Gijos (sinchronizavimas)

Vykdomos gijos gavimas

Norėdami gauti vykdomą giją, ją galime gauti šiais būdais:

Iš thread klasės:

```
currentThread();
```

Iš bet kurios kitos klasės

```
Thread.currentThread();
```

Norėdami gauti arba pakeisti gijos pavadinimą, tai galime atlikti metodais:

```
setName("name");
getName();
```

Interrupt() ir sleep() metodai

Kiekvieną giją mes galime patalpinti į laukimo būseną panaudoję sleep metodą. Pvz.: `gija.sleep(1000);`

Tačiau mes privalome visuomet užmigdymo metodą apglėbti try..catch bloku, kadangi jį gali iškviesti pagrindinė programa arba kita gija:

```
try {
   gija.sleep(1000);
} catch (InterruptedException e) {
   //Veiksmai vykdomi jei gija pažadinta ankščiau nei reikia
}
```

Giją atgaivinti galime metodo interupt() pagalba, pavyzdžiui: `gija.interupt();`

Gijų apjungimas

Mes galime padaryti giją kuri pasileistų tik tuomet kai prieš tai buvusi gija baigs darbą, panaudodami join() metodą.

```
public class Irasymas extends Thread {
  private Thread skaiciavimai;
  public Irasymas(Thread skaiciavimai) {
      this.skaiciavimai=skaiciavimai;
  }

  public void run() {
    try {
      this.skaiciavimai.join();
    } catch (InterruptedException e) {
      System.out.println("Kazkas mane nutrauke");
    }
    System.out.println("Baigesi skaiciavimai pradesiu irasyma");
  }
}
```

Join() metodas

Join () metodas bus vykdomas iki tol kol parinkta gija bus nutraukta

Taip pat galime nurodyti jog gija testų darbą po tam tikro laiko:

join(1000) //Tęsti darbą kai gija baigsis, arba po 1 sekundės

Konkurencinės gijos

```
class Countdown{
  public void count(){
    for (int i=10; i>0; i--)
     System.out.println(
       Thread.currentThread().getName()+
       ": i="+i);
  }
}

class CThread extends Thread{
 private Countdown tc;
 public CThread(Countdown tc){ this.tc=tc; }
 public void run(){  tc.count(); }
}
```

Paleidimo kodas

```
Countdown cd=new Countdown();
CThread t1=new CThread(new Countdown());
t1.setName("Gija1");
CThread t2=new CThread(new Countdown());
t2.setName("Gija2");
t1.start();
t2.start();
}
```

Užduotis

Pamėginkime abiem objektams paduoti vieną ir tą patį Coundown objektą.

Pamėginkime skaičiavimo skaičių iškelti iš metodo į objekto parametrus.

Metodų sinchronizavimas

Norėdami jog kita gija nenutrauktų tam tikro metodo (metodas būtų įvykdytas nuo pradžios iki galo) galime sinchronizuoti visą metodą pridedant žodelį synchronized:

```
public synchronized void metodoPavadinimas(){
  //Kodas bus vykdomas iš eilės
}
```

Arba galime sinchronizuoti pagal kintamąjį (ne lokalų):

```
synchronized (mutex) {


}
```

Užduotis

Pakoreguoti programą taip kad būtų atvaizduoti visi skaičiai nuo 10 iki 1 (nepasikartojant).

Gamintojas ir naudotojas

Gamintojo ir naudotojo problema (angl. Producer and Consumer).

Tam kad dalinai išvegtumėme aklaviečių galime naudoti wait(), notify() ir notifyAll() metodus.

wait() – metodas patalpina giją į laukiančiųjų resurso sąrašą

notify() – pažadina visus metodus (nebutinai juos vykdo)

notifyAll() - pažadina visus metodus