**CAPITOLUL 3**

**3.1. Metodele clasei Thread**

## Constructorii clasei Thread

* Crearea unui nou obiect Thread

**public Thread();**

* Crearea unui nou obiect Thread cu indicarea obiectului pentru care va fi activată metoda run().

**public Thread(Runnable target);**

* Analogic constructorului precedent, dar se indică și numele obiectului Thread

**public Thread(Runnable target, String name);**

* Crearea unui nou obiect Thread cu indicărea numelui acestuia

**public Thread(String name);**

* Crearea unui nou obiect Thread, cu indicărea numelui grupului căruii îi aparține și obiectul pentru care va fi activată metoda run().

**public Thread(ThreadGroup group, Runnable target);**

* Analogic constructorului precedent, dar se indică și numele obiectului Thread

**public Thread(ThreadGroup group, Runnable target, String name);**

* Crearea unui nou obiect Thread cu indicarea numelui grupului căruii îi aparține și numele acestuia.

**public Thread(ThreadGroup group, String name);**

## 3.1.2. Metodele clasei Thread

## activeCount – determină numărul de thread-uri active din grupul căruia îi aparține

## public static int activeCount();

## currentThread – determină thread-ul curent activ

## public static Thread currentThread();

## destroy – destrugerea forțată a thread-ului

## public void destroy();

## enumerate – copiie în tabelul specificat în argument toate thread-urile active din grupa dată.

## public static int enumerate(Thread tarray[]);

## getName – returnează numele thread-ului.

## public final String getName();

## getPriority returnează prioritatea curentă a thread-ului

## public final int getPriority();

## getThreadGroup – returnează grupul căruii îi aparține thread-ul

## public final ThreadGroup getThreadGroup();

## interrupt – întrerupe realizarea thread-ului

## public void interrupt();

## Interrupted – determină dacă thread-ul este întrerupt

## public static boolean interrupted();

## isAlive – determină dacă thread-ul se realizează sau nu

## public final boolean isAlive();

## isDaemon – determină este thread-ul daemon

## public final boolean isDaemon();

## resume – restartarea thread-ului temporar oprit

## public final void resume();

## join – așteaptă realizarea complectă a thread-ului sau așteaptă realizarea thread-ului pe parcursul timpului indicat în milisecunde sau nanosecunde

## public final void join();

## public final void join(long millis);

## public final void join(long millis, int nanos);

## run – metoda este activată pentru a realiza thread-ul

## public void run();

## setDaemon – indică că thread-ul creat va fi daemon

## public final void setDaemon(boolean on);

## setName – atribue nume thread-ului

## public final void setName(String name);

## yield – oprește temporar thread-ul activ și permite execuția altui thread

## public static void yield();

## setPriority – setarea priorității

## public final void setPriority(int newPriority);

## sleep – oprește execuția (”adormirea”) thread-ului pe timpul indicat în milisecunde și nanosecunde

## public static void sleep(long millis);

## public static void sleep(long millis, int nanos);

## start – activarea thread-ului

## public void start();

## stop – dezactivarea finală a thread-urilor

## public final void stop();

## suspend – dezactivarea temporară a thread-ului

## public final void suspend();

##### Lucrarea de laborator nr. 3

**1. Tema lucrării:**

Sincronizarea firelor de execuție utilizînd metodele clasei Thread.

**2. Scopul lucrării:**

* Însuşirea modalităţilor de creare a thread-urilor în Java;
* Însuşirea metodelor de creare a unui grup de thread-uri;

**3. Etapele de realizare:**

1. Utilizarea clasei **Thread** pentru crearea unei clase noi;
2. Utilizarea interfeţei **Runnable** pentru crearea şi lansarea thread-urilor;
3. Utilizarea clasei **ThreadGroup** pentru crearea grupurilor de thread-uri;
4. Setarea priorităţilor thread – urilor utilizînd metoda **setPriority(….);**
5. Enumerarea thread-urilor şi a priorităţilorlor utilizînd metoda **enumerate();**
6. Utilizare metodei **list()**;
7. Prezentarea lucrării.
8. **Exemplu de realizare:**

public class Lab3PCD {

static int size = 100;

static int counter = 0;

static int[] b = new int[size];

static int[] a;

static Straight first = new Straight();

static Reverse second = new Reverse();

static ReverseInterval third = new ReverseInterval();

static StraightInterval fourth = new StraightInterval();

static intpairone = 0;

static intpairtwo = 0;

static class Straight extends Thread {

int result = 0;

@Override

public void run(){

System.out.println("Starting Thread 1");

for (inti = 0; i< counter; i+=4){

try{

Thread.sleep(100);

}

catch (InterruptedExceptione){

e.printStackTrace();

}

if ((i+4) >= counter){

}

else {

pairone = a[i] + a[i+1];

pairtwo = a[i+2] + a[i+3];

result = pairone + pairtwo;

System.out.println ("Current value for Thread 1 is: " + pairone + " + " + pairtwo + " = " + result);

}

}

while(fourth.isAlive()){

try{

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println("1: Name");

}

}

static class Reverse extends Thread {

int result = 0;

@Override

public void run(){

System.out.println("Starting Thread 2");

for (inti = counter-1; i>= 0; i-=4){

try{

Thread.sleep(100);

}

catch (InterruptedExceptione){

e.printStackTrace();

}

// if (a[i] <= 106 && a[i] >= 16) {

if ((i - 4) <= 0) {

}

else {

pairone = a[i] + a[i-1];

pairtwo = a[i-2] + a[i-3];

result = pairone + pairtwo;

System.out.println ("Current value for Thread 2 is: " + pairone + " + " + pairtwo + " = " + result);

}

}

System.out.println("2: Surname");

}

}

public static class StraightIntervalextends Thread {

@Override

public void run(){

System.out.println("Starting Thread 4 ");

for (inti = 200; i<= 300; i++){

System.out.print(i + " ");

}

System.out.println(" ");

while(second.isAlive()){

try{

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println("4: Group");

}

}

public static class ReverseInterval extends Thread {

@Override

public void run(){

System.out.println("Starting Thread 3 ");

for (inti = 1000; i<= 1100; i++){

System.out.print(i + " ");

}

System.out.println(" ");

while(first.isAlive()){

try{

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println("3: P C D");

}

}

public static void main(String[] args) {

System.out.println("Printing Array: ");

for (inti = 0; i< 100; i++) {

b[i] = (int) Math.round((Math.random() \* 100) + 15);

System.out.print (b[i] + " ");

if (i == 50){

System.out.println (b[i] + " ");

}

if (i == 99)

System.out.println();

if (b[i]%2==0){

counter++;

}

}

a = new int[counter+1];

int k = 0;

for (inti = 0; i< 100; i++){

if (b[i]%2==0){

a[k] = b[i];

k++;

}

}

first.start();

second.start();

third.start();

fourth.start();

try{

first.join();

second.join();

third.join();

fourth.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

**Rezultatul realizării:**

Printing Array:

44 25 18 115 103 83 53 67 95 56 107 17 82 16 85 59 55 30 16 114 40 20 112 39 101 102 59 20 27 60 68 21 86 106 87 81 26 107 68 52 16 92 113 79 62 80 109 67 56 100 63 63 105 114 47 39 34 94 59 26 43 17 40 23 81 104 94 55 56 57 20 18 65 88 81 43 77 103 101 48 40 82 88 91 29 19 28 53 19 51 77 20 38 56 54 33 57 20 70 103 108

Starting Thread 1

Starting Thread 2

Starting Thread 4

Starting Thread 3

1000 1001 1002 1003 200 1004 201 202 203 1005 204 1006 205 1007 206 1008 207 1009 208 1010 209 1011 210 211 1012 212 213 1013 214 215 1014 216 217 1015 218 1016 219 1017 220 1018 221 1019 222 1020 223 1021 224 1022 225 1023 226 1024 227 1025 228 1026 229 1027 230 1028 231 1029 232 1030 233 1031 234 1032 235 1033 236 1034 237 1035 238 1036 239 1037 240 1038 241 1039 242 1040 243 1041 244 1042 245 1043 246 1044 247 248 1045 249 1046 250 1047 251 252 253 1048 254 255 256 1049 1050 257 258 259 260 1051 1052 1053 261 262 263 264 265 266 267 1054 268 1055 269 1056 270 1057 271 1058 272 1059 273 1060 274 275 276 277 278 1061 279 1062 280 1063 281 1064 1065 282 1066 283 1067 284 1068 285 1069 1070 286 1071 287 1072 288 1073 1074 289 290 1075 291 1076 292 293 294 295 1077 296 1078 297 1079 298 1080 299 300 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099 1100

Current value for Thread 2 is: 178 + 74 = 252

Current value for Thread 1 is: 62 + 138 = 200

Current value for Thread 2 is: 94 + 48 = 142

Current value for Thread 1 is: 46 + 130 = 176

Current value for Thread 1 is: 60 + 214 = 274

Current value for Thread 2 is: 170 + 88 = 258

Current value for Thread 2 is: 106 + 154 = 182

Current value for Thread 1 is: 80 + 154 = 234

Current value for Thread 1 is: 132 + 66 = 252

Current value for Thread 2 is: 198 + 66 = 264

Current value for Thread 1 is: 128 + 214 = 342

Current value for Thread 2 is: 128 + 214 = 342

Current value for Thread 2 is: 136 + 154 = 290

Current value for Thread 1 is: 156 + 154 = 304

Current value for Thread 1 is: 68 + 94 = 212

Current value for Thread 2 is: 68 + 94 = 162

Current value for Thread 1 is: 192 + 128 = 188

Current value for Thread 2 is: 192 + 128 = 320

Current value for Thread 1 is: 136 + 122 = 258

Current value for Thread 2 is: 136 + 122 = 254

Current value for Thread 1 is: 116 + 58 = 174

Current value for Thread 2 is: 154 + 46 = 200

Current value for Thread 1 is: 110 + 90 = 200

2: Surname

4: Group

1: Name

3: P C D

5. **Probleme propuse spre realizare**:

##### Problema:

##### Scrieţi un program care creează patru fire de execuţie. Toate firele vor citi datele din diapazonul indicat în sarcină. Perechile de thread-uri 1-2 și 3-4 au sarcini commune. Primul fir Th1 va afişa: Condiţia 1 din tabelul 3, al doilea fir Th2 va afișa Condiția 2 din tabelul 3,al treilea fir Th3 va afișa Condiția 3 din tabelul 3, al patrulea fir Th4 va afișa Condiția 4 din tabelul 3.

##### După terminarea tuturor firelor de execuţie thread-ul Th2 va afişa Numele studentului care a efectuat lucrarea dată de laborator, Th4 va afișa grupa, Th1 va afișa Prenumele studentului, Th3 va afișa denumirea disciplinii. Literele textului vor apărea pe ecran cu un interval de 100 milisecunde.

Tabelul 3 Condiţiile pentru realizarea sarcinii problemei conform variantelor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Condiţia 1 | Condiţia 2 | Condiția3 | Condiția4 |
| 1 | Suma diferențilornumerelor pare douăcâtedouăîncepând de la ânceputuldeapazonului [100, 200] | Suma diferențilornumerelor pare douăcâtedouăîncepând de la  sfîrșituldeapazonului  [10, 110] | De parcurs de la începutintervalul  [100,500] | De parcurs de la sfîrșitintervalul  [300,700] |
| 2 | Suma produselornumerelor pare douăcâtedouăîncepând de la ânceputuldiapazonului  [1, 100] | Sumelepoziţiilornumerelor pare douăcâtedouăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [120, 690] | De parcurs de la sfîrșitintervalul  [1000, 1567] |
| 3 | Sumelenumerelor pare două cate douăîncepând de la ânceputuldiapazonului  [2, 102] | Sumelenumerelor pare două cate douăîncepândde la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [0, 798] | De parcurs de la sfîrșitintervalul  [1456, 2111] |
| 4 | Sumelepoziţiilornumerelor pare douăcâtedouăîncepând de la ânceputuldiapazonului  [15, 115] | Sumelepoziţiilornumerelor pare douăcâtedouăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [234, 987] | De parcurs de la sfîrșitintervalul  [123, 890] |
| 5 | Sumeleproduselornumerelor de pepoziţii pare douăcâtedouăîncepând de la ânceputuldiapazonului  [100, 200] | Sumeleproduselornumerelor de pepoziţii pare douăcâtedouăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [567, 1002] | De parcurs de la sfîrșitintervalul  [567, 1100] |
| 6 | Sumeleproduselornumerelor de pepoziţiiimparedouăcâtedouăîncepând de la ânceputuldiapazonului  [19, 119] | Sumeleproduselornumerelor de pepoziţiiimparedouăcâtedouăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [654, 1278] | De parcurs de la sfîrșitintervalul  [123, 908] |
| 7 | Sumeleproduselornumerelor paredouăcâtedouăîncepând de la ânceputuldiapazonului  [200, 300] | Sumeleproduselornumerelor paredouăcâtedouăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [234, 1000] | De parcurs de la sfîrșitintervalul  [456,1234] |
| 8 | Sumeleproduselornumerelorimparedouăcâtedouăîncepând de la ânceputuldiapazonului  [1, 100] | Sumeleproduselornumerelorimparedouăcâtedouăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [126, 987] | De parcurs de la sfîrșitintervalul  [213, 899] |
| 9 | Sumelenumerelor de peprimeletreipoziţiipare începând de la ânceputuldiapazonului  [1, 100] | Sumelenumerelor de pepoziţii pare câtetreiîncepândde la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  [222, 999] | De parcurs de la sfîrșitintervalul  3333, 3999[] |
| 10 | Smele produselornumerelor de pepoziţiiimparecâtedouăîncepând de la ânceputuldiapazonului  [1, 100] | Sumeleproduselornumerelor de pepoziţiiimparedouă cate douăîncepând de la sfîrșituldiapazonului  [6, 106] | De parcurs de la începutintervalul  11,548[] | De parcurs de la sfîrșitintervalul  [1234, 678] |

1. **Întrebări de verificare:**
   1. Definiți un fir de execuție?
   2. Ce motodă în limbajul Java permite crearea unui obiect nou Thread?
   3. Utilizarea cărei metode din limbajul Java permite setarea priorității?
   4. Numiți stările unui fir de execuție?
   5. Pentru ce se folosește metoda setNume(String)?
   6. Explicați noțiunea de procese concurente?