الملزمة الثالثة

Threads

- The previous processes assume that a process was an executing program with *a single thread* of control.
- Many modern O.S provide features for a process to contain *multiple* threads of control.
- Since process has multiple threads of control, the process can do more than one task at a time.
- Ex: a web browser might have one thread display images or text while another thread retrieves data from the network.

Benefits

1. *Responsiveness*: program continue running even if part of it is blocked.
2. *Resource sharing*: treads share the memory and the resources of the process to which they belong.
3. *Economy*: it is more time consuming to create and manage process than threads.
4. *Utilization of multiprocessor architectural*: threads running in parallel in multiprocessor system.

Note: in a single processor architecture, the CPU moves between each thread so quickly.

Thread Creation

```
Class woker1 extends Thread {
      Public void run( ) {
         System.out.println ("I am a worker thread");
       }
 }
```

```
public class first {
  public static void main ( String args [] ) {
      worker1 runner = new worker1( );
       runner.start( );
        system.out.println (" I am the main Thread");
    }
}
```

* الصنف first يقوم بخلق خيط thread جديدة بالاسم runner وهو عبارة عن كائن من الصنف worker1 الذي هو بدوره مشتق من الصنف الجاهز Thread . بعد ذلك يتم تشغيل الخيط باستخدام الدالة( )start والتي توم بما يلي: 1. تخصيص حيز خزني للخيط الجديد. 2. تستدعي الدالة( )run من الخيط.

* البرنامج first فيه خيطين هما 1. الدالة ( )main 2. الخيط runner.

* لغة Java هي اللغة **الوحيدة** التي تدعم استخدام الخيوط treads على مستوى اللغات. مما يعطيها قوة غير موجودة في اللغات التي اعتمدت عليها مثل C, C++.

*** توفر لغة Java الطرق(الدوال) التالية للتحكم بحالات الخيوط:**

1- **suspend**( ) : تعلّق الخيط الحالي.

2- **sleep**( ) : توقف تنفيذ الخيط لفترة من الزمن.

3- **resume**( ): تعيد تنفيذ الخيط الذي علّق تنفيذه.

4- **stop**( ) : توقف تنفيذ الخيط.

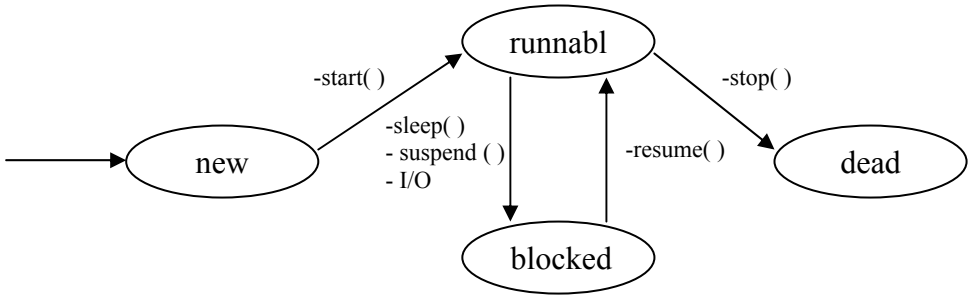
حالات الخيط Thread States

1- ***new***: يبقى الخيط بهذه الحالة لحين استدعاء الطريقة ( )start التي تنقله للحالة runnable.

2- ***runnable*** : عند استدعاء الطريقة ( )start من قبل الخيط فانه يتم تخصيص حيز خزني للخيط ويتم استدعاء الطريقة ( )run .

3- **Blocked**: يكون الخيط بهذه الحالة عندما ينفذ عمليةI/O او ينفذ الطريقتين ( )sleep او ( )suspend.

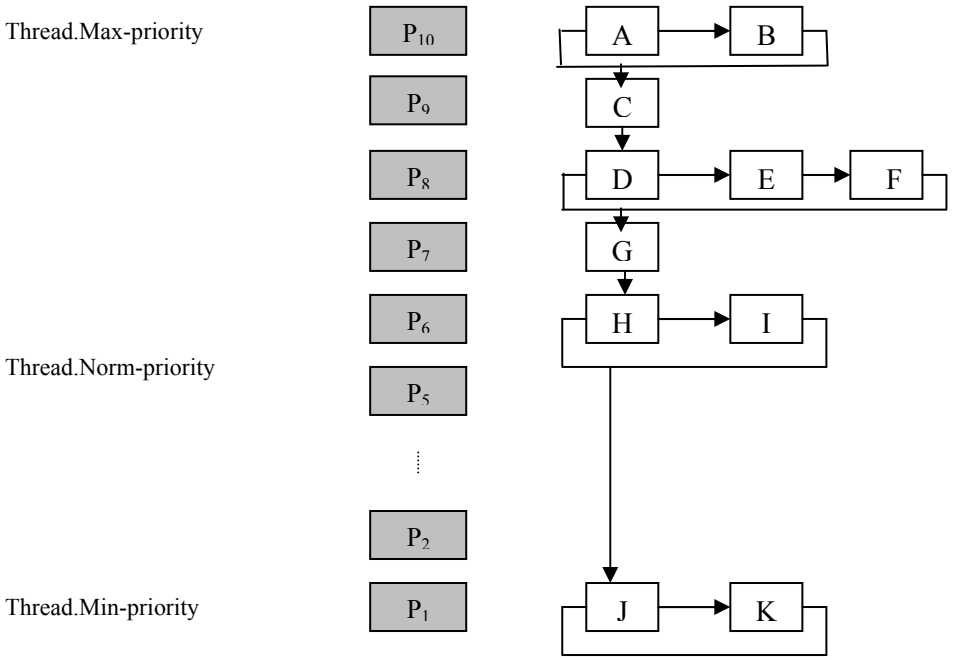
4- ***Dead***: عندما ينتهي الخيط من عمله أو يستدعي الطريقة( )stop .



[Java thread states]

Threads Priority and Scheduling أسبقيات وجدولة الخيوط

- تعتمد لغة جافا على خوارزمية Round Robin , حيث تبدأ بتنفيذ الخيوط ذات الأسبقية الأعلى ثم الأدنى, وهكذا.... إلى الخيوط الأقل أسبقية.



Q1) Write program to create and execute three threads, the first print the letter 'A' 100 times, the second print the letter 'B' 100 times, and the third print the numbers (1-100).

Answer:

```
import javax.swing.JOptionPane;
public class testthread {
  public static void main (String args[]){
    printchar pa=new printchar('A',100);
    printchar pb=new printchar('B',100);
    printnum  pn=new printnum(100);
    pa.start();
    pb.start();
    pn.start();
   JOptionPane.showMessageDialog(null,"end");
   System.exit(0);
  }
  }
```

```
class printchar extends Thread {
  private char ch;
  private int times;
  public printchar(char c,int t){
    ch=c;
    times=t;
  }

  public void run(){
   for(int i=0;i<times;i++)
     System.out.println(ch);
  }
  }
class printnum extends Thread {
   private int last;
   public printnum(int t){
    last=t;
  }
  public void run(){
   for(int i=0;i<last;i++)
     System.out.println("  "+i);
  }
  }
```

Q2) the following program shows multiple threads printing at different intervals.

```
import javax.swing.JOptionPane;
public class ThreadTester {
   public static void main( String args[] )
      {
       PrintThread thread1, thread2, thread3, thread4;
        thread1 = new PrintThread( "thread1" );
        thread2 = new PrintThread( "thread2" );
        thread3 = new PrintThread( "thread3" );
```

```
            thread4 = new PrintThread( "thread4" );
             System.out.println( "\nStarting threads" );
           //Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
            thread1.start();
            thread2.start();
            thread3.start();
            thread4.start();
             System.out.println( "end main\n" );
             JOptionPane.showMessageDialog(null,"end");
             System.exit(0);
           }
         }

class PrintThread extends Thread {
  private int sleepTime;
  private String nm;
  // PrintThread constructor assigns name to thread
   // by calling Thread constructor

public PrintThread( String name )
  {
    // sleep between 0 and 5 seconds
    sleepTime = (int) ( Math.random() * 5000 );
    nm=name;
  }
  // execute the thread

public void run()
  {
    // put thread to sleep for a random interval

    try {
          System.out.println( nm + " going to sleep "+sleepTime );
          Thread.sleep( sleepTime );
    }
    catch ( InterruptedException exception ) {
      System.out.println( exception.toString() );
    }
    System.out.println( nm + " done sleeping" );
  }
}
```