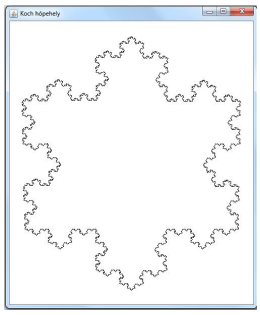
Koch hópehely



Készítsen alkalmazást a Koch-hópehely alakulásának bemutatására. (http://hu.wikipedia.org/wiki/Koch-görbe).

Az alapelv: kiindulunk egy szabályos háromszögből. Ez a 0. szint.

A következő lépésben a háromszög minden oldalát három részre bontjuk, és a középső rész fölé egy egyharmad oldalhosszúságú szabályos háromszöget rajzolunk úgy, hogy kimarad az eredeti vonalon lévő szakasz. Ez az 1. szint. A továbbiakban az előző szakaszban leírtakat ismételjük az összes vonalra.

Animált módon a háromszögből indulva jusson el mondjuk az 5. vagy 6. szintig (tovább nem nagyon érdemes, mert túlzottan forrásigényes a rekurzió), majd csökkenjen vissza a háromszögig, aztán megint jusson el az utolsó szintig, megint csökkenjen vissza, stb.

Megoldásrészlet:

A frame felületére rákerül egy panel, erre a panelre rajzoljuk a hópelyheket, és mivel animálni szeretnénk, ezért célszerű szálként kezelni a panelt. (Létezik más megoldási mód is, de most a cél a szálkezelés gyakorlása.)

A panel kódja (generált részek nélkül):

```
public class HopehelyPanel extends javax.swing.JPanel implements Runnable {

    private int szint = -1;
    private Thread szal;
    private boolean fut = true;

    public HopehelyPanel() {
        initComponents();
    }
```

```java
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    int keret = 30;
    int alsoKeret = 150;
    int szel = this.getWidth() - 2 * keret;
    int mag = this.getHeight() - keret - alsoKeret;

    int x1 = keret, y1 = keret + mag;
    int x2 = keret + szel, y2 = keret + mag;
    int x3 = keret + szel / 2, y3 = keret;

    if (szint > -1) {
        hopehelyVonalRajz(g, szint, x1, y1, x2, y2);
        hopehelyVonalRajz(g, szint, x2, y2, x3, y3);
        hopehelyVonalRajz(g, szint, x3, y3, x1, y1);
    }
}

private void formAncestorAdded(javax.swing.event.AncestorEvent evt) {
    if (szal == null) {
        szal = new Thread(this);
        szal.start();
    }
}

private void hopehelyVonalRajz(Graphics g, int szint, int x1, int y1,
                               int x5, int y5) {
    int deltaX, deltaY, x2, y2, x3, y3, x4, y4;

    if (szint == 0) {

        g.drawLine(x1, y1, x5, y5);
    } else {
        deltaX = x5 - x1;
        deltaY = y5 - y1;

        x2 = x1 + deltaX / 3;
        y2 = y1 + deltaY / 3;

        x3 = (int) (0.5 * (x1 + x5) + Math.sqrt(3) * (y1 - y5) / 6);
        y3 = (int) (0.5 * (y1 + y5) + Math.sqrt(3) * (x5 - x1) / 6);

        x4 = x1 + 2 * deltaX / 3;
        y4 = y1 + 2 * deltaY / 3;

        hopehelyVonalRajz(g, szint - 1, x1, y1, x2, y2);
        hopehelyVonalRajz(g, szint - 1, x2, y2, x3, y3);
        hopehelyVonalRajz(g, szint - 1, x3, y3, x4, y4);
        hopehelyVonalRajz(g, szint - 1, x4, y4, x5, y5);
    }
}
```

```
@Override
public void run() {
    while (fut) {
        while (szint < 6) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                Logger.getLogger(HopehelyPanel.class.getName()).
                                                log(Level.SEVERE, null, ex);
            }
            szint++;
            this.repaint();
        }
        while (szint > 0) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ex) {
                Logger.getLogger(HopehelyPanel.class.getName()).
                                                log(Level.SEVERE, null, ex);
            }
            szint--;
            this.repaint();
        }
    }
}
```

Megjegyzés: Ebbe a kódba most viszonylag sok konstans van „beégetve”, amit általában nem szabad csinálni, hiszen programot az esetek túlnyomó többségében azért írunk, hogy sok hasonló esetre is lehessen alkalmazni, vagyis muszáj, hogy könnyen változtatni tudjuk a benne lévő konstansokat. Ez a feladat azonban nagyon speciális, inkább egy fogalom illusztrálására szolgál, lényegében fix adatokkal.