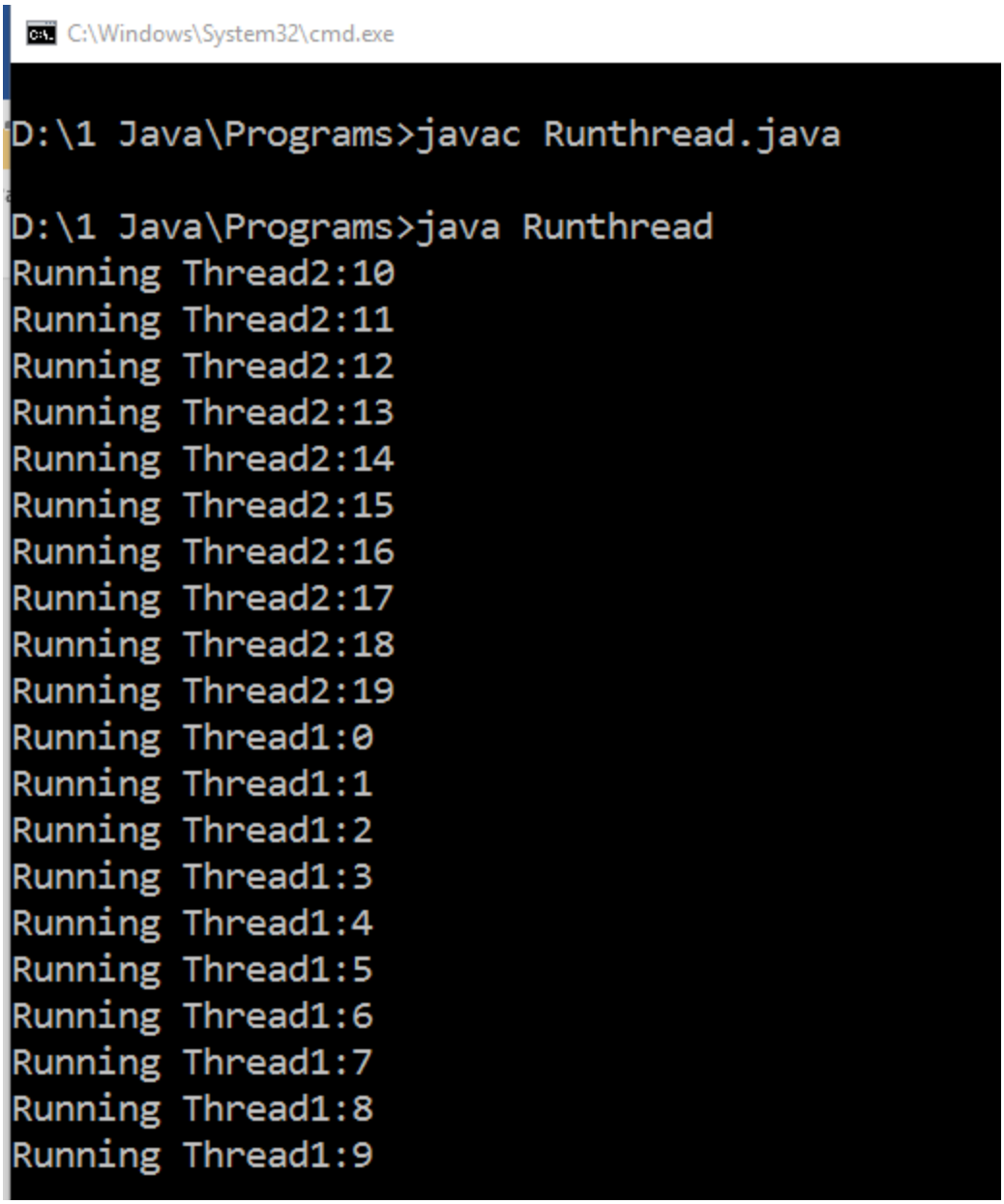
Important Methods of Thread Class:

sleep() method:
sleep(): This method causes the currently executing thread to sleep for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.

Syntax:
public static void sleep(long milliseconds) throws InterruptedException

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=0;i<10;i++) {
                        try{
        Thread.sleep(1000); // sleeps thread for 1 sec
      } catch(InterruptedException e){
        System.out.println(e);
      }
                        System.out.println("Running Thread1:"+i); }
        } }
class Mythread2 extends Thread {
        public void run() {
                for(int i=10;i<20;i++)
                {
        System.out.println("Running Thread2:"+i); }
        } }
class Runthread {
        public static void main(String arg[]){
    Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
      t1.start();
    Mythread2 r2=new Mythread2();
         r2.start();
                }
}
```



The output shows that thread 2 is executed first, after that thread 1 is executed.

Did we achieve multithreading through this program—YES/NO

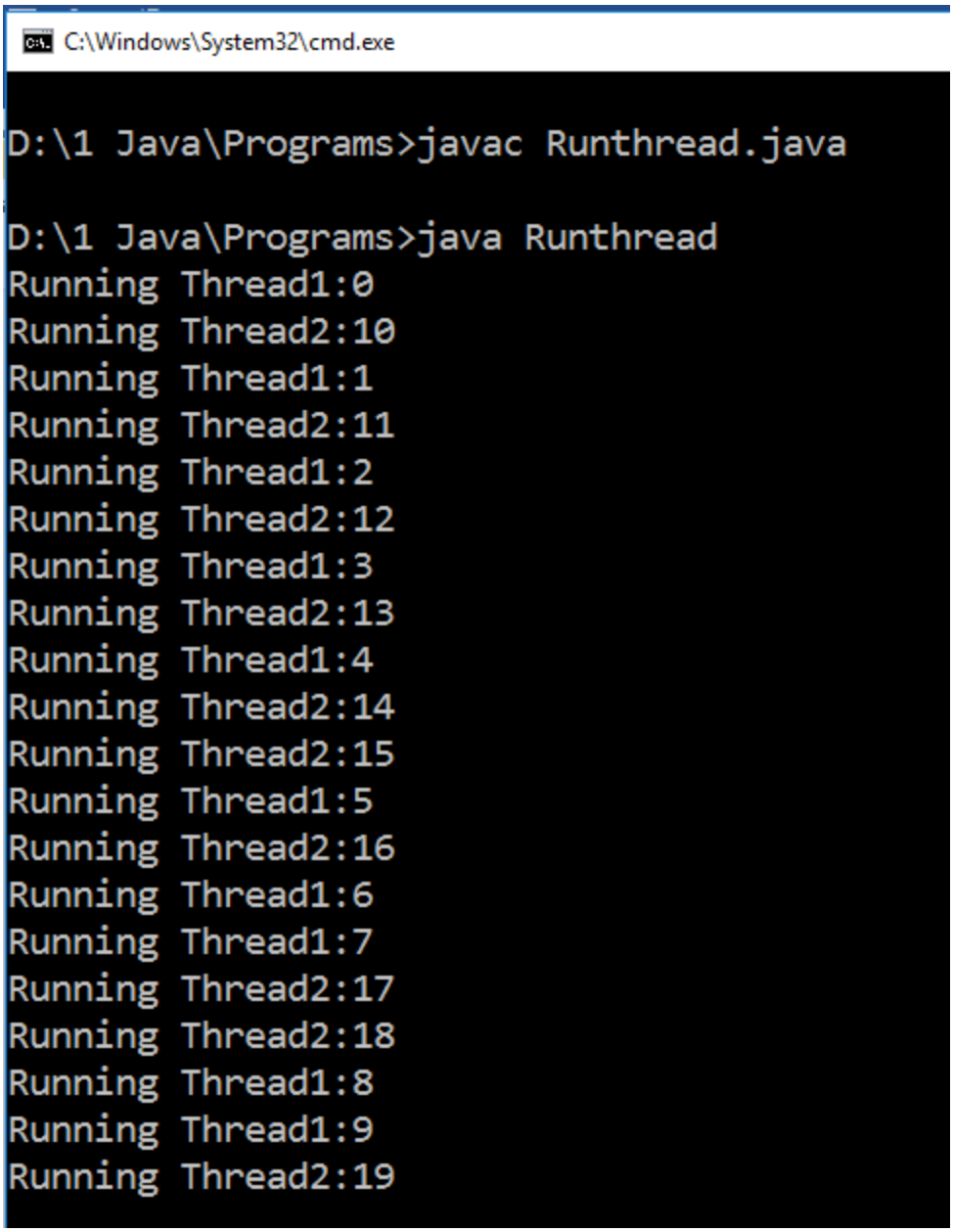
answer: yes

As soon as the execution of thread 1 is started, the compiler found that it has a sleep() method that tells t1 to stop its task for 1 second, as JVM has stopped the task of t1 for 1 second, The other thread 2 gets a chance to execute and t2 finishes its job within 1 second.

Try this program using increase the load of both the threads and decrease the sleep time of thread1, you will get mixed output means multithreading.

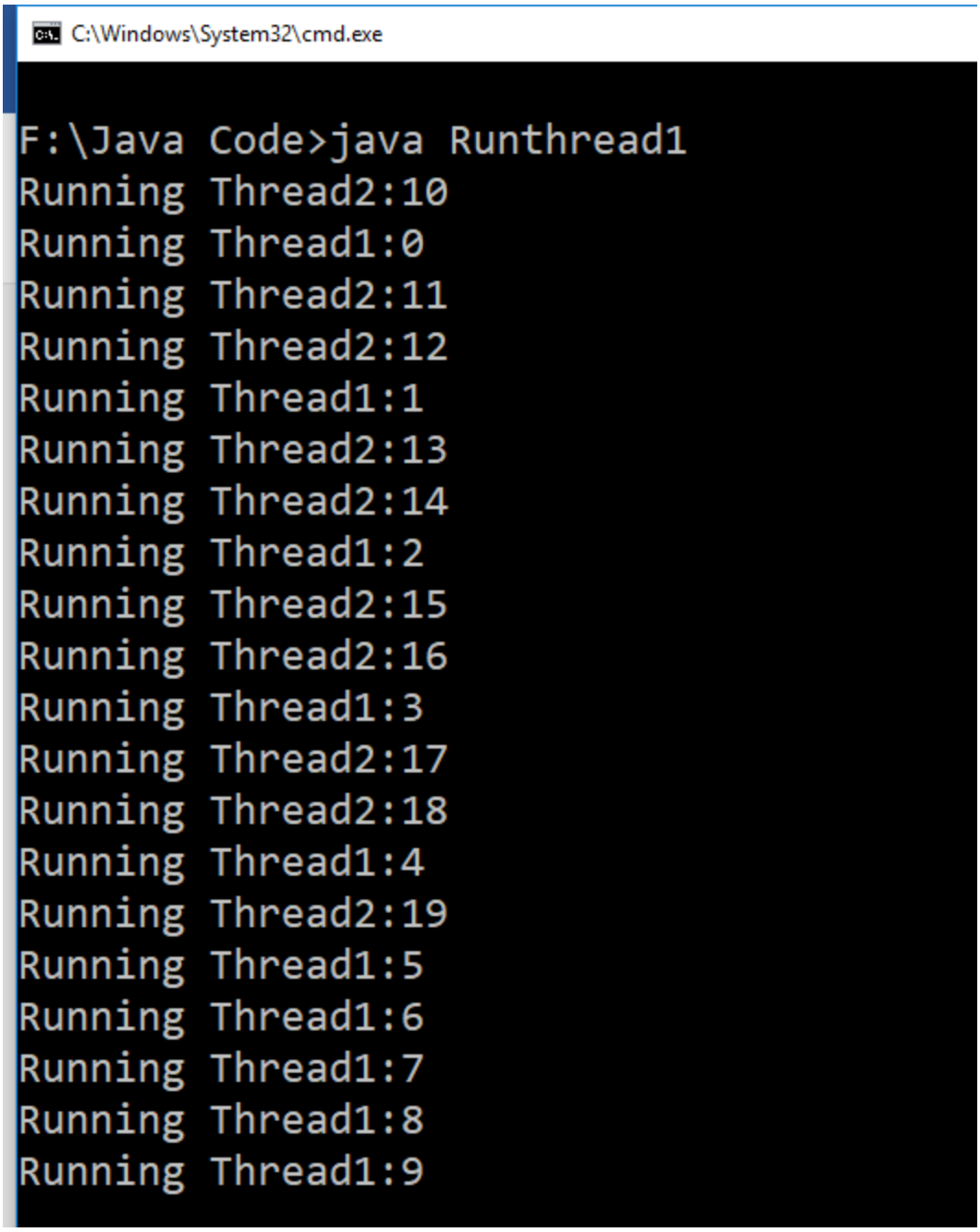
The following program shows that both threads have a sleep method to stop their task for a few seconds.

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=0;i<10;i++) {
                        try{
        Thread.sleep(1000); // sleeps thread for 1 sec
      } catch(InterruptedException e){
        System.out.println(e);
      }
                        System.out.println("Running Thread1:"+i); }
        } }
class Mythread2 extends Thread {
        public void run() {
                for(int i=10;i<20;i++)
                {
        try{
        Thread.sleep(1000); // sleeps thread for 1 sec
      } catch(InterruptedException e){
        System.out.println(e);
      }
        System.out.println("Running Thread2:"+i); }
        } }
class Runthread {
        public static void main(String arg[]){
    Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
      t1.start();
    Mythread2 r2=new Mythread2();
         r2.start();
                }
}
```


C:\Windows\System32\cmd.exe
D:\1 Java\Programs>javac Runthread.java
D:\1 Java\Programs>java Runthread
Running Thread1:0
Running Thread2:10
Running Thread1:1
Running Thread2:11
Running Thread1:2
Running Thread2:12
Running Thread1:3
Running Thread2:13
Running Thread1:4
Running Thread2:14
Running Thread2:15
Running Thread1:5
Running Thread2:16
Running Thread1:6
Running Thread1:7
Running Thread2:17
Running Thread2:18
Running Thread1:8
Running Thread1:9
Running Thread2:19

The following program shows that thread 1 is stopping its work for 2 seconds and thread 1 is stopping its work for 1 second, which means that thread 2 is getting more execution time than thread 1.

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=0;i<10;i++) {
                        try{
        Thread.sleep(2000); // sleeps thread for 2 sec
      } catch(InterruptedException e){
        System.out.println(e);
      }
                        System.out.println("Running Thread1:"+i); }
        } }
class Mythread2 extends Thread {
        public void run() {
                for(int i=10;i<20;i++)
                {
        try{
        Thread.sleep(1000); // sleeps thread for 1 sec
      } catch(InterruptedException e){
        System.out.println(e);
      }
        System.out.println("Running Thread2:"+i); }
        } }
class Runthread1 {
        public static void main(String arg[]){
    Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
      t1.start();
    Mythread2 r2=new Mythread2();
         r2.start();
                }
}
```


C:\Windows\System32\cmd.exe
F:\Java Code>java Runthread1
Running Thread2:10
Running Thread1:0
Running Thread2:11
Running Thread2:12
Running Thread1:1
Running Thread2:13
Running Thread2:14
Running Thread1:2
Running Thread2:15
Running Thread2:16
Running Thread1:3
Running Thread2:17
Running Thread2:18
Running Thread1:4
Running Thread2:19
Running Thread1:5
Running Thread1:6
Running Thread1:7
Running Thread1:8
Running Thread1:9

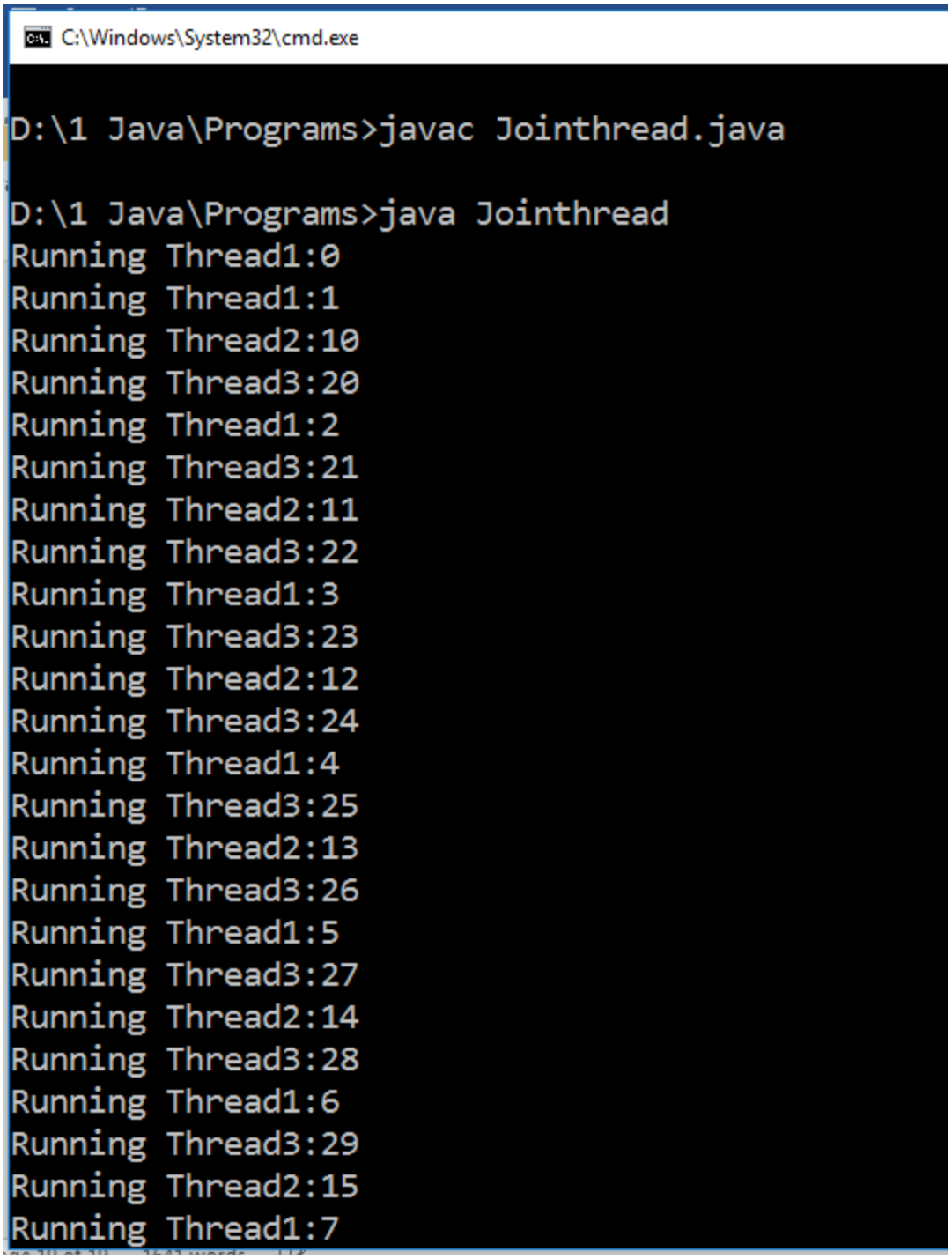
This program shows the multithreading among three threads.

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=0;i<10;i++) {
System.out.println("Running Thread1:"+i); }
        } }

class Mythread2 extends Thread {
        public void run() {
                for(int i=10;i<20;i++)

                {
        System.out.println("Running Thread2:"+i); }
        } }

        class Mythread3 extends Thread {
        public void run() {
                for(int i=20;i<30;i++)
                {
        System.out.println("Running Thread3:"+i); }
        } }
class Jointhread {
        public static void main(String arg[]){
    Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
    t1.start();
    Mythread2 r2=new Mythread2();
         r2.start();
         Mythread3 r3=new Mythread3();
         r3.start();
                }
}
```


C:\Windows\System32\cmd.exe
D:\1 Java\Programs>javac Jointhread.java
D:\1 Java\Programs>java Jointhread
Running Thread1:0
Running Thread1:1
Running Thread2:10
Running Thread3:20
Running Thread1:2
Running Thread3:21
Running Thread2:11
Running Thread3:22
Running Thread1:3
Running Thread3:23
Running Thread2:12
Running Thread3:24
Running Thread1:4
Running Thread3:25
Running Thread2:13
Running Thread3:26
Running Thread1:5
Running Thread3:27
Running Thread2:14
Running Thread3:28
Running Thread1:6
Running Thread3:29
Running Thread2:15
Running Thread1:7

.
.
.

join() Method

When the join() method is invoked, the current thread stops its execution and the thread goes into the wait state. The current thread remains in the wait state until the thread on which the join() method is invoked has achieved its dead state. If an interruption of the thread occurs, then it throws the InterruptedException.

Syntax:

- public void join()throws InterruptedException
- public void join(long milliseconds) throws InterruptedException

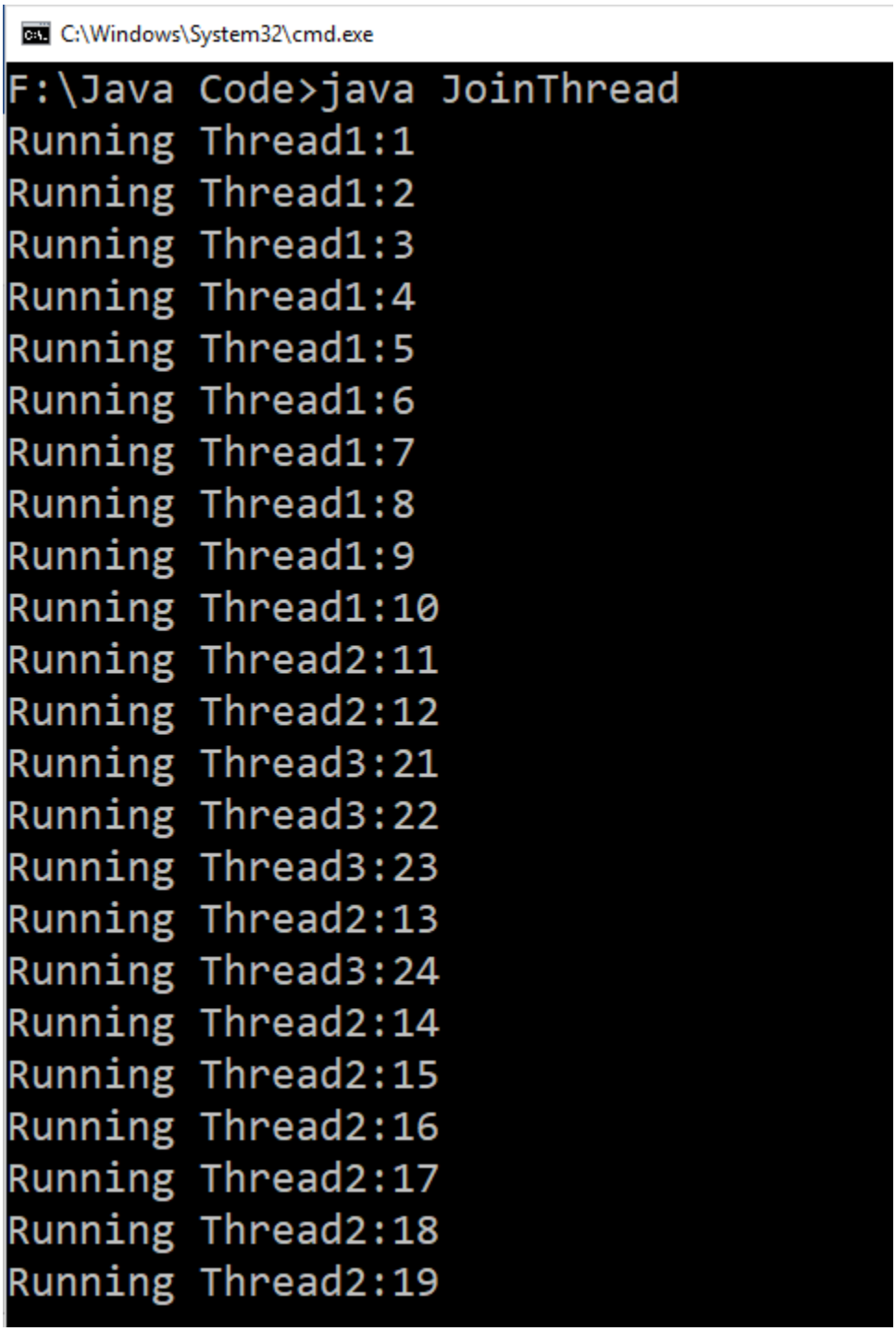
//This program shows the working of join() method.

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=1;i<11;i++) {
                        System.out.println("Running Thread1:"+i); }
        } }
class Mythread2 extends Thread {
        public void run() {
                for(int i=11;i<21;i++)
                {
        System.out.println("Running Thread2:"+i); }
        } }

        class Mythread3 extends Thread {
        public void run() {
                for(int i=21;i<31;i++)
                {
        System.out.println("Running Thread3:"+i); }
        } }

class JoinThread {
        public static void main(String arg[]){
        Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
      t1.start();
                try{
                        t1.join();  //it force to complete the task of t1 thread.
                        }
                catch(Exception e)
                {
                        System.out.println(e);
                }
    Mythread2 t2=new Mythread2();
         t2.start();
         Mythread3 t3=new Mythread3();
         t3.start();
```

```
        }
}
```

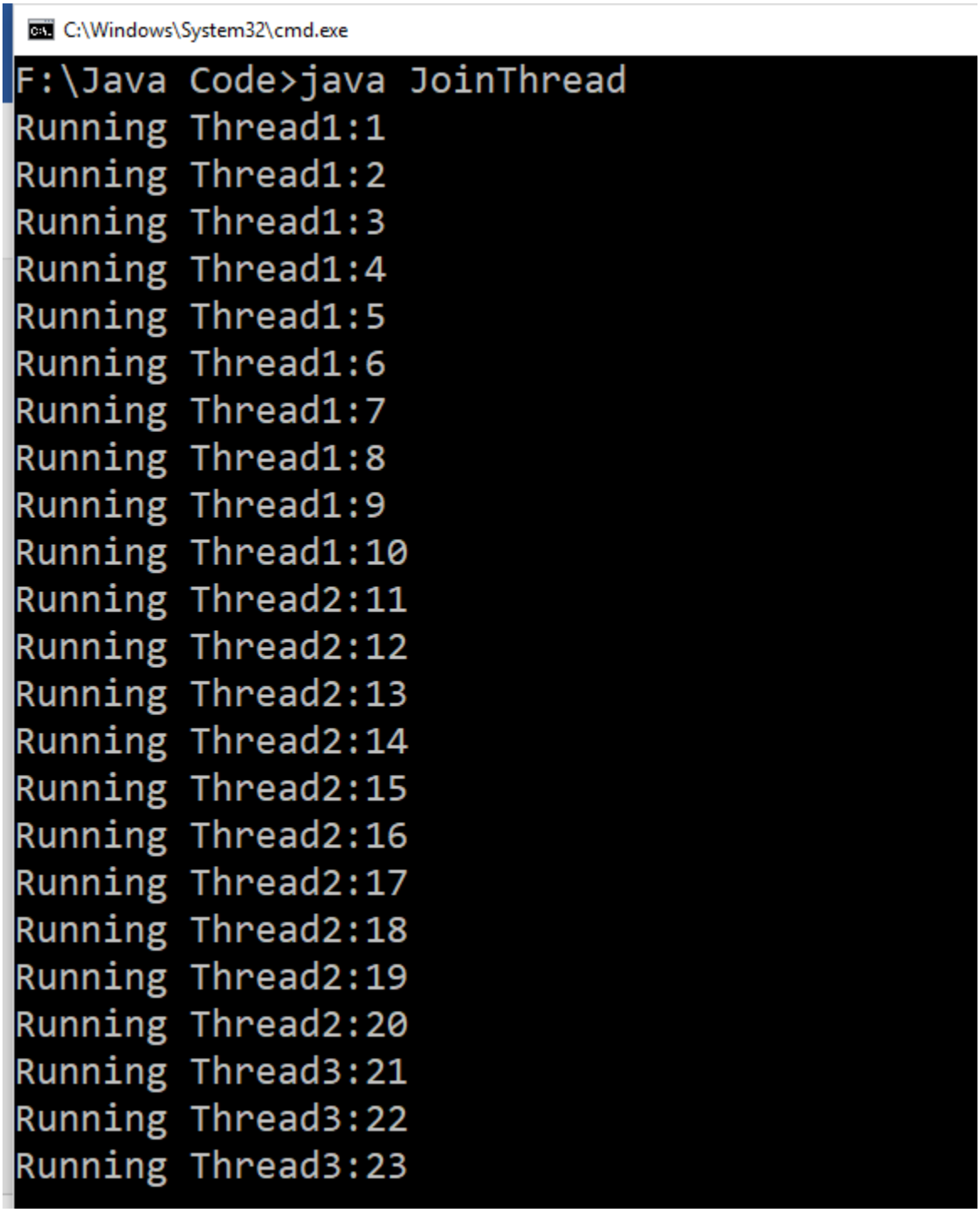


The output shows that as soon as thread 1 gets the processor to perform its task, it forces the JVM to complete its work (because of the join() method), and after the full execution of the t1 thread, it allows the processor to perform the task of the other thread.

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=1;i<11;i++) {
                        System.out.println("Running Thread1:"+i); }
        } }
class Mythread2 extends Thread {
        public void run() {
                for(int i=11;i<21;i++)
                {
        System.out.println("Running Thread2:"+i); }
        } }
        class Mythread3 extends Thread {
        public void run() {
                for(int i=21;i<31;i++)
                {
        System.out.println("Running Thread3:"+i); }
        } }

class JoinThread {
        public static void main(String arg[]){
        Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
      t1.start();
                try{
                        t1.join();  //it force to complete the task of t1 thread.
                        }
                catch(Exception e)
                {
                        System.out.println(e);
                }

    Mythread2 t2=new Mythread2();
         t2.start();
         try{
                        t2.join();  //it force to complete the task of t2 thread.
                        }
                catch(Exception e)
                {
                        System.out.println(e);
                }
         Mythread3 t3=new Mythread3();
         t3.start();
                }
}
```



Did we achieve multithreading through this program? yes/no
yes

The output shows that as soon as thread 1 gets the processor to perform its task, it forces the JVM to complete its task (because the join () method) and after the full execution of the t1 thread it

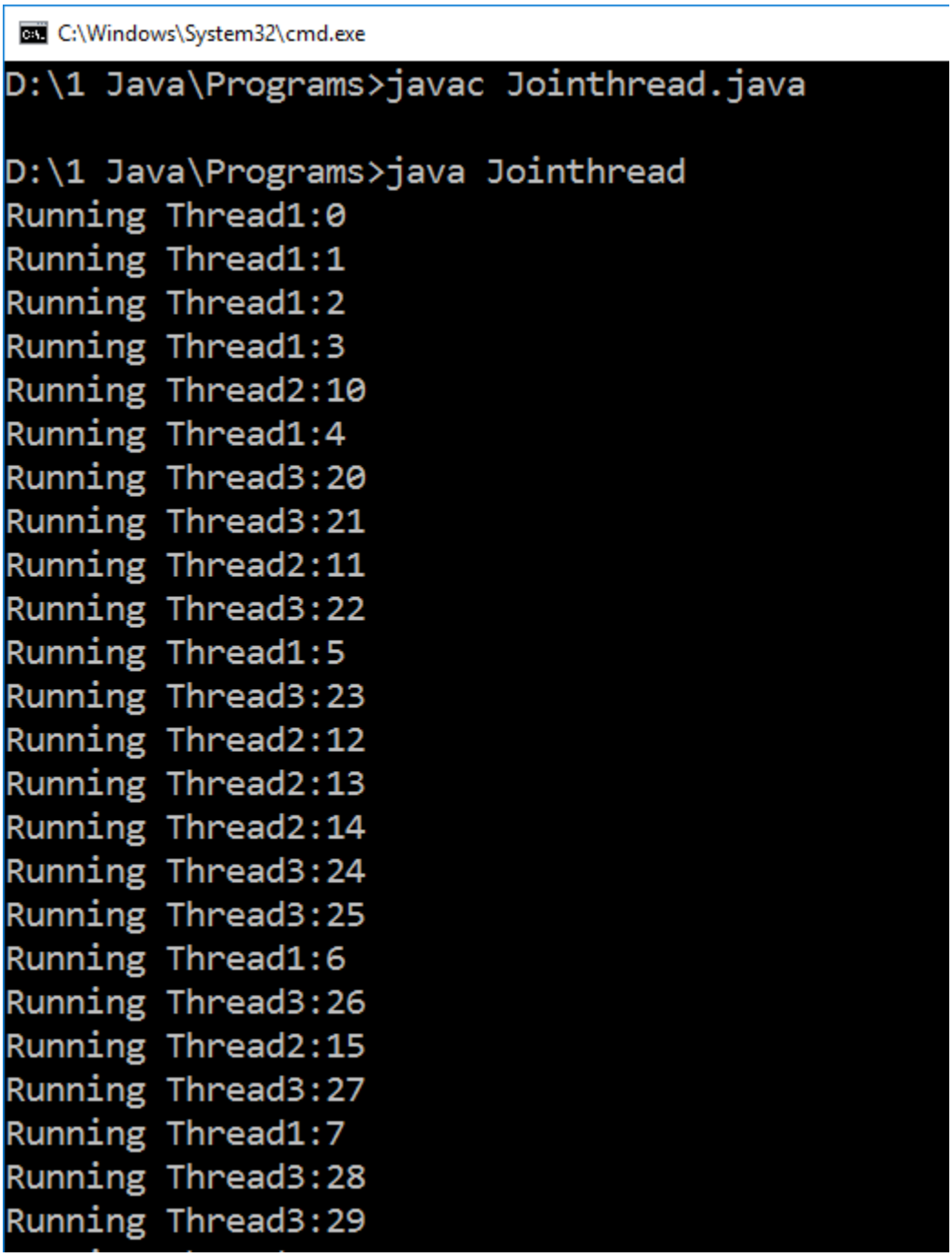
releases the processor. After that, thread 2 gets a chance, it finishes due to the join method, and finally, thread 3 executes its task.

This program shows how the join(long milliseconds) method works. In this example, when t1 completes its task for 10 milliseconds, then r2 and r3 start executing.

```
class Mythread1 implements Runnable {
        public void run() {
                for(int i=0;i<10;i++) {

                        System.out.println("Running Thread1:"+i); }
        } }
class Mythread2 extends Thread {
        public void run() {
                for(int i=10;i<20;i++)

                {
        System.out.println("Running Thread2:"+i); }
        } }
        class Mythread3 extends Thread {
        public void run() {
                for(int i=20;i<30;i++)
                {
        System.out.println("Running Thread3:"+i); }
        } }
class Jointhread {
        public static void main(String arg[]){
    Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
       t1.start();
                try{
          t1.join(10);  //it force to complete the task of t1 thread for 10 milli sec.
                        }
                catch(Exception e)
                {
                        System.out.println(e);
                }
    Mythread2 r2=new Mythread2();
         r2.start();
         Mythread3 r3=new Mythread3();
         r3.start();
                }
}
```


C:\Windows\System32\cmd.exe
D:\1 Java\Programs>javac Jointhread.java
D:\1 Java\Programs>java Jointhread
Running Thread1:0
Running Thread1:1
Running Thread1:2
Running Thread1:3
Running Thread2:10
Running Thread1:4
Running Thread3:20
Running Thread3:21
Running Thread2:11
Running Thread3:22
Running Thread1:5
Running Thread3:23
Running Thread2:12
Running Thread2:13
Running Thread2:14
Running Thread3:24
Running Thread3:25
Running Thread1:6
Running Thread3:26
Running Thread2:15
Running Thread3:27
Running Thread1:7
Running Thread3:28
Running Thread3:29

getName(),setName(String) and getId() ,currentThread() methods:

```
class Mythread1 implements Runnable {
        public void run() {
                System.out.println("Thread 1 is running");
System.out.println("Current Thread    is:"+Thread.currentThread().getName());
        } }

class Mythread2 extends Thread {
        public void run() {
                System.out.println("Thread 2 is running");
System.out.println("Current Thread is:"+Thread.currentThread().getName());
       } }


class Runthread3 {
  public static void main(String arg[]){
        Mythread1 r1=new Mythread1();
         Thread t1=new Thread(r1,"thread1");
          t1.start();
         Mythread2 r2=new Mythread2();
         Thread t2=new Thread(r2,"thread2");
        t2.start();

        System.out.println("Name of t1:"+t1.getName());
        System.out.println("Name of t2:"+t2.getName());
        System.out.println("id of t1:"+t1.getId());
        System.out.println("id of t2:"+t2.getId());
        t1.setName("IFM");
       System.out.println("After changing name of t1:"+t1.getName());
       System.out.println("Current Thread is:"+Thread.currentThread().getName());
                }}
```

```
D:\1 Java\Programs>java Runthread
Thread 1 is running
Name of t1:thread1
Thread 2 is running
Current Thread is:thread1
Current Thread is:thread2
Name of t2:thread2
id of t1:10
id of t2:12
After changing name of t1:IFM
Current Thread is:main
```