Programozás III

SZÁL PÉLDÁK -
folytatás

EGY KIS KITÉRŐ

MOZGÁS-VARIÁCIÓK

Két pont közötti egyenes mozgás

A két pont: (kx, ky), (vx, vy).

Az aktuális pont: (aktx, akty).

MOZGÁS-VARIÁCIÓK

```
@Override
public void run() {
    double tavolsag = Math.sqrt((vy - ky)*(vy - ky) + (vx - kx)*(vx - kx));
    if (tavolsag > 0 && lepes >0) {
        double aktualisTav = 0;
        while (aktualisTav < tavolsag) {
            try {
                aktx = (int) (kx + (vx - kx) / tavolsag * aktualisTav);
                akty = (int) (ky + (vy - ky) / tavolsag * aktualisTav);
                aktualisTav += lepes;

                Thread.sleep(ido);
                szalVezerlo.frissit();
            } catch (InterruptedException ex) {
                Logger.getLogger(KekSzal.class.getName()).log(Level.SEVERE,
            }
        }
    }
}
```

kék

MOZGÁS-VARIÁCIÓK

```
@Override
public void run() {
    double szog = Math.atan2(vy - ky, vx - kx);
    double tavolsag = Math.sqrt((vy - ky)*(vy - ky) + (vx - kx)*(vx - kx));
    if (tavolsag > 0 && lepes > 0) {

        for (double tav = 0; tav <= tavolsag; tav += lepes) {
            try {
                aktx = (int) (kx + tav * Math.cos(szog));
                akty = (int) (ky + tav * Math.sin(szog));

                Thread.sleep(ido);
                szalVezerlo.frissit();
            } catch (InterruptedException ex) {
                Logger.getLogger(PirosSzal.class.getName()).log(Level.SEVERE
            }
        }
    }
}
```

piros

MOZGÁS-VARIÁCIÓK

```
@Override
public void run() {
    double tavolsag = Math.sqrt((vy - ky)*(vy - ky) + (vx - kx)*(vx - kx));
    if (tavolsag > 0 && lepes > 0) {
        double lepesSzam = tavolsag/lepes;
        double dx = (vx-kx)/lepesSzam;
        double dy = (vy-ky)/lepesSzam;

        for(int i=0; i<= lepesSzam; i++) {
            try {
                aktx = (int) (kx + i * dx);
                akty = (int) (ky + i * dy);

                Thread.sleep(ido);
                szalVezerlo.frissit();
            } catch (InterruptedException ex) {
                Logger.getLogger(ZoldSzal.class.getName()).log(Level.SEVERE,
            }
        }
    }
}
```

zöld

VÁJTFÜLŰEKNEK (IGÉNYESEKNEK)

A szálaknál van jobb megoldás:

A java.util.concurrent csomag részét alkotó végrehajtó keretrendszer (Executor Framework) – ez rugalmasabb a szálnál.

```
private List<Szal> szalak = new CopyOnWriteArrayList<>();

private ExecutorService executorService=Executors.newSingleThreadExecutor();
```

Léteznek más végrehajtó típusok is

(pl. newCachedThreadPool(), stb.)

(A szalak lista a kirajzolás miatt kell.)

VÁJTFÜLŰEKNEK (kipróbálandó!)

```
PirosSzal pirosSzal = new PirosSzal(kx, ky, vx, vy, r, lepes, ido, this);
KekSzal kekSzal = new KekSzal(kx, ky, vx, vy, r, lepes, ido, this);
ZoldSzal zoldSzal = new ZoldSzal(kx, ky, vx, vy, r, lepes, ido, this);
szalak.add(pirosSzal);
szalak.add(kekSzal);
szalak.add(zoldSzal);

//      pirosSzal.start();
//      kekSzal.start();
//      zoldSzal.start();

executorService.execute(kekSzal);
executorService.submit(pirosSzal);
executorService.execute(zoldSzal);
```

VÁJTFÜLŰEKNEK (kipróbálandó!)

A végrehajtó szolgáltatással sok mindent meg lehet oldani, pl.

- meg lehet várni, amíg egy szolgáltatás véget ér;
- vagy amíg egy feladatcsoport véget ér;
- egyesével kinyerhetjük az egyes feladatok eredményét, stb.

A wait – notify is kiváltható

VÁJTFÜLŰEKNEK

Irodalom:

Joshua Bloch: Hatékony Java, Bp., Kiskapu Kft., 2008.

http://docs.oracle.com/javase/tutorial/essential/concurrency/exinter.html

http://www.vogella.com/articles/JavaConcurrency/article.html

http://winterbe.com/posts/2015/04/07/java8-concurrency-tutorial-thread-executor-examples/

http://java-latte.blogspot.hu/2012/11/executor-framework-executor-interface.html

stb.