Working of Thread.join() method

Java provides one method join in thred which holds the execution of current thread until joined thread completes its execution.

**Thread X**

**At time t1 Thread Y joins current thread ie Thread X and at that time current thread Thread X holds its execution and Thread Y starts its execution.**

**At time t2 Thread Y completes its execution and and Thread X resumes its execution back.**

**Time t2**

**Time t1**

**Execution of X**

**Execution of X**

**Execution of Y**

**Thread Y completes its execution**

**Join Thread Y**

**Thread X holds its execution**

Lets understand it better with one below given real life example.

We Indian like to eat pranthas(a type of an Indian stuffed chapati) in breakfast very much and I usually have prantha on Sundays.

One day it was Sunday and I were having breakfast and I were almost done and about to leave the dining table. i were having my last bite of paratha and suddenly one of my friend came and joined me. Though I had completed my breakfast but as per table antiquates we should not leave dining table until the people sitting with complete their food we should not leave the table.

Here we can relate this scenario with thread join concept like me who were sitting on table was having breakfast is current thread and the guy who comes suddenly is a thread join and I cannot leave table until that guy completes his breakfast.

**How join method works internally.**

Consider a scenario below and we will try to understand join() method by going through example.

class ThreadJoinDemo extends Thread{

static ThreadJoinDemo thread1;

public void run(){

try{

synchronized(thread1){

System.out.println(Thread.currentThread().getName()+" acquired a lock on thread1");

Thread.sleep(5000);

System.out.println(Thread.currentThread().getName()+" completed");

}

}

catch (InterruptedException e){ }

}

public static void main(String[] ar) throws Exception{

thread1 = new ThreadJoinDemo();

thread1.setName("thread1");

thread1.start();

synchronized(thread1){

System.out.println(Thread.currentThread().getName()+" acquired a lock on thread1");

Thread.sleep(1000);

thread1.join();

System.out.println(Thread.currentThread().getName()+" completed");

}

}

}

**Output:**  
main acquired a lock on thread1  
thread1 acquired a lock on thread1 //after 1 second this line is printed  
thread1 completed //after 5 second this line is printed  
main completed

In above example, we created 2 Threads,

1. "main" thread
2. "thread1" thread

Based on output, the flow went as shown below,

