
161         setVisible(false);
162     }
163 }
164 if(e.getSource()==deposit) {
165     String vaccNum = t1.getText();
166     String vwithdrw = t2.getText();
167     String accNumpattern = "[0-9]{4}" ;
168     Scanner scanl = new Scanner( vaccNum ) ;
169     String matchedl = scanl.findInLine( accNumpattern ) ;
170     if ( matchedl == null ) {
171         JOptionPane.showMessageDialog(this, "INVALID ACCOUNT NUMBER\nIt
should contain only digits with minimum length of 4 digit.", "Error",
JOptionPane.ERROR_MESSAGE);
172         t1.setText("");
173     }
174     else

```

Withdraw.java

```

175         flagAccNum    = true;
176         String withdrawpattern = "^[0-9]{3}" ;
177         Scanner  scan = new Scanner( vwithdrw ) ;
178         String  matched = scan.findInLine( withdrawpattern ) ;
179         if ( matched == null ) {
180             JOptionPane.showMessageDialog(this, "INVALID WITHDRAW AMOUNT\nIt
should contain only digits with minimum withdraw amount of INR 100.", "Error",
JOptionPane.ERROR_MESSAGE);
181             t2.setText("");
182         }
183         else
184             flagWithdraw    = true;
185         if(flagAccNum == true) {
186             if(flagWithdraw == true) {
187                 try {
188                     FileInputStream fin=new FileInputStream("Regis.dat");
189                     ObjectInputStream oin=new ObjectInputStream(fin);
190                     list1=(ArrayList<Create>)oin.readObject();
191                     for(Create re : list1) {
192                         if(re.getAc().equals(search)) {
193                             flagAccMatch = true;
194                             amnt=Integer.parseInt(re.getInitialamnt());
195                             w=Integer.parseInt(t2.getText());
196                             t=(amnt+w);
197                             account=Integer.parseInt(t1.getText());
198                             d=0;
199                             r++;
200                         }
201                     }
202                     if((flagAccNum == true) && (flagWithdraw== true) &&
(flagAccMatch == true)) {
203                         new Actotalcreate(account,d,w,t);
204                         new TotalUpdate(account,t);
205                         JOptionPane.showMessageDialog(this, "Deposit done
successfully");
206                         new Passdisp(t1.getText());
207                         t1.setText("");
208                         t2.setText("");
209                     }
210                     if(flagAccMatch == false) {
211                         JOptionPane.showMessageDialog(this, "Account number
does not exists");
212                         t1.setText("");
213                         t2.setText("");
214                     }
215                     flagAccMatch = false;
216                 }catch(Exception e1) {
217                     JOptionPane.showMessageDialog(this, "No file found in data
base", "Error", JOptionPane.ERROR_MESSAGE);
218                     t1.setText("");
219                     t2.setText("");
220                 }
221             }
222         }
223         flagAccNum =false;
224         flagWithdraw = false;
225     }
226 }
227 }

```