Java Thread labor

Készítette: László Zoltán, BME IIT, 2015.

A feladatok megoldásához felhasználandó osztályok (Thread, Object, stb) leírásait az alábbi URL-en találja meg:

http://download.oracle.com/javase/8/docs/api/

1 Producer osztály

Készítsen egy **Producer** osztályt! Az osztálynak legyen egy **void go()** metódusa, amely végtelen ciklusban másodpercenként üzenetet ír ki a standard outputra, a konstruktorában pedig vegyen át egy Stringet.

A kipróbáláshoz készítsen egy **Application** osztályt, aminek a **main** metódusa (a program belépési pontja) elindít egy **Producer**-t!

Korlátozások:

- a várakozást a *Thread* osztály *sleep* metódusával valósítsa meg, mert így az objektum nem tartja magánál a CPU-t, nem történik foglalva várakozás (*busy-wait*).
- a kiírt üzenet a következő formátumú legyen:
  `<szoveg> <szam> <ido>`
  ahol
  `<szoveg>` az objektum konstruktorában átadott String
  `<szam>` a kiírás sorszáma (0-ról indul)
  `<ido>` a kiírás ideje (a millisec-ben mért rendszeridő utolsó 5 jegye, javaslat: *Date* object *getTime()* , vagy *System.currentTimeMillis()* metódus)

  Egy példa futás eredménye:

```
demo 0 69562
demo 1 70562
demo 2 71562
...
```

2 Producer osztály szállal

Módosítsa úgy a fenti **Producer**-t, hogy önálló szálként tudjon futni! A két tanult lehetőség közül használja az öröklést!
Módosítsa úgy az **Application** osztályt, hogy a **main** metódus fél másodperc különbséggel elindít két **Producer** szálat!
Egy példa futás eredménye:

```
elso 0 54796
masodik 0 55281
elso 1 55796
masodik 1 56296
elso 2 56796
masodik 2 57296
...
```

3 Fifo puffer

Készítsen egy maximum 10 elemű, Stringeket tároló, konkurens működésű **Fifo** osztályt, amelynek metódusai:

- **void put(String)**. A paraméterként kapott Stringet berakja a *Fifo*ba. Ha a *Fifo* tele van, akkor a futtató szál korlátlan ideig várakozik és csak akkor ébred fel, ha a puffer tartalma változik.
- **String get()**. Visszaadja és a *Fifo*ból törli a legöregebb elemet. Ha a *Fifo* üres, akkor a futtató szál korlátlan ideig várakozik és csak akkor ébred fel, ha a puffer tartalma változik.

Javaslat: a tároláshoz használja a *LinkedList<String>* osztályt.

A Fifo puffer legyen szálbiztos, vagyis egy időben csak egy szál futtathassa a metódusait! Minden kötelezően kezelendő kivételt (*InterruptedException*) adjon tovább a hívónak!
Ne felejtse el, hogy a várakozással és értesítéssel kapcsolatos metódusokat csak akkor szabad meghívni, ha a futó szálnak kizárólagos futtatási joga van a használt objektumra (belépett az objektum monitorába).

Módosítsa a 2. feladatban készült *Producer* osztályt az alábbiak szerint!

- A konstruktorban a *Producer* kap egy *Fifo*t is.
- A <szöveg> <szám> üzenetet tegye be a *Fifo*ba *put* metódussal!
- A *Fifo*ba történő beírást követően a standard outputra írja ki:

```
produced <szoveg> <szam> <ido>
```

4 Consumer osztály

Készítsen egy **Consumer** osztályt a *Fifo* végtelen ciklusban történő olvasására!

- A *Consumer* szálként tudjon futni
- A konstruktorban kap egy **f** *Fifo*t, egy **s** Stringet és egy **n** intet.
- Consumer az *f* getjét használva stringeket olvas. Az olvasást követően a standard inputra írja ki:

```
consumed <s> <olvasott string> <ido>
```

- Ezután az **n** paraméternek megfelelő ideig alszik, majd újra kezdi a ciklust.

Módosítsa az *Application main*-jét, amelyben létrehoz egy *Fifo*t, majd külön-külön szálon elindít egy-egy *Producer*t és *Consumer*t!

5 Sebességarányok

Módosítsa úgy a *Producer*t, hogy konstruktor paramétéként meg lehessen adni a várakozási idejét! Indítsa a *Producer* és a *Consumer* objektumokat eltérő várakozások értékekkel! Mit tapasztal? Ha hibát észlel, javítsa az algoritmust.

6 Szálkezelés delegálással

A *Producer*t írja át úgy, hogy a Thread-ből való származtatás helyett a **Runnable** interfészt implementálja! Próbálja ki az alkalmazást a 4. feladat szerint!

7 Több termelő és fogyasztó

Az *Application*ben példányosítson több *Producer*t és *Consumer*t!
Legyen 3 db Producer 1000-1000 ms várakozási idővel, és 4 db Consumer 100-100ms várakozási idővel. Indítsa őket közvetlenül egymás után! Mit tapasztal? Ha problémát észlel, akkor javítsa a hibásan implementált algoritmust!

Fix várakozási idők helyett próbálkozhat véletlen időkkel is. Véletlen szám generálásához használja a **Math.random()** metódust, vagy a **java.util.Random** osztályt!

8 Szálak jellemzői

Módosítsa a *Fifo* osztályt az alábbiak szerint!

A *Fifo* osztályban mind a *put*, mind a *get* metódus írja ki a metódus nevét, valamint a metódust futtató szál azonosítóját! Használja a *Thread* osztály metódusait!