Uzupełnij program symulujący model Isinga za pomocą algorytmu Metropolisa. Poniższa wersja odpowiada modelowi w nieskończonej temperaturze, gdzie każdy wybrany spin zmienia znak (bez względu na zmianę energii z tym się wiążącą):

```
//Simulates Ising model with Metropolis Algorithm
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Ising1 extends JApplet implements ActionListener{
  plotno1 mplotno = new plotno1();
       JButton but= new JButton("Start");
       public void init(){
   setSize(600,600);
   setLayout(new FlowLayout());
   add(mplotno);
   add(but);
   but.addActionListener(this);
}
public void actionPerformed(ActionEvent e){
                if(mplotno.flaga == 0){
                     mplotno.flaga=1;
                but.setText("Stop");
                mplotno.repaint();
         }
         else{
                mplotno.flaga=0;
           but.setText("Start");
         }

        }
}   // end of Ising


class plotno1 extends JPanel{
int n=200;
int [][] siec = new int[n][n];
int [] forw = new int[n];
int [] back = new int[n];
int i,j,k,l,last,niter=1;
double temp=2.0;
int flaga=0;
//temp_c=2.2691853;
plotno1() {
        for (i=0;i<n-1;i++){
              forw[i]=i+1;
              back[i+1]=i;
        }
        forw[n-1]=0;
        back[0]=n-1;
        for (i=0;i<n;i++){
        for (j=0;j<n;j++){
              siec[i][j]=0*(int)(2.0*Math.random())-1;
        }
        }
}
@Override
public Dimension getPreferredSize() {
```

```
  return new Dimension(500, 500);
}
public void step(){
int iteracje;
int l1,l2;
double energia;
 for (iteracje=1;iteracje<=n*n;iteracje++){
     l1=(int)(n*Math.random());
     l2=(int)(n*Math.random());

     // TODO

     siec[l1][l2]=-siec[l1][l2];
 }
}
//************************************************************//
  public void paintComponent(Graphics g) {
  super.paintComponent(g);
  int i,j,rad=1,mnoz=2,shift=40;
      for (i=0;i<n;i++){
      for (j=0;j<n;j++){
      if(siec[i][j]==1)
         g.setColor(Color.red);
      else
         g.setColor(Color.blue);
      g.fillRect(mnoz*i-rad,mnoz*j-rad,2*rad,2*rad);
      }
      }
      if(flaga==1){

      step();
      try{Thread.sleep(30);} catch (InterruptedException exc){}

      repaint();
      }
            }//end of paint
}// end of plotno
```

Dodaj pole tekstowe (JTextField), w którym ustawiana będzie wartość temperatury (temp).

Double.valueOf(xxx.getText()) - pobiera łańcuch z pola tekstowego xxx i zamienia go na liczbę typu double.

Uzupełnij program tak aby do pliku zapisywał wartości średniego namagnesowania m w funkcji czasu t. Zrób wykres m(t) dla Temp=4, 3, 2.5, 2.2691 oraz 2.2. Konfiguracja początkowa jest w pełni ferromagnetyczna (m=1).

```
import java.io.*;

try {
    FileWriter outFile = new FileWriter("magnet5.txt",true);
   PrintWriter out1 = new PrintWriter(outFile);
   out1.println(time+"  ");
   out1.close();
} catch (IOException e){
```

```
   e.printStackTrace();
}
```

Dodaj pole tekstowe, gdzie podana będzie wartość zewnętrznego pola magnetycznego.

Uzupełnij program tak aby obliczał również średnią wartość podatności magnetycznej

```
//Simulates Ising model with Metropolis Algorithm
//calculates magnetization as a function of temperature
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class Ising_suscept{
	static int n=100;
	static int [][] siec = new int[n][n];
	static int [] forw = new int[n];
	static int [] back = new int[n];
	static double temp;
	static double h=0.0;
	//temp_c=2.2691853;
	public static void main(String [] args) {
	int i,j;
	int lrelax=2000;  // relaxation
int measurements=2000;  // measurements
double magnet;
	for (i=0;i<n-1;i++){
		forw[i]=i+1;
		back[i+1]=i;
		}
		forw[n-1]=0;
		back[0]=n-1;
		for (int itemp=0;itemp<25;itemp++){
			temp=6.+1.*itemp;
		// initial configuration
		for (i=0;i<n;i++){
		for (j=0;j<n;j++){
		siec[i][j]=1;		// ferromagnetic
//		siec[i][j]=2*(int)(2.0*Math.random())-1;
		}
		}
		// relaxation
		for (int step=0;step<lrelax;step++){
			mcstep();
		}// end of relaxation
//measurements
		magnet=0.0;
		for (int step=0;step<measurements;step++){
			mcstep();
		int mag=0;
			for (i=0;i<n;i++){
				for (j=0;j<n;j++){
					mag=mag+siec[i][j];
				}
				}
			magnet=magnet+mag;
		}  // end of measurements
		double susc;
		  try {
```

```
                            FileWriter outFile = new
FileWriter("suscept_100.txt",true);
                           PrintWriter out1 = new PrintWriter(outFile);
                           out1.println(temp+" "+1.0*magnet/(n*n)/measurements+"
");
                           out1.close();
                       } catch (IOException e){
                           e.printStackTrace();
                       }

                    System.out.println(temp+" "+1.0*magnet/(n*n)/measurements+"
");

             }  // end for itemp
} //end of main
static void mcstep(){
       int l1,l2;
       double energia;
        for (int iteracje=1;iteracje<=n*n;iteracje++){
            l1=(int)(n*Math.random());
            l2=(int)(n*Math.random());
            energia=siec[l1][l2]*(siec[l1][forw[l2]]+siec[l1][back[l2]]+
            siec[forw[l1]][l2]+siec[back[l1]][l2])+h*siec[l1][l2];
            if(Math.random()<Math.exp(-2.0*energia/temp)) siec[l1][l2]=-
siec[l1][l2];
        }
}
}   // end of Ising_autocorel
```