**Thread**

* Thread allows the Program to operate more efficiently by doing multiple things at the same time.
* Multithreading is used to achieve multitasking more efficiently
* Thread uses shared memory area.
* Faster context switching
* Another advantage is quick response, if we use multiple threads in a process and if a thread gets stuck due to lack of resources or an exception, the other threads can continue to execution, allowing to continue to be responsive.

For example, in MS Word. one thread automatically saves the document while another thread is taking user input.

1. **Threading using Thread Class**

public class MyThread1 extends Thread {

@Override

public void run() {

int i=0;

while (i < 10 ) {

System.out.println("My threadOne is Runnning. ");

System.out.println("Thread 1:=> I am happy ! ");

i++;

}

}

}

public class MyThread2 extends Thread {

@Override

public void run() {

int i=0;

while (i < 10 ) {

System.out.println("My threadTwo is Runnning. ");

System.out.println("Thread2 :=> I am happy ! ");

i++;

}

}

}

OUTPUT

My threadOne is Runnning.

My threadTwo is Runnning.

Thread2 :=> I am happy !

My threadTwo is Runnning.

Thread2 :=> I am happy !

Thread 1:=> I am happy !

My threadOne is Runnning.

My threadTwo is Runnning.

Thread2 :=> I am happy !

1. **Threading using runnable Interface.**

* We cannot use directly start method, when we use class of runnable interface
* First, we create object of Thread Class.
* Then, put the Object of interface implemented class, to the Object of Thread class.
* Only then we can pass run method to Object of thread class

public class MyThreadRun implements Runnable{

@Override

public void run() {

int i =0;

while (i<10) {

System.out.println("Runnable Thread One. ");

i++;

}

}

}

public class MyThreadRunTwo implements Runnable{

@Override

public void run() {

int i =0;

while (i<10) {

System.out.println("Runnable Thread Two. ");

i++;

}

}

}

OUTPUT

Runnable Thread Two.

Runnable Thread Two.

Runnable Thread One.

Runnable Thread One.

Runnable Thread Two.

Runnable Thread Two.

1. **THREAD LIFE Cycle**
2. **NEW: whenever we create Object of Thread Class that State is known as NEW state of Thread.**
3. **RUNNABLE: After invocation of start() and before selected to be run by Schedular.**
4. **RUNNING: After Thread schedular has selected it.**
5. **NON-RUNNABLE: Thread alive, not eligible to run. (Thread is waiting to get response due to network issue, so thread get blocked and it’s status is not runnable. Thread Scheduler will send another Thread.)**
6. **Terminated: run() method has exited.**

**We can achieve Thread State by using getState() method**

**Refer below program**

**public static void main(String[] args) {**

**MyThreadRun mt1 = new MyThreadRun();**

**Thread t1 =new Thread(mt1);**

**System.out.println("T1 Thread State: "+t1.getState());**

**MyThreadRunTwo mt2 = new MyThreadRunTwo();**

**Thread t2 = new Thread(mt2);**

**System.out.println("T2 Thread State: "+t2.getState());**

**t1.start();**

**System.out.println("T1 Thread State: "+t1.getState());**

**t2.start();**

**System.out.println("T2 Thread State: "+t2.getState());**

**System.out.println("T1 Thread State: "+t1.getState());**

**System.out.println("T2 Thread State: "+t2.getState());**

**System.out.println("T1 Thread State: "+t1.getState());**

**System.out.println("T2 Thread State: "+t2.getState());**

**}**

**}**

**OUTPUT**

**T1 Thread State: NEW**

**T2 Thread State: NEW**

**T1 Thread State: RUNNABLE**

**Thread One.**

**Thread One.**

**T2 Thread State: RUNNABLE**

**Thread One.**

**Thread One.**

**Thread One.**

**T1 Thread State: WAITING**

**T2 Thread State: RUNNABLE**

**T1 Thread State: TERMINATED**

**T2 Thread State: RUNNABLE**

**Thread Two.**

**Thread Two.**

**Thread Two.**

**Thread Two.**

**Thread Two.**

1. **THREAD CONSTRUCTOR**
2. **Thread()**
3. **Thread(Runnable target)**
4. **Thread(Runnable target, String name)**
5. **Thread(String name)**
6. **THREAD PRIORITIES**

Thread tr1 =new Thread ();

tr1.setPriority(MAX\_PRIORITY);

tr1.Start();

Thread tr2 = new Thread ();

Tr2.setPriority(MIN\_PRIORITY);

Tr2.Start();

OUTPUT

Tr1 running

Tr1 running

Tr1 running

Tr2 running

Tr2 running

Tr2 running

1. THREAD METHOD

Join()

Thread Schedular waits till current thread stops working