Model Eden

Uzupełnij definicję metody addborder(int i,int j)**, która dodaje (jeżeli trzeba) węzeł [i][j] do tablicy** boder**.**

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class eden_cluster extends Applet implements ActionListener {
    int n;
    int size=500;     //applet size
    int square_size=5;
    int lsize=size/square_size;   // lattice size
    TextField tfield;
    int [][] lattice = new int [lsize][lsize];
    // lattice=0 virgin
    // lattice=1 ocuppied
    // lattice=2 border
    int lborder;
    int [][] border = new int [lsize*lsize/10][2];
      public void init(){
            resize(size,size);
            setBackground(Color.yellow);
            add(new Label("number of steps"));
            tfield = new TextField(10);
            add(tfield);
            tfield.addActionListener(this);
      }  // end of init
public void paint(Graphics g){
            for (int i=0;i<lsize;i++){
                  for (int j=0;j<lsize;j++){
                        if(lattice[i][j]==1) {g.setColor(Color.blue);}
                        if(lattice[i][j]==2) {g.setColor(Color.black);}
                        if(lattice[i][j]>0) g.fillRect(square_size*i,
square_size*j, square_size, square_size);
                  }}
            if ((n>0)&&(step<n)){
                  step++;
                         int k=(int)(Math.random()*(lborder+1));
                         int i = border[k][0];
                         int j = border[k][1];

   // TODO
//  ostatni element z tablicy border przesuń na k-te miejsce
//  miejsce o współrzędnych i,j ustaw na 1 (occupied)
//  sprawdź czy któryś z sąsiadów nie staje się brzegowy

                try {
                    Thread.sleep(10);               // sleep for 10 msec
                  } catch (InterruptedException t){}
            repaint();
                  }
      } // end of paint

       public void actionPerformed(ActionEvent e){
             n=Integer.parseInt(tfield.getText());
             // konfiguracja początkowa
             for (int i=0;i<lsize;i++){
                         for (int j=0;j<lsize;j++){
```

```
                                lattice[i][j]=0;
                            }}
                        int k=lsize/2;
                         lborder=0;
                         border[0][0]=k;
                         border[0][1]=k;
                         step=0;
                    repaint();
                }   // end of actionPerformed
               void addborder(int i,int j){
// TODO
               } // end of addborder
}  // end of eden_cluster
```

Rozszerzenia:

1) Zmodyfikuj konfigurację początkową, tak aby wzrost klastra rozpoczynał się od górnej krawędzi. Dodaj poziome periodyczne warunki brzegowe.

2) Wprowadź tablicę wiek[][], która przechowywać będzie informację o wieku (step) dołączonej do klastra cząstki. Klaster odrysuj w kolorach: najstarsze na niebiesko, najmłodsze na czerwono (np. g.setColor(new Color(r,0,255-r));}).

3) W wersji gdzie początkowo klaster umieszczony jest na górnej krawędzi wprowadź aktywność węzła (0<aktywność[][]<1). Wybrany z tablicy border węzeł jest akceptowany z prawdopodobieństwem aktywność[][]. W przypadku odrzucenia, nowy węzeł brzegowy jest losowany. Aktywność jest dziedziczna: dodając węzeł do brzegu, przepisujemy aktywność z węzła, który aktywował ten dodawany węzeł.