**Difference between multithreading and multiprocessing**

|  |  |
| --- | --- |
| Multiprocessing | Multithreading |
| Multi processing assigned separate memeory and resources for each of the process  It is divided into categories  Creation of process is depend on the resources  Processing unit can be more than one but not necessary | Threads belonging to the same process share the same memory and resources at that of the process  It is not divided into categories  Creation of Thread is based on function  It contains more than one processing unit |

Ways of creating A Thread

-By extending Thread class

-by Implementing Runnable interface.

**process**:

-A process is a series of steps and decisions involved in the way task is completed.

**Multiprocess**:

-Executing several tasks simultaneosly where each task is seperate process is called multiprocess.

-->**thread**:

-thread is path for execution.

-thread is also a seperate call stack.

**Multithreading**:

-Executing multiple tasks at a time where each task is a seperate thread is called as multithreading.

**Preemptive scheduling**

-Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to ready state.

**Non-Preemptive scheduling**:

-Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state.

**thread priority**:

-Thread priority in Java is a number assigned to a thread that is used by Thread scheduler to decide which thread should be allowed to execute.

-Thread priorities are represented by a number from 1 to 10 that specifies the relative priority of one thread to another.

The thread with the highest priority is selected by the scheduler to be executed first.

**Set priority for thread:**

-The setPriority() of Thread class is used to set the priority of a thread. This method accepts an integer value as

an argument and sets that value as priority of a thread through which it is called.

Synatx:

-ThreadName.setPriority(n);

where, n is an integer value which ranges from 1 to 10.

-->**thread life cycle**:

-**New**: Whenever a new thread is created, it is always in the new state.

-**Active**:When a thread invokes the start() method, it moves from the new state to the active state.

The active state contains two states within it: one is runnable, and the other is running.

**Runnable**: A thread, that is ready to run is then moved to the runnable state. In the runnable state,

the thread may be running or may be ready to run at any given instant of time.

**Running**: When the thread gets the CPU, it moves from the runnable to the running state.

Generally, the most common change in the state of a thread is from runnable to

running and again back to runnable.

-**Blocked or Waiting**: Whenever a thread is inactive for a span of time (not permanently) then,

either the thread is in the blocked state or is in the waiting state.

-**Time waiting**: waiting for some other thread to perform a specific action for a specified period.

-**Terminated**: A thread reaches the termination state because of the following reasons:

When a thread has finished its job, then it exists or terminates normally.

**Abnormal termination**: It occurs when some unusual events such as an unhandled exception or segmentation fault.

**Which Approach Is Better To Create Thread**

* When we extend Thread class, we can’t extend any other class even we require
* When we implement Runnable, we can save a space for our class to extend any other class in future or now.
* When we extend Thread class, each of our thread creates unique object and associate with it.
* When we implements Runnable, it shares the same object to multiple threads

**Sleep() Method(ref scjp pg no721)**

* The sleep() method is a static method of class Thread.
* You use it in your code to "slow a thread down" by forcing it to go into a sleep mode before coming back to runnable (where it still has to beg to be the currently running thread).
* When a thread sleeps, it drifts off somewhere and doesn't return to runnable until it wakes up.

**Yeild() method(ref 726 scjp )**

* The yield() is supposed to do is make the currently running thread head back to runnable to allow other threads of the same priority to get their turn.
* So the intention is to use yield() to promote graceful turn-taking among equal-priority threads**.**
* A yield() won't ever cause a thread to go to the waiting/sleeping/ blocking state. At most, a yield() will cause a thread to go from running to runnable

**Advantages Of multi Threading**

**Responsiveness**

* Multithreading in an interactive application may allow a program to continue running even if a part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

**Resource Sharing**

* Threads will share same resources

**Economy**

* Allocating memory and resources for process creation is a costly job in terms of time and space.

**Scalability**

* The benefits of multi-programming greatly increase in case of multiprocessor architecture, where threads may be running parallel on multiple processors.

**Difference Between Run method and Start Method**

|  |  |
| --- | --- |
| Run | Start |
| While if run method is executed directly than no new Thread is created and code inside run() will execute on current Thread and no multi-threading will take place.  multiple invocation is possible in case of run method.  run method no new thread is created and task is made to be executed on same current thread so only one thread is there and hence no multithreading is introduced.  Run method executed by start method or get called directly  . | Start method of thread class is implemented as when it is called a new Thread is created and code inside run() method is executed in that new Thread.  start method can't be invoked more than once on same object otherwise it will throw java.lang.IllegalThreadStateException.  start method a new thread is created along with the current thread so atleast two threads are there and hence multithreading is introduced  start method internally called run method after creating a new thread. |