**Threads**

**Introduction:**

* Java provides built-in support for multithreaded programming.
* A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution.
* **Process:** A process consists of the memory space allocated by the operating system that can contain one or more threads. A thread cannot exist on its own; it must be a part of a process. A process remains running until all of the non-daemon threads are done executing.

1. What do you mean by thread?

A thread is a sequence of instructions that is executed one after another in its own call stack.

1. Which thread always runs in java program by default?

Main thread (when public static void main(String args[]) is called).

1. Why threads are called light-weight?

Threads are light weight because they utilize minimum resources of the system. This means they take less memory and less processor time.

1. What is the difference between single tasking and multi tasking?

|  |  |
| --- | --- |
| Single tasking (single threaded) | Multitasking (multi threaded) |
| Executing only one job at a time is called single tasking | Executing several jobs at a time is called multi tasking |
| Processor time is wasted | Here we can utilize processor time in an optimum way. |

1. Which method is executed by the thread by default?

Public void run() method.

1. In how many ways we can create a thread?

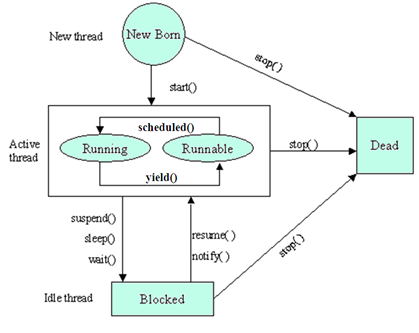
Java defines two ways in which this can be accomplished:

* By implementing the Runnable interface.
* By extending the Thread class, itself.

1. What the difference is between ‘extends thread’ and implements ‘runnable’? which one is advantageous?

|  |  |
| --- | --- |
| extends thread | implements ‘runnable’ |
| Both are functionally same, implements runnable is adavantageous | |
| When we write extends thread, there is no scope to extend another class, as multiple inheritance is not supported by java | If we write implements runnable, then still there is a scope to extend another class. |

1. Explain thread life cycle?

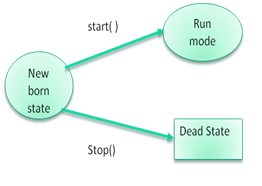


A thread can be in one of the following states

* New born state
* Ready to run state
* Running state
* Blocked state
* Dead state

**New Born state**

* The thread enters the new born state as soon as it is created. The thread is created using the new operator.
* From the new born state the thread can go to ready to run mode or dead state.
* If start( ) method is called then the thread goes to ready to run mode. If the stop( ) method is called then the thread goes to dead state.



**Ready to run mode (Runnable Mode)**

* If the thread is ready for execution but waiting for the CPU the thread is said to be in ready to run mode.
* All the events that are waiting for the processor are queued up in the ready to run mode and are served in FIFO manner or priority scheduling.
* From this state the thread can go to running state if the processor is available using the scheduled( ) method.
* From the running mode the thread can again join the queue of runnable threads.
* The process of allotting time for the threads is called time slicing.

**Running state**

* If the thread is in execution then it is said to be in running state.
* The thread can finish its work and end normally.
* The thread can also be forced to give up the control when one of the following conditions arise

|  |
| --- |
| A thread can be suspended by suspend( ) method. A suspended thread can be revived by using the resume() method. |
| A thread can be made to sleep for a particular time by using the sleep(milliseconds) method. The sleeping method re-enters runnable state when the time elapses. |
| A thread can be made to wait until a particular event occur using the wait() method, which can be run again using the notify( ) method. |

**Blocked state**

* A thread is said to be in blocked state if it prevented from entering into the runnable state and so the running state.
* The thread enters the blocked state when it is suspended, made to sleep or wait.
* A blocked thread can enter into runnable state at any time and can resume execution.

**Dead State**

* + The running thread ends its life when it has completed executing the run() method which is called natural dead.
  + The thread can also be killed at any stage by using the stop( ) method.

1. Explain how single threaded application works?

|  |
| --- |
| * A thread can be employed to execute one task at a time. * Suppose that there are three tasks to be executed, we can create a thread and passes 3 tasks one by one to the thread. * For this purpose, we can write all these tasks separately in separate methods: task1 (), task2 (), task3 (). * Then these methods should be called from run () method. One by one * It can never execute other methods unless they are called from run() |
| Example: By extending the Thread class |
| Public class MyThread extends Thread {  Public void run()  {  Task1();  Task2();  Task3();  }  Void task1(){  //statements  }  Void task2(){  //statements  }  Void task3(){  //statements  }  Public static void main(String Args[])  {  MyThread obj=new MyThread();  Thread t=new Thread(obj)  t.start();  }  } |

1. Explain how multi threaded application works?

* Using more than one thread is called multi-threading.
* Scenario

|  |
| --- |
| * When we go to the movie theater, generally a person is at the door- checking and cutting the tickets. * When we enter the hall, there is another person who shows the seats to us. * Suppose there is only one person (1 thread) doing these two tasks. * He has to first cut the ticket and then come along with to us to show the seat. * Then he goes back to door to cut the second ticket and then again enter the hall to show the seat for the second ticket. * Like this, if he does the things one by one it takes a lot of time in the mean time show will be running and some people will still outside the theatre. * To overcome with the above problem, They employee two persons( 2 threads), the first person will cut the ticket and second person will show the seat, * Like this both persons can act simultaneously and there will no wastage of time? |
| Example: By implementing the Runnable interface. |
| Public class MyThread implements runnable  {  String str;  MyThread(String str)  {  this.str=str;  }  Public void run(){  for(int i=1;i<=10;i++)  {  Sysout(str+ “:”+i)  }  }  }  //another class  Class Theatre {  Public static void main(String Args[]){  MyThread obj1=new MyThread(“cut the ticket”);  MyThread obj2=new MyThread(“show the seat”);  Thread t1=new Thread(obj1);  Thread t2=new Thread(obj2);  t1.start();  t2.start();  }  } |
| Output: cut the ticket: 1  Show the seat:1  cut the ticket: 2  Show the seat:2  …….  cut the ticket: 10  Show the seat:10 |

1. What are the advantages of using thread?

* It increases the speed of execution.
* It allows to run more tasks simultaneously
* It reduces the complexity of the program.
* It maximizes the CPU utilization.

1. Explain Thread priorities?

* Every Java thread has a priority that helps the operating system determine the order in which threads are scheduled.
* Java priorities are in the range between MIN\_PRIORITY (a constant of 1) and MAX\_PRIORITY (a constant of 10). By default, every thread is given priority NORM\_PRIORITY (a constant of 5).
* Threads with higher priority are more important to a program and should be allocated processor time before lower-priority threads. However, thread priorities cannot guarantee the order in which threads execute and very much platform dependentant.

1. What is the default priority of a thread?

Default priority will be: 5

1. Explain thread Synchronization?

* When a thread is already acting on an object, preventing any other thread from acting on the same object is called Synchronization or thread-safe.
* The object on which the threads are synchronized is called ‘synchronized object’.
* Synchronization is recommended when multiple threads are used on the same object.

1. What is the difference between synchronized block and synchronized keyword?

* Synchronized block is useful to synchronize the block of statements.
* Synchronized keyword is useful to synchronize an entire method.

1. What is the difference when the synchronized keyword is applied to a static method or to a non static method?

|  |  |
| --- | --- |
| static method | non static method |
| When a synch non static method is called, a lock is obtained on the object. | When a synch static method is called, a lock is obtained on the class and not on the object. |
|  |  |

1. What is the difference between yield() and sleep()?

|  |  |
| --- | --- |
| yield() | sleep() |
| yield() allows the current thread to release its lock from the object. | sleep() allows the thread to go to sleep state for x milliseconds. When a thread goes into sleep state it doesn’t release the lock. |

1. What is difference between notify() and notfiyAll()?

|  |  |
| --- | --- |
| notify() | notfiyAll() |
| This method releases an object and sends notification to a waiting thread that the object is available. | This method is useful to send notification to all waiting threads at once that the object is available. |
| ***Wait():*** this method makes a thread wait for an object till it receives a notification from a notify() and notifyAll() methods | |

1. What is the difference between sleep() and wait() methods?

|  |  |
| --- | --- |
| sleep() | wait() |
| Both are used to suspend a thread execution for a specified time | |
| When sleep() method is executed inside the synchronized block, the object is still under lock | When wait() method is executed inside the synchronized block, it breaks the synchronized block, so that object lock is removed and it is available |
| Sleep() is used for making a thread to wait for some time | Wait() method makes a thread wait for an object till it receives a notification from a notify() and notifyAll() methods |

1. Can the variables or classes be synchronized?

No. Only methods can be synchronized.

1. How many locks does an object have?

Each object has only one lock.

1. What happens if a start method is not invoked and the run method is directly invoked?

If the start() method is not invoked and the run() method is directly called on the Thread instance, the code inside the run() method will not run in a separate new thread but it will start running in the existing thread.

1. Can a class have both synchronized and non-synchronized methods?

Yes a class can have both synchronized and non-synchronized methods.

1. What is thread deadlock?

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other.

1. What is daemon thread?

* A daemon thread is a thread that executes contineouslely.
* Daemon threads are service providers for other threads or objects.
* It generally provides a background processing