Paintbrush
Utwórz Paintbrush:

Metoda mousePressed tworzy punkt start (na rzecz obiektów klasy mouseEvent można wywołać metodę getX() oraz getY()).

W metodzie mouseDragged tworzymy kontekst graficzny (getGraphics()), punkt end oraz rysujemy do kontekstu graficznego linię od start do end.

Rysowanie odbywa się bez przedefiniowania metody paint.

```
//paintBrush in Java
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Paintbrush extends Applet implements MouseListener,
MouseMotionListener
{
 Point start, end;
 public void init()
 {
     setBackground(new Color(255, 204, 0));
     setForeground(new Color(51, 153, 204));
     // Add the Event Listeners
     addMouseListener(this);
     addMouseMotionListener(this);
 }
 // Capture the mousePressed event
 public void mousePressed(MouseEvent e)
 {
         // TODO
}
 // Define the other events for MouseListener
 public void mouseClicked(MouseEvent e) {}
 public void mouseEntered(MouseEvent e) {}
 public void mouseExited(MouseEvent e) {}
 public void mouseReleased(MouseEvent e) {}

 // Capture the mouseDrag event
  public void mouseDragged(MouseEvent e)
 {
     // TODO
}
 // Define the other events for MouseMotionListener
 public void mouseMoved(MouseEvent e) {}
}
```

RubberBand
Obsługa myszy, rysowanie bezpośrednio do kontekstu graficznego. Dokończ definicję metody mouseDragged, w której pobierane są aktualne współrzędne myszy, oraz rysowane są dwa prostokąty. Jeden z nich rysowany jest kolorem tła czyli zamalowuje poprzedni prostokąt. Wykorzystaj setXORMode.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
```

```
public class RubberBand extends Applet {
  int startX, startY, lastX, lastY;

  public void init() {
     setSize(400,400);
    addMouseListener(new RectRecorder());
    addMouseMotionListener(new RectDrawer());
    setBackground(Color.red);
  }

  /** Draw the rectangle, adjusting the x, y, w, h
   *  to correctly accommodate for the opposite corner of the
   *  rubberband box relative to the start position.
   */

  void drawRectangle(Graphics g, int startX, int startY,
                     int stopX, int stopY ) {
    int x, y, w, h;
    x = Math.min(startX, stopX);
    y = Math.min(startY, stopY);
    w = Math.abs(startX - stopX);
    h = Math.abs(startY - stopY);
    g.drawRect(x, y, w, h);
  }

  class RectRecorder extends MouseAdapter {

    /** When the user presses the mouse, record the
     *  location of the top-left corner of rectangle.
     */

    public void mousePressed(MouseEvent event) {
      startX = event.getX();
      startY = event.getY();
      lastX = startX;
      lastY = startY;
    }

    /** Erase the last rectangle when the user releases
     *  the mouse.
     */

    public void mouseReleased(MouseEvent event) {
      Graphics g = getGraphics();
// TODO
      drawRectangle(g, startX, startY, lastX, lastY);
    }
  }
```

```
 class RectDrawer extends MouseMotionAdapter {

   /** This draws a rubberband rectangle, from the location
    *  where the mouse was first clicked to the location
    *  where the mouse is dragged.
    */

   public void mouseDragged(MouseEvent event) {
// TODO
   }
 }
}
```

Gaz sieciowy z asymetrycznym błądzeniem. Wybór przeszkody przez zaznaczenie myszką wierzchołków prostokąta.

Dodaj etykietę (u góry) „Zaznacz lewy górny i prawy dolny narożnik”.
Zdefiniuj metody obsługujące zdarzenia: mousePressed i mouseReleased. Aktywacja przycisku „START” po zaznaczeniu przeszkody (przeszkód).

```
// dowolny wybór wierzchołków przeszkody
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class Rect_Obstacles1 extends JFrame implements ActionListener,MouseListener{
  Point punkt1,punkt2;
    int applet_width=800;
      int applet_height=800;
      Wykres2aa wykr = new Wykres2aa();
      JButton przycisk;

      Rect_Obstacles1() {
     // ustalenie tytułu okna
     super("Gaz + przeszkody");
     // ustalenie rozkładu; jeśli trzeba np:
     setLayout(new BorderLayout());
     // tworzenie komponentów np.
     przycisk = new JButton("ZAZNACZ PRZESZKODĘ");
     przycisk.addActionListener(this);
     przycisk.setEnabled(false);
     // Ustalenie własciwości komponentów,
     // np:
     // Dodanie komponentów do okna np.
     add(wykr,BorderLayout.CENTER);
     add(przycisk,BorderLayout.SOUTH);
     addMouseListener(this);
     // ustalenie rozmiarów okna, np.:
     pack();
    // pokazanie okna
     setVisible(true);
}
public void mousePressed(MouseEvent e)
{
   // TODO  - pobierz współrzędne myszy i podstaw je do punkt1
}
```

```
public void mouseClicked(MouseEvent e) {}
public void mouseEntered(MouseEvent e) {}
public void mouseExited(MouseEvent e) {}
public void mouseReleased(MouseEvent e) {
      // TODO  - pobierz współrzędne myszy i podstaw je do punkt2
      // aktywuj przycisk, ustal na nim napis na START
      // zaznacz (na czerwono) w tablicy siec prostokąt
     //  o wierzchołkach punkt1,punkt2
      // odrysuj ponownie wykr

}

public static void main(String[] args) {
   // w metodzie main tworzymy
   //obiekt naszej klasy
   // i wywołujemy konstruktor
    new Rect_Obstacles1();
}

public void actionPerformed(ActionEvent e){
    wykr.czystart=true;
      wykr.repaint();
}

} // koniec klasy GuiSwing

class Wykres2aa extends JPanel{
       boolean czystart=false;
       int wielkosc=400;
       int [][] siec = new int [wielkosc][wielkosc];
       int lsquare=150;
       int lczastek=lsquare*lsquare;
       int [][] walker = new int [lczastek][2];
       public Wykres2aa(){
              setBorder(BorderFactory.createLineBorder(Color.blue));
                        for (int i=0;i<wielkosc;i++){
                           for (int j=0;j<wielkosc;j++){
                                  siec[i][j]=0;
                        }
                        }

                        for (int i=0;i<lsquare;i++){
                           for (int j=0;j<lsquare;j++){
                     siec[i+(wielkosc-lsquare)/2][j+(wielkosc-lsquare)/2]=1;
                     walker[i*lsquare+j][0]=i+(wielkosc-lsquare)/2;
                     walker[i*lsquare+j][1]=j+(wielkosc-lsquare)/2;
                        }
                        }



                        }
       @Override
       public Dimension getPreferredSize()
       {return new Dimension(wielkosc,wielkosc);
       }
      void step (int ktory){
        int x=walker[ktory][0];
```

```
   int y=walker[ktory][1];
              double losowa=Math.random();
              if(losowa<0.35) {x=x+1;if (x==wielkosc){x=0;}}
              else if(losowa<0.5) {x=x-1;if (x==-1){x=wielkosc-1;}}
              else if(losowa<0.75) {y=y+1;if (y==wielkosc){y=0;}}
              else {y=y-1;if (y==-1){y=wielkosc-1;}}
        if(siec[x][y]==0){
         siec[walker[ktory][0]][walker[ktory][1]]=0;
         walker[ktory][0]=x;
         walker[ktory][1]=y;
         siec[x][y]=1;
        }
}
public void paintComponent(Graphics graf){
       super.paintComponent(graf);
  for (int i=0;i<wielkosc;i++){
       for (int j=0;j<wielkosc;j++){

if(siec[i][j]==1){graf.setColor(Color.BLACK);graf.drawRect(i,j,1,1);}

if(siec[i][j]==5){graf.setColor(Color.RED);graf.drawRect(i,j,1,1);}
  }
  }
     if(czystart){
              for (int czastka=0;czastka<lczastek;czastka++){
                     int indeks=(int)(lczastek*Math.random());
                     step(indeks);
              }
         try {
             Thread.sleep(20);                // sleep for 10 msec
           } catch (InterruptedException t){}
       repaint();
     }
       }
```