Java – S06

Alin Zamfiroiu

Java – S06

Continut:

- Threads
- Runnable
- Synchronize

S06 – Threads

- Se creeaza clasa **Notificare** cu 3 atribute: **mesaj**, **nr_notificari** si **pauza**.

- Aceasta clasa are o metoda notifica(). Aceasta metoda afiseaza mesajul **mesaj** de un numar de **nr_notificari** ori.

S06 – Threads

```
public void notifica(){
    for(int i=0;i<number;i++){
        System.out.println(i+".   "+mesaj+".");
    }
}
```

```
public Notificare(String _mesaj, int number, int pauza){
    mesaj=_mesaj;
    this.number=number;
    this.pauza=pauza;
}
```

```
public class Notificare {

    private String mesaj;
    private int number;
    private int pauza;
```

S06 – Threads

- Folositi metoda statica sleep din clasa Thread pentru a afisa mesajele cu o pauza egala cu valoarea atributului **pauza** intre ele.

In programul principal sunt definite 2 obiecte de tip **Notificare** si apelati metoda **notifica** pentru acestea

S06 – Threads

```
try {
    Thread.sleep(pauza);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

S06 – Threads

- Realizati o alta clasa numita **Notificare2** asemanatoare cu prima clasa.

- Diferenta este ca aceasta extinde clasa **Thread**.

```
public class Notificare2 extends Thread{
```

Si metoda publica de tip void: **run**().
In cadrul metodei run(), apelam metoda notifica().

S06 – Threads

```
public static void main(String[] args) {
    Notificare2 notificare=new Notificare2(
            "Adresa de mail a fost schimbata",
            20,500);
    Notificare2 notificare2=new Notificare2(
            "Parola a fost schimbata",
            10,1000);
    notificare.start();
    notificare2.start();
```

S06 – Runnable

- Sa se realizeze o alta clasa Notificare3.
- Aceasta nu extinde Thread, ci implementeaza interfata Runnable.

- In programul principal se creaza doua fire de executie pe baza a doua obiecte de tip Notificare3.

S06 – Runnable

```
Thread t1=new Thread(notificare);
t1.start();
Thread t2=new Thread(notificare2);
t2.start();
```

S06 – ExectorService

- In programul principal sa se creeze un obiect de tipul **ExecutorService** prin intermediul metodei statice ***newFixedThreadPool*** din clasa **Executors**.
- Metoda primeste ca si parametru numarul de fire de executie.
- Adaugarea obiectelor de tip **Notificare3** se face prin intermediul metodei **execute**().

S06 – ExectorService

Clasa este Notificare sau Notificare3

```
int nrThreads=4;
ExecutorService es=Executors.newFixedThreadPool(nrThreads);
for(int i=0;i<nrThreads;i++)
{
    es.execute(new Notificare(i+"---"+i, 10*(i+1), 100/(i+1)));
}
es.shutdown();
es.execute(new Notificare("gggggg", 10, 10));
```

Clasa este Notificare sau Notificare3

S06 – ExectorService

- Afisati ceva dupa terminarea rularii tuturor firelor de executie.

- Se foloseste metoda **awaitTermination()** pentru asteptarea terminarii tuturor firelor de executie.

S06 - ExectorService

```
try {
    es.awaitTermination(Integer.MAX_VALUE, TimeUnit.NANOSECONDS);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
System.out.println("MESSSSSSSSSSSSSSSSSSSSSSSAJ");
```

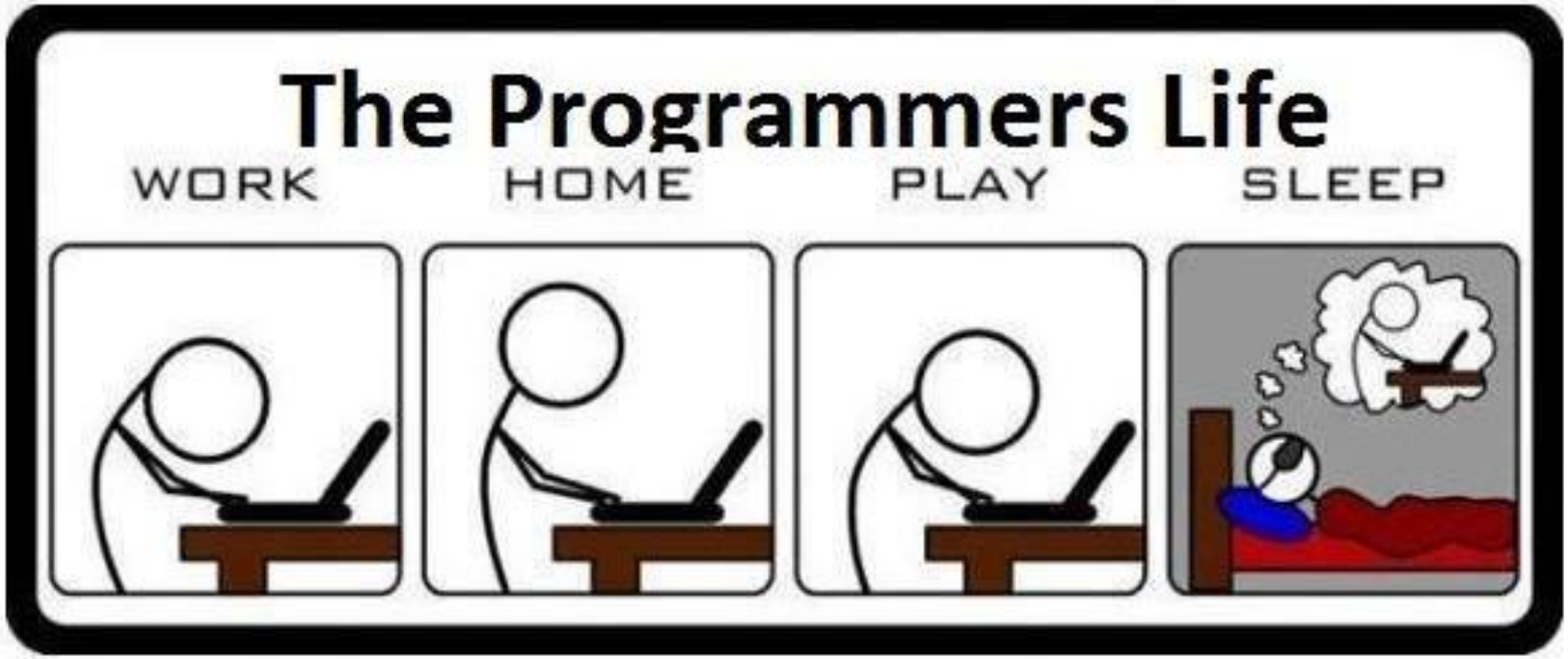
S06 – Synchronize

- Sa se realizeze clasa Notificare4 asemanatoare cu cele de pana acum.
- Clasa Notificare4 are un atribut static de tip Object.
- In cadrul metodei notifica() sa se sincronizeze operatiile realizate.

S06 – Synchronize

```
public void notifica(){
    synchronized (object) {
        for(int i=0;i<number;i++){
            System.out.println(i+".  "+mesaj+".");
            try {
                Thread.sleep(pauza);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

O zi frumoasa, in continuare!



http://cjug.org/page/2/