Appendix A

```
import javax.swing.JOptionPane; // import class JOptionPane
import java.awt.Container;
import javax.swing.*;
import java.io.*;
public class Cell
  {
        protected static int sim_time = 3600,Gen_seq = 6,arrival_seq =
3,moving_act,duration_act,instant_time,gen,i,cel,id,rit,j;
        //protected static int
sim_time=0,Gen_seq=0,arrival_seq=0,moving_act,duration_act,instant_time,gen,i,cel,id,
rit;
        protected static int users = ( arrival_seq * sim_time ) / Gen_seq,request,er,go;
        protected static int cellNo,mobiNo,cluster_size = 7,users_count;
        protected static double system_utilization,T_holding_time,arrival,sequence;
                                //Channel consumption ratio (for the system)
        protected static double
GOS[]={0,0,0,0,0,0,0},powintensity[]={0,0,0,0,0,0,0},c_intensity[]={0,0,0,0,0,0,0},sum
mation[]={0,0,0,0,0,0,0};                //Erlang-B parameters
        protected static double sumpowintensity,kfactorial,cfactorial,c,fc,k,fk,s;
        //Erlang-B parameters
        protected static double tchannels=84;
                        //Number of Truncked Channels
        protected static double c_holding_time[]=new double[7];
                                //Traffic time on each cell
        protected static double ch[]={84,84,84,84,84,84,84};
                                                                        //channel(s)
limit in each cell
        protected static double Tu_in_c[]={0,0,0,0,0,0,0};
                                                        //Total Number of users Enter each
cell
        protected static double         cell_utilization[] = new double[7];
                                                                //Channels utilization
in each cell
        protected static double accepted_u_in_c[]={0,0,0,0,0,0,0};
                                                                        //accepted
Users
        protected static double block[]={0,0,0,0,0,0,0};
                                //Number of Blocking in each cell
        protected static double u_in_c[]={0,0,0,0,0,0,0};
                //user(s) in each cell
        protected static double drop[]={0,0,0,0,0,0,0};
                 //Number of Dropping in each cell
```

```
        protected static double block_probability[]={0,0,0,0,0,0,0};
                //Blocking Probability of each cell
        protected static double drop_probability[]={0,0,0,0,0,0,0};
        //Dropping Probability of each cell
        protected static double Expo_rand,U,mu=-0.05,mn=-1;
                //Exponential Parameters
        protected static double
Rand,Positive_Rand,Negative_Rand,Pois_rand,ExPois_rand,P;
        //Poisson Parameters
        protected static double x1,x2,Gaus_rand1,Gaus_rand2,w,z;
//Gaussian Parameters
        protected double ax[]=new double[7];
//Mobile X axis measures
        protected double ay[]=new double[7];
//Mobile Y axis measures
        protected double dest[]=new double[7];
//Destination from BTS
      //protected double power_u_in_c[]=new double[7];   // new
      //protected static int  power=0;         //new
        protected double x[]={125,200,200,125,50,50,125};    //BTS (cell center) X
measures


        protected double y[]={129.9,86.6,173.2,216.5,173.2,86.6,43.3};
            //BTS (cell center) Y measures

public void Exponential()
{
                U = Math.random();
                Expo_rand = mu*Math.log(U/20);
}
public void Poisson()
{
        Rand = Math.random()+0.000001;
        Positive_Rand = Math.log(5-Rand) - 1;
        Negative_Rand = mu*Math.log(Positive_Rand);
        Pois_rand = Positive_Rand + Negative_Rand;
        P = Math.random()+0.000001;
        ExPois_rand = mn*Math.log(P);

}
public void Gaussian()
{
        do{
                        do{
                        x1 = 2.0 * ((double) (Math.random())) - 1.0;
```

```
                        x2 = 2.0 * ((double) (Math.random())) - 1.0;
                        z = (x1 * x1) + (x2 * x2);
                        }  while ( z >= 1 );

                w = Math.sqrt( (-2.0 * Math.log( z ) ) / z );

        } while ( ( Math.log( z ) / z ) <=  ( -0.5 ) );

        Gaus_rand1 = x1 * w;
        Gaus_rand2 = x2 * w;
}
public void ErlangB()
{
        for(cellNo=0;cellNo<=6;cellNo++)
                {
                        c_intensity[cellNo] = ( c_holding_time[cellNo] / sim_time );
        //Total Traffic Intensity in each cell
                        System.out.println("Traffic: "+c_holding_time[cellNo]);
                        System.out.println("\nIntensity = "+c_intensity[cellNo]);
                        powintensity[cellNo] = Math.pow(c_intensity[cellNo],tchannels);

                        for(k=0;k<=tchannels;k++)
                                {
                                        sumpowintensity =
Math.pow(c_intensity[cellNo],k);

                                        kfactorial = 1;
                                        for(fk=k;fk>=1;fk--)
                                                {
                                                        kfactorial = kfactorial * fk;
                                                }
                                        summation[cellNo] = summation[cellNo] +
( sumpowintensity / kfactorial );
                                }
                }
        cfactorial = 1;
        for(fc=tchannels;fc>=1;fc--)
                {
                        cfactorial = cfactorial * fc;
                }
        for(cellNo=0;cellNo<=6;cellNo++)
                {
                        GOS[cellNo] =( ( ( powintensity[cellNo] / cfactorial ) /
( summation[cellNo] ) ) *100 );                                         //GOS (Grade
of service)
```

```
                        System.out.println("\nCell("+cellNo+") GOS (%):
"+GOS[cellNo]);
                }
}
public void Result()
{
        System.out.println("\tSimulated Model - Mobile Cluster, N = "+cluster_size+"
Cells");
        System.out.println("\nTotal # of users Entered the system: "+users_count);

        for(int i=0;i<=6;i++)
        {
                System.out.println("\n\tCell ID ("+i+")");
                System.out.println("# of User(s) Entered The Cell: "+Tu_in_c[i]);

                accepted_u_in_c[i] = ( Tu_in_c[i] - ( block[i] + drop[i] ) );
                System.out.println("# of Accepted User(s) in The Cell:
"+accepted_u_in_c[i]);                  //Accepted Users with no Drop
                System.out.println("# of block(s) in The Cell: "+block[i]);
                System.out.println("# of drop(s) in The Cell: "+drop[i]);

                block_probability[i] = (  block[i]  / Tu_in_c[i] ) * 100;
                                                //Blocking Probability Formula "Blocked
Calls / Attempted Calls"
                drop_probability[i] = ( drop[i] / ( Tu_in_c[i] - block[i] ) ) * 100;
                        //Dropping Probability Formula "Dropped Calls / Accepted Calls"
                System.out.println("\nBlocking Probability (%) : "+block_probability[i]);
                System.out.println("\nDropping Probability (%) : "+drop_probability[i]);
                cell_utilization[i] = ( ( c_holding_time[i] / ( sim_time * tchannels ) ) *
100 );
               System.out.println("\nCell Channels Utilization (%) : "+cell_utilization[i]);
        }
                system_utilization = ( ( T_holding_time / ( sim_time * tchannels *
cluster_size ) ) * 100);
                System.out.println("\nSystem Channels Utilization (%) :
"+system_utilization);
}
public void filecell()
{
        try {
                        BufferedWriter erlang = new BufferedWriter(new
FileWriter("ErlangB.xls"));
                        erlang.write("Traffic Intensity");
                        for(er=0;er<=6;er++)
                                erlang.write("\t"+c_intensity[er] );
                        erlang.write("\nGrade Of Service");
```

```
                        for(go=0;go<=6;go++)
                                erlang.write("\t"+GOS[go] );
                        erlang.close();

                        BufferedWriter cell = new BufferedWriter(new
FileWriter("cells.xls"));
                        cell.write("\t\t\tTable of the System Cells\n");
                        cell.write("\tSimulated Model - Mobile System Cluster, N =
"+cluster_size+" Cells\n");
                        cell.write("\tTotal MS Entered System\t"+users_count+"\n");
                        cell.write("\tSystem Channels
Utilization\t"+system_utilization+"(%)\n");
                        cell.write("Cell ID\tTotal MS Entered the Cell\tAccepted
MS\tBlocked MS\tDropped MS\tTraffic Intensity (Erlang)\tBlocking Probability
(%)\tDropping Probability(%)\tCell Channels Utilization (%)\tGrade of Service (%)\n");
                        for(cel=0;cel<=6;cel++)

        cell.write(cel+"\t"+Tu_in_c[cel]+"\t"+accepted_u_in_c[cel]+"\t"+block[cel]+"\t"
+drop[cel]+"\t"+c_intensity[cel]+"Erlang\t"+block_probability[cel]+"(%)\t"+drop_proba
bility[cel]+"(%)\t"+cell_utilization[cel]+"\t"+GOS[cel]+"(%)\n");
                        cell.close();
                    }
        catch (IOException e)
   {
System.err.println ("Error writing to file");
}
}

}
```

Appendix B

```
import javax.swing.JOptionPane; // import class JOptionPane
 import javax.swing.*;
 import java.io.*;

public class Mobile extends Cell
{
        protected double
ym,angle,v,xm,ranv,rano,dur,holding_time,handover_count,act_time,depart_time,talking
_time,handover_time,cell_time;
        protected int cell_id,old_cell_id,act,Mob_id,power;
        protected String handover = "No",status;
        protected double m_holding_time[] = new double[7];

public void Gen_Mob()

{

                Exponential();
                Gaussian();
                dur = 60 + ( (double) ( 120 * Expo_rand ) );
                xm = 125 + ( (double) ( Gaus_rand1 * 125 ) );
                ym = 129.9 + ( (double) ( Gaus_rand2 * 74.8 * Math.sqrt( 3 ) ) );
                ranv = ( (double) (Math.random() ) );
                rano = ( (double) (Math.random() ) );

                if(ranv<=0.2)
                        v=0;
                if((ranv>0.2)&&(ranv<=0.35))
                        v=0.8333;
        if(ranv>0.5)
                v=13.8889;

                if(rano<=0.1)
                        angle=0;
                if((rano>0.1)&&(rano<=0.6))
                        angle=2.356;
                if(rano>0.6);
                        angle=5.495;

        //1st location
                for (i=0;i<=6;i++)
```

```
                {
                        ax[i] =  xm-x[i] ;
                        ax[i] = ( (int) (Math.abs(ax[i]) ) );
                        ay[i] = ym-y[i];
                        ay[i] = ( (int) (Math.abs(ay[i]) ) );;
                        dest[i] = ( (int) (Math.sqrt( ( ax[i]*ax[i] ) + ( ay[i]*ay[i] ) ) ) );
                        if (dest[i] <= dest[0])
                                {
                                        dest[0] = dest[i];
                                        cell_id = i;
                                }
                }

        Tu_in_c[cell_id]++;

                //Admission Control
                if(u_in_c[cell_id]<ch[cell_id])
                        {
                                act=1;
        //Activate new user
                                u_in_c[cell_id]++;                              //Add new
user
                                ch[cell_id]--;
        //occupy channel
                                holding_time = dur;
                        }
                else
                  {
                                act = 0;
                                block[cell_id]++;
                                status = "Admission Control (No available Channels)";
                                depart_time = instant_time;
                        }


}
public void find_loca()
{
        //movement
        if(xm > 250 || xm < 0 || ym > 216.5 || ym < 0)
                OutofRange();
        else
                for (i=0;i<=6;i++)
                {

                        ax[i]= xm-x[i];
```

```
                ax[i]=(int)(Math.abs(ax[i]));
                ay[i]= ym-y[i];
                ay[i]=(int)(Math.abs(ay[i]));
                dest[i]=(int) (Math.sqrt((ax[i]*ax[i])+(ay[i]*ay[i])));
                if (dest[i]<=dest[0])
                        {
                                dest[0]=dest[i];
                                cell_id=i;

                                }

          //* new    u_in_c[i]
         // for (j=0;j<=u_in_c[i];j++)
         // {
           //  xm=ax[i]; //  ma 3rfta laha
            // ym=ay[i];

  // {  //*new
  if((dest[i]<= 10)&&(dest[i]>0))
     power=0;
  else if ((dest[i]<= 20)&&(dest[i]>10))
     power=1;
  else if ((dest[i]<= 30)&&(dest[i]>20))
     power=2;
  else if ((dest[i]<= 40)&&(dest[i]>30))
     power=3;
  else if ((dest[i]<= 50)&&(dest[i]>40))
     power=4;
}
       /*
  if(xm >112.50  || xm < 75 || ym >138.56  || ym <86.6 )
        power=110; //.Mob_id[i]=0;
   else
    if(xm >150 || xm < 112.5 || ym >147.22  || ym <138.56 )
        power=1; //.Mob_id[i]=1;

   else
    if(xm >175  || xm < 150 || ym >155.88  || ym <147.22 )
        power=2;
   else
    if(xm > 187.5 || xm < 175 || ym >164.54  || ym <155.88 )
        power=3;
  else
    if(xm >200  || xm < 187.5  || ym >173.2  || ym <164.54 )
        power=4;
```

```
            }

            // if(xm > 200 || xm < 75 || ym > 173.2 || ym < 86.6)

             }

                  //*new
        */

}
public void Mov_Mob()
{
        if(act==1)
        {
                        xm = xm + ( v * (Math.cos(angle)));
                        ym = ym + ( v * (Math.sin(angle)));

        find_loca();
        if(old_cell_id!=cell_id)
                {
                        Tu_in_c[cell_id]++;
                        hand_over();
                }
        }
}
public void hand_over()
{
        if(u_in_c[cell_id]<ch[cell_id])
                {
                        u_in_c[old_cell_id]--;
                        ch[old_cell_id]++;
                        handover_count++;
                        ch[cell_id]--;
                        u_in_c[cell_id]++;
                        cell_time = ( instant_time - handover_time );
                        m_holding_time[old_cell_id] = ( m_holding_time[old_cell_id] +
cell_time );
                        old_cell_id = cell_id;
                        handover_time = instant_time;
                        handover = "YES";
                }
        else
                {
                        u_in_c[old_cell_id]--;
                        ch[old_cell_id]++;
                        act=0;
```

```
                        depart_time=instant_time;
                        talking_time = depart_time - act_time;
                        cell_time = ( depart_time - handover_time );
                        m_holding_time[old_cell_id] = ( m_holding_time[old_cell_id] +
cell_time );
                        status = "Dropped (No available Channels)";
                        drop[old_cell_id]++;
                }
}
public void OutofRange()
{
        u_in_c[cell_id]--;
        ch[cell_id]++;
        act = 0;
        depart_time = instant_time;
        talking_time = ( depart_time - act_time );
        cell_time = ( depart_time - handover_time );
        m_holding_time[cell_id] = ( m_holding_time[cell_id] + cell_time );
        drop[cell_id]++;
        status = "Handover to other RNC ";

    //status = "Dropped (Out of Reach)";
}
public static void main(String args[])
{
                sim_time=Integer.parseInt(JOptionPane.showInputDialog(null,"Enter the
simulation time","sim_time",JOptionPane.QUESTION_MESSAGE));
                Gen_seq=Integer.parseInt(JOptionPane.showInputDialog(null,"Enter The
Generation Sequence","Gen_seq",JOptionPane.QUESTION_MESSAGE));
                arrival_seq=Integer.parseInt(JOptionPane.showInputDialog(null,"Enter
The Arrival Sequence","arrival_seq",JOptionPane.QUESTION_MESSAGE));
                Mobile M[]=new Mobile[users];

                Cell c=new Cell();

                for(gen=0;gen<users;gen++)
                M[gen]=new Mobile();

                while(instant_time<=sim_time)
                                        //Simulation Time
                {
                        c.Poisson();
                        arrival = 1 + ( (int) ( 2 * Pois_rand ) );
                        sequence = 6 + ( (int) ( 4 * ExPois_rand ) );
                        if((instant_time%sequence)==0)
                                for(request=0;request<=arrival;request++)
```

```
                                        {
                                                M[id].Gen_Mob();
                                                M[id].Mob_id = id;
                                                M[id].old_cell_id = M[id].cell_id;
                                                M[id].act_time = instant_time;
                                                users_count++;
                                                id++;
                                        }

                        duration_act=0;
                                        while(duration_act<id)
                                                        //Mobile Channel Occupation Time
                                {
                                        if(M[duration_act].act==1)
                                        {
                                                if(M[duration_act].dur<=0)
                                                {

                                                        M[duration_act].act = 0;
                                                        M[duration_act].depart_time =
instant_time;
                                                        M[duration_act].talking_time =
M[duration_act].holding_time;
                                                        M[duration_act].cell_time =
( M[duration_act].depart_time - M[duration_act].handover_time );

        M[duration_act].ch[M[duration_act].old_cell_id]++;

        M[duration_act].m_holding_time[M[duration_act].cell_id] =
( M[duration_act].m_holding_time[M[duration_act].cell_id] +
M[duration_act].cell_time );
                                                        M[duration_act].status = "Call
Ended";
                                                }
                                                        M[duration_act].dur--;
                                        }
                                                        duration_act++;
                                }

                        moving_act=0;
                                while(moving_act<id)
                                                        //MS Movement
                                        {
                                                M[moving_act].Mov_Mob();
                                                moving_act++;
                                        }
```

```
                        instant_time++;
                }

for(cellNo=0;cellNo<=6;cellNo++)
        {
                for(mobiNo=0;mobiNo<=users_count;mobiNo++)
                        {
                                c_holding_time[cellNo] = c_holding_time[cellNo] +
M[mobiNo].m_holding_time[cellNo];
                                //System.out.println("\nHolding Time =
"+c_holding_time[cellNo]);
                                //System.out.println("\nMobile Holding Time =
"+M[mobiNo].m_holding_time[cellNo]);
                        }
                T_holding_time = T_holding_time + c_holding_time[cellNo];
        }

        try{
                        BufferedWriter mobile = new BufferedWriter(new
FileWriter("mobiles.xls"));
                        mobile.write("\t\t\tTable of Mobiles in the System\n");
                        mobile.write("Mobile ID\tVelocity(M per Sec)\tAngle
(Radian)\tAccess Time (Second)\tHolding Time (Seconds)\tHandover\tHandover
Count\tDepart Time (Second)\tDepart Status\tpower (dbm)\n");
                        for(rit=0;rit<=users_count;rit++)

        mobile.write(M[rit].Mob_id+"\t"+M[rit].v+"\t"+M[rit].angle+"\t"+M[rit].act_time
+"\t"+M[rit].talking_time+"\t"+M[rit].handover+"\t"+M[rit].handover_count+"\t"+M[rit].
depart_time+"\t"+M[rit].status+"\t"+M[rit].power+"\n");
                        mobile.close();
                }
        catch (IOException e)
   {
        System.err.println ("Error writing to file");
        }

c.Result();
c.ErlangB();
c.filecell();
System.exit( 0 ); // terminate application
}
}
```