**Capitolul 2**

**2.1. Gruparea thread-urilor (firelor de execuție). Prioritatea firelor de execuție.**

## 2.1.1. Gruparea firelor de execuţie

Gruparea firelor de execuţie pune la dispoziţieun mecanism pentru manipularea acestora ca un întreg. De exemplu, putem să lansăm sau să suspendăm toate firele dintr-un grup cu un singur apel la metodă. Gruparea firelor de execuţie se realizează prin intermediul clasei ThreadGroup.

Fiecare fir de execuţie Java este membru al unui grup, indiferent dacă specificăm explicit acest lucru. Afilierea unui fir de execuţie la un anumit grup, se realizează la crearea acestuea şi devine permanentă, deci nu vom putea muta un fir de execuţie dintr-un grup în altul, după ce acesta a fost creat. În cazul în care, creăm un fir de execuţie fără a specifica în constructor din ce grup face parte, el va fi plasat automat în acelaşi grup cu firul de execuţie care l-a creat. La pornirea unui program Java se creează automat un obiect de tip **ThreadGroup** cu numele **main**, care va reprezenta grupul tuturor firelor de execuţie create direct din program, şi care nu au fost ataşate explicit altui grup. Se poate ignora complet plasarea firelor de execuţie în grupuri şi lăsînd sistemul să se ocupe de aceasta, plasîndu-le pe toate în grupul **main**. Există situaţii când programul creează o mulțime de fire de execuţie, iar gruparea lor poate uşura substanţial manevrarea acestora. Crearea unui fir de execuţie şi plasarea lui într-un grup (altul decât cel implicit) se realizează prin următorii constructori ai clasei Thread:

public Thread(ThreadGroup group, Runnable target)

public Thread(ThreadGroup group, String name)

public Thread(ThreadGroup group, Runnable target,

 String name)

Fiecare din aceşti constructori creează un fir de execuţie, îl iniţializează şi îl plasează într-un grup specificat în argumentul acestuea. În exemplul următor vor fi create două grupuri, primul cu două fire de execuţie, iar al doilea cu trei fire de execuţie:

ThreadGroup grup1 = new ThreadGroup("Producatori");

Thread p1 = new Thread(grup, "Producator 1");

Thread p2 = new Thread(grup, "Producator 2");

ThreadGroup grup2 = new ThreadGroup("Consumatori");

Thread c1 = new Thread(grup, "Consumator 1");

Thread c2 = new Thread(grup, "Consumator 2");

Thread c3 = new Thread(grup, "Consumator 3");

Pentru a afla cărui grup aparţine un anumit fir de execuţie putem folosi metoda getThreadGroup a clasei Thread. Un grup poate avea ca părinte un alt grup, ceea ce înseamnă că firele de execuţie pot fi plasate într-o ierarhie de grupuri, în care rădăcina este grupul implicit main, după cum este arătat în Figura 3.

Figura 3 Gruparea firelor de execuţie

Exemplu: Enumerarea firelor de execuţie active

public class EnumerateTest {

 public void listCurrentThreads() {

 ThreadGroup currentGroup =
 Thread.currentThread().getThreadGroup();

//aflu numarul firelor de execuţie active

 int numThreads = currentGroup.activeCount();

//pun intr-un vector referinţe la firele de exec.
//active

 Thread[] listOfThreads = new Thread[numThreads];

 currentGroup.enumerate(listOfThreads);

 //le afisez pe ecran

 for (int i = 0; i < numThreads; i++)

 System.out.println("Thread #" + i + " = " +

 listOfThreads[i].getName());

 }

}

Structura firelor de execuţie şi grupurilor de execuţie poate fi reprezentată în următoarea forma:

MainGrupa{subGrupa1{subGrupa3{Tha, Thb, Thc, Thd}, ThA},subGrupa2{Th1, Th2, Th3},Th1, Th2}.

##### 2.1.2.Prioritatea firelor de execuție.

Prioritatea firelor de execuție este un număr întreg, care determină prioritatea relativă a unui fir față de altul.

Fiecare fir de execuţie are o prioritate cuprinsă între valorile MIN\_PRIORITY şi MAX\_PRIORITY. Aceste două variabile finale sunt declarate în clasa **Thread** . În mod implicit, un fir de execuţie are prioritatea NORM\_PRIORITY, de asemenea definită în clasa **Thread** .

Mediul de execuţie Java ordoneaza firele de execuţie pentru preluarea controlul unităţii centrale în funcţie de prioritatea lor. Dacă există mai multe fire cu prioritate maximă, acestea sunt ordonate după un algoritm numit ”round-robin”. Firele cu prioritate mai mică sunt prelucrate doar atunci când toate firele cu prioritate mare sunt în starea **”Nu Rulează”** .

Prioritatea unui fir de execuţie poate fi determinată prin metoda **getPriority().** Metoda **getPriority()** returnează prioritatea firului de execuție, care este un număr întreg și reprezintă prioritatea curentă a firului de execuţie. Pentru a stabili prioritatea, se foloseşte metoda **setPriority()** și primeşte ca parametru un număr întreg, care reprezintă prioritatea dorită.

Pentru a modifica prioritatea grupului de fire de execuție se folosește metoda setMaxPriority().

Schimbarea priorităţii unui fir de execuţie este periculoasă, dacă metoda cu prioritate mare nu se termină foarte repede sau dacă nu are opriri dese. În caz contrar, celelalte metode nu vor mai putea primi controlul unităţii centrale.

Există însă situaţii în care putem schimba această prioritate fără pericol, de exemplu când avem un fir de execuţie care nu face altceva decât să citească caractere de la utilizator şi să le memoreze într-o zonă temporară. În acest caz, firul de execuţie este în cea mai mare parte a timpului în starea Nu Rulează din cauză că aşteaptă terminarea unei operaţii de intrare/ieşire. În clipa în care utilizatorul tastează un caracter, firul va ieşi din starea de aşteptare şi va fi primul planificat la execuţie din cauza priorităţii sale ridicate. În acest fel utilizatorul are senzaţia că aplicaţia răspunde foarte repede la comenzile sale.

În alte situaţii, avem de executat o sarcină cu prioritate mică. În aceste cazuri, putem stabili, pentru firul de execuţie, care execută aceste sarcini o prioritate redusă.

##### Lucrare de laborator nr. 2

**1. Tema lucrării:** Gruparea firelor de execuție. Modificarea priorității lor.

**2. Scopul lucrării:**

* Studierea modificării priorității thread-urlilor în Java;
* Însuşirea metodelor de creare a unui grup de thread-uri;

**3. Etapele de realizare:**

1. Utilizarea clasei ThreadGroup pentru crearea grupelor de thread-uri;
2. Setarea priorităţilor threadurilor utilizînd funcția setPriority(….);
3. Utilizare funcției list();
4. Prezentarea lucrării.

**4. Exemplu:**

publicclass ThreadGroup1 {

publicstaticvoid main(String[] args) {

 ThreadGroup sys =

 Thread.currentThread().getThreadGroup();

 sys.list();

 sys.setMaxPriority(Thread.MAX\_PRIORITY - 1);

 Thread curr = Thread.currentThread();

 curr.setPriority(curr.getPriority() + 1);

 sys.list();

 ThreadGroup g1 = new ThreadGroup("g1");

 g1.setMaxPriority(Thread.MAX\_PRIORITY);

 Thread t = new Thread(g1, "A");

 t.setPriority(Thread.MAX\_PRIORITY);

 g1.list();

 g1.setMaxPriority(Thread.MAX\_PRIORITY - 2);

 g1.setMaxPriority(Thread.MAX\_PRIORITY);

 g1.list(); t = new Thread(g1, "B");

 t.setPriority(Thread.MAX\_PRIORITY);

 g1.list();

 g1.setMaxPriority(Thread.MIN\_PRIORITY + 2);

 t = new Thread(g1, "C");

 g1.list();

 t.setPriority(t.getPriority() -1);

 g1.list();

 ThreadGroup g2 = new ThreadGroup(g1, "g2");

 g2.list(); // (8)

 g2.setMaxPriority(Thread.MAX\_PRIORITY);

 g2.list(); // (9)

for (int i = 0; i < 5; i++)

new Thread(g2, Integer.toString(i));

 sys.list(); // (10)

 System.out.println("Starting all threads:");

 Thread[] all = new Thread[sys.activeCount()];

 sys.enumerate(all);

for(int i = 0; i < all.length; i++)

if(!all[i].isAlive())

 all[i].start();

}} ///:~

Rezultatul rulării programului:

(1) ThreadGroup[name=system,maxpri=10]

 Thread[main,5,system]

(2) ThreadGroup[name=system,maxpri=9]

 Thread[main,6,system]

(3) ThreadGroup[name=g1,maxpri=9]

 Thread[A,9,g1]

(4) ThreadGroup[name=g1,maxpri=8]

 Thread[A,9,g1]

(5) ThreadGroup[name=g1,maxpri=8]

 Thread[A,9,g1]

 Thread[B,8,g1]

(6) ThreadGroup[name=g1,maxpri=3]

 Thread[A,9,g1]

 Thread[B,8,g1]

 Thread[C,6,g1]

(7) ThreadGroup[name=g1,maxpri=3]

 Thread[A,9,g1]

 Thread[B,8,g1]

 Thread[C,3,g1]

(8) ThreadGroup[name=g2,maxpri=3]

(9) ThreadGroup[name=g2,maxpri=3]

(10)ThreadGroup[name=system,maxpri=9]

 Thread[main,6,system]

 ThreadGroup[name=g1,maxpri=3]

 Thread[A,9,g1]

 Thread[B,8,g1]

 Thread[C,3,g1]

 ThreadGroup[name=g2,maxpri=3]

 Thread[0,6,g2]

 Thread[1,6,g2]

 Thread[2,6,g2]

 Thread[3,6,g2]

 Thread[4,6,g2]

Starting all threads:

All threads started

**5. Sarcinile propuse spre realizare:**

##### Problema:

##### Creați ostructură, conform formulei date în tabelul 2, conform variantei. Enumerați toate firele de execuţie din grupa principală şi subgrupele ce le conţine. Afişați informaţia despre numele firului de execuţie, numele grupei din care face parte, și prioritatea sa. Prioritatea fiecărui Thread este indicată în paranteze () pentru fieacare fir de execuție (vezi tabelul 2).

Tabelul 2. Structura firelor de execuțe în dependență de variantă

|  |  |
| --- | --- |
| Nr. Variantei | **Formula structurii, compusă din fire şi grupuri de fire** |
| 1 | Main{G1{G3{Tha(3), Thb(3), Thc(3), Thd(3)}, ThA(3)},G2{Th1(4), Th2(5), Th3(5)},Th1(7), Th2(7)} |
| 2 | Main{G2{G1{Tha(1), Thb(3), Thc(8), Thd(3)}, ThA(1)},G3{Th1(4), Th2(3), Th3(5)},Th1(3), Th2(6)} |
| 3 | Main{GN{GH{Tha(4), Thb(3), Thc(6), Thd(3)}, ThA(3)},GM{Th1(2), Th2(3), Th3(3)},Th1(8), Th2(3)} |
| 4 | Main{GO{GZ{Tha(1), Thb(3), Thc(3), Thd(7)}, ThA(3)},GF{Th1(5), Th2(3), Th3(9)},Th1(3), Th2()} |
| 5 | Main{GE{GH{Tha(4), Thb(3), Thc(2), Thd(1)}, ThA(3)},GK{Th1(3), Th2(6), Th3(3)},Th1(3), Th2(7)} |
| 6 | Main{G1{G3{Thf(3), Thb(7), Thc(3), Thd(3)}, ThA(3)},G2{Th8(3), Th9(4), Th3(3)},Th1(3), Th2(3)} |
| 7 | Main{ThA(3)},G2{Th1(5), Th2(3), Th33(7)},Th11(3), Th22(3), G1{G3{Thaa(2), Thbb(3), Thcc(8), Thdd(3)}} |
| 8 | Main{G4{G3{Tha(2), Thb(8), Thc(3), Thd(3), G2{Th1(3), Th2(3), Th3(3), ThA(3)},Th1(8), Th2(3)}} } |
| 9 | Main{G6{ ThA(3)},Th1(4), Th2(3), G2{Th1(2), G3{Tha(2), Thb(3), Thc(4), Thd(3)}, Th2(3), Th3(3)}} |
| 10 | Main{G7{G3{Tha(6), Thb(3), Thc(6), Thd(3)}, ThA(7)},Th1(6), Th2(3), G2{Th1(7), Th2(3), Th3(3)}} |

1. **Întrebări de verificare:**
2. Ce permite gruparea firelor de execuție?
3. Ce metodă ne permite să determinăm grupul firului curent de execuție?
4. Ce constructor permite crearea și plasarea firului de execuție într-un unumit grup?
5. Ce prioritate are firul de execuție îm mod implicit?
6. Ce metode ne permit determinarea și setarea priorității unui fir de execuție?
7. Ce metodă ne permite să setăm prioritatea unui grup de fire de execuție?