Java Thread Join Example

Java Thread join method can be used to pause the current thread execution until unless the specified thread is dead. There are three overloaded join functions.

public final void join(): This method puts the current thread on wait until the thread on which it's called is dead. If the thread is interrupted, it throws InterruptedException.

public final synchronized void join(long millis): This method is used to wait for the thread on which it's called to be dead or wait for specified milliseconds. Since thread execution depends on OS implementation, it doesn't guarantee that the current thread will wait only for given time.

public final synchronized void join(long millis, int nanos): This method is used to wait for thread to die for given milliseconds plus nanoseconds.

Here is a simple example showing usage of **Thread join** methods. The goal of the program is to make sure main is the last thread to finish and third thread starts only when first one is dead.

```
public class ThreadJoinExample {

    public static void main(String[] args) {
        Thread t1 = new Thread(new MyRunnable(), "t1");
        Thread t2 = new Thread(new MyRunnable(), "t2");
        Thread t3 = new Thread(new MyRunnable(), "t3");

        t1.start();

        //start second thread after waiting for 2 seconds or if it's dead
        try {
            t1.join(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t2.start();

        //start third thread only when first thread is dead
        try {
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t3.start();

        //let all threads finish execution before finishing main thread
```

```
        try {
            t1.join();
            t2.join();
            t3.join();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        System.out.println("All threads are dead, exiting main thread");
    }

}

class MyRunnable implements Runnable{

    @Override
    public void run() {
        System.out.println("Thread started:::"+Thread.currentThread().getName());
        try {
            Thread.sleep(4000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Thread ended:::"+Thread.currentThread().getName());
    }

}
```

Output of the above program is:

```
Thread started:::t1
Thread ended:::t1
Thread started:::t2
Thread started:::t3
Thread ended:::t2
Thread ended:::t3
All threads are dead, exiting main thread
```