**Java Daemon Thread Examples**

* You can make any java thread as daemon thread. Daemon threads acts like service providers for other threads running in the same process.
* Daemon threads will be terminated by the JVM when there are none of the other threads running, it includs main thread of execution as well.
* To specify that a thread is a daemon thread, call the setDaemon method with the argument true.
* To determine if a thread is a daemon thread, use the accessor method isDaemon.

**Daemon Thread Sample Code**

|  |
| --- |
| package com.myjava.threads;  public class DaemonThread extends Thread{    public DaemonThread(){  setDaemon(true);  }  public void run(){  System.out.println("Is this thread Daemon? - "+isDaemon());  }  public static void main(String a[]){  DaemonThread dt = new DaemonThread();  // even you can set daemon constrain here also  // it is like dt.setDeamon(true)  dt.start();  }  } |

**Example Output**

Is this thread Daemon? - true

**Java Thread Join Examples**

* Imagine the following scenario. You are preparing for tomorrow's final examination and feel a little hungry. So, you give your younger brother ten bucks and ask him to buy a pizza for you. In this case, you are the main thread and your brother is a child thread. Once your order is given, both you and your brother are doing their job concurrently (i.e., studying and buying a pizza). Now, we have two cases to consider. First, your brother brings your pizza back and terminates while you are studying. In this case, you can stop studying and enjoy the pizza. Second, you finish your study early and sleep (i.e., your assigned job for today - study for tomorrow's final exam - is done) before the pizza is available. Of course, you cannot sleep; otherwise, you won't have a chance to eat the pizza. What you are going to do is to wait until your brother brings the pizza back.
* A thread can execute a thread join to wait until the other thread terminates
* A parent thread may join with many child threads created by the parent. Or, a parent only join with some of its child threads, and ignore other child threads. In this case, those child threads that are ignored by the parent will be terminated when the parent terminates.

**Thread Join Sample Code**

|  |
| --- |
| Code: |
| [?](http://www.java2novice.com/java_thread_examples/thread_join/)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | package com.myjava.threads;    import java.util.ArrayList;  import java.util.List;    public class MyThreadJoin {        public static List<String> names = new ArrayList<String>();        public static void main(String a[]){          List<SampleThread> list = new ArrayList<SampleThread>();          for(int i=0;i<5;i++){              SampleThread s = new SampleThread();              list.add(s);              s.start();          }          for(SampleThread st:list){              try{                  st.join();              } catch (Exception ex){}          }          System.out.println(names);      }  }    class SampleThread extends Thread{        public void run(){          for(int i=0; i<10; i++){              try{                  Thread.sleep(10);              } catch(Exception ex){}          }          MyThreadJoin.names.add(getName());      }  } | |

**Example Output**

[Thread-0, Thread-2, Thread-1, Thread-4, Thread-3]

**Java Thread Sleep**

* It makes current executing thread to sleep specified number of milli seconds. Below example shows sample code for thread sleep.

**Thread Sleep Sample Code**

|  |
| --- |
| Code: |
| package com.myjava.threads;  import java.util.ArrayList;  import java.util.List;  public class MyThreadSuspend {  public static void main(String a[]){  List<ExmpThread> list = new ArrayList<ExmpThread>();  for(int i=0;i<4;i++){  ExmpThread et = new ExmpThread(i+10);  list.add(et);  et.start();  }  for(ExmpThread et:list){  try {  et.join();  } catch (InterruptedException ex) { }  }  }  }  class ExmpThread extends Thread{  private int suspendCount;  public ExmpThread(int count){  this.suspendCount = count;  }  public void run(){  for(int i=0;i<50;i++){  if(i%suspendCount == 0){  try {  System.out.println("Thread Sleep: " + getName());  Thread.sleep(500);  } catch (InterruptedException ex) { }  }  }  }  } |

**Java Thread Yield Examples**

* When a thread executes a thread yield, the executing thread is suspended and the CPU is given to some other runnable thread. This thread will wait until the CPU becomes available again.
* Technically, in process scheduler's terminology, the executing thread is put back into the ready queue of the processor and waits for its next turn.

**Thread Yield Sample Code**

|  |
| --- |
| Code: |
| package com.myjava.threads;  import java.util.ArrayList;  import java.util.List;  public class MyThreadYield {  public static void main(String a[]){  List<ExmpThread> list = new ArrayList<ExmpThread>();  for(int i=0;i<3;i++){  ExmpThread et = new ExmpThread(i+5);  list.add(et);  et.start();  }  for(ExmpThread et:list){  try {  et.join();  } catch (InterruptedException ex) { }  }  }  }  class ExmpThread extends Thread{    private int stopCount;  public ExmpThread(int count){  this.stopCount = count;  }  public void run(){  for(int i=0;i<50;i++){  if(i%stopCount == 0){  System.out.println("Stoping thread: "+getName());  yield();  }  }  }  } |