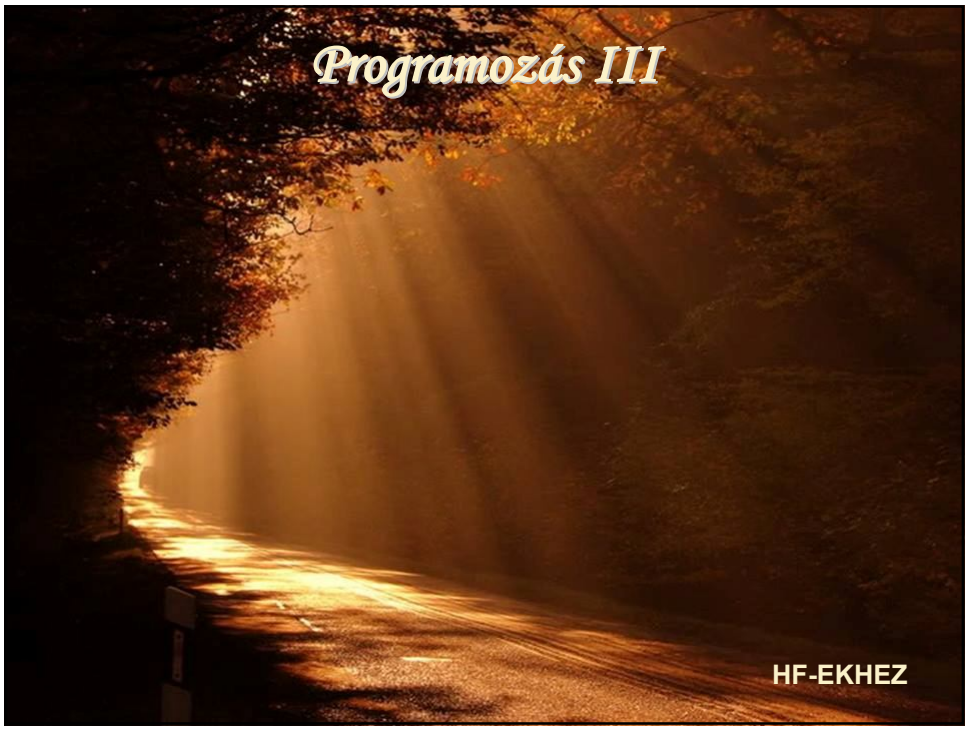


EGY KIS ISMÉTLÉS

EGY KIS ISMÉTLÉS

Szükséges osztályok:

JFrame – szerepe:	main() + egymáshoz rendelés
Rajzpanel :	kirajzoltatás
Vezérlő panel:	események vezérlése, adminisztráció
Vezérlő:	Az egész program irányítása
Labda:	☺

EGY KIS ISMÉTLÉS

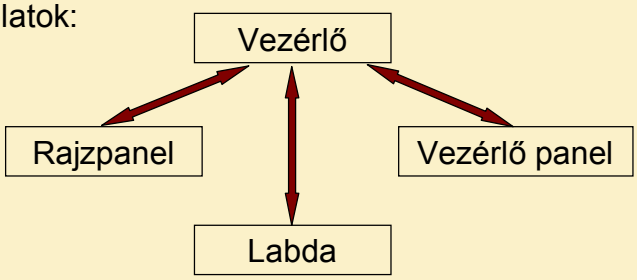
Labda osztály:

– Szál, azaz vagy extends Thread
vagy implements Runnable

Tiszta kód: az osztályok legyenek minél függetlenebbek egymástól.

EGY KIS ISMÉTLÉS

Kapcsolatok:



Elvileg a labdának szüksége lenne a rajzpanelre, de nem muszáj közvetlenül elérnie, rábízhatja a frissítést a vezérlőre.

EGY KIS ISMÉTLÉS

Logikus lenne, ha minden lényegi dolgot a vezérlő végezne, vagyis a beolvasás, adatkezelés is az ő dolga, a paneleké csak a megjelenítés. (Persze hivatkozhat egyéb osztályokra is.)

Rajzpanel: … vezerlo.rajzol(g);

Vezérlő: rajzPanel.frissit();

Vezérlőpanel: vezerlo.allj();

Vezérlő: vezerloPanel.listabaIr(…);

Ebben a feladatban a labdák bizonyos időközönként keletkeznek. Hogyan?

Ki viselkedjen szálként?

NÉHÁNY ÉSZREVÉTEL

Illik tisztában lenni az elemi geometria legalapvetőbb fogalmaival ☺:

pont koordinátái, polár-koordináták;

egyenes egyenlete – egyenes vonalú mozgás két pont között

kör egyenlete, de legalábbis két pont távolságának meghatározása, stb.

A paintComponent() metóduson belül
TILOS a repaint() hívás!!!

NÉHÁNY ÉSZREVÉTEL

Alap-osztályban NEM hivatkozunk a Global osztály adataira! Helyette: mezők (esetleg statikus) setterrel.

Szálpéldányokat tartalmazó lista típusa:

CopyOnWriteArrayList<T>

De ha nem szál-példányok, akkor ArrayList<T>

NÉHÁNY KONKRÉT ÉSZREVÉTEL

NEM mindig while(fut).
A boszorkányok pl. addig működjenek, amíg átértek.
Tovább ne. (while(balX < feluletSzelesseg))

~~rajzPanel.getVezerlo().getMadarak().remove(this);~~

vagy ~~vezerlo.getMadarak().remove(this);~~ NE!!

Fordítva, ez az osztály kérje meg a másikat, hogy az átadott adatokkal csináljon valamit.

pl. vezerlo.torol(this);

NÉHÁNY KONKRÉT ÉSZREVÉTEL

Egyenes vonalú egyenletes mozgás:
A dx, dy értékeket NEM kívülről adjuk meg!!
Bemenő adat: kezdőpont és végpont koordinátái.

Szerintem a legegyszerűbb változat a mintapéldaként kirakott mozgás_teszttel.

Nem biztos, hogy minden egyszerűen megoldható egyetlen mozgas() metódusban.
pl lehet odamegy(), tancol(), visszamegy().

NÉHÁNY KONKRÉT ÉSZREVÉTEL

Apróság, de ehelyett:

```
metódus(){
   vezerlo.csinald1();
   vezerlo.csinald2();
}
```

jobb lenne így:

```
metódus(){
   vezerlo.csinald();
}
```

és a vezérlő csinald() metódusából meghívni a csinald1-csinald2-t.

NÉHÁNY KONKRÉT ÉSZREVÉTEL

Ebben sajnos én vagyok a hibás, bár utólag korrigáltam magam: A toURI() a fajta elérési út nem működik jar-ból, csak az inputstream-en keresztül való olvasás.

Szerintem fölösleges ennyi panel, bár éppen lehet is, de a háttércseréket simán meg lehetne oldani három háttérkép cserélgetésével.

Kb X százalék véletlen kiválasztása:

Végigmegyünk egy ciklussal az összes lakón, és ha Math.random() < szazalek, akkor az illető boszi, egyébként nem.

NÉHÁNY KONKRÉT ÉSZREVÉTEL

Úgy tűnik, már úgy fejezem be a pályámat, hogy nem értem meg, miért egyszerűbb

ez:

```
while(valami){
 ...
  if(feltétel) valami = true;
}
```

ehelyett:

```
while(feltetel){
  ...
}
```