Ex 5 DAEMON PROGRAM

- A daemon (pronounced DEE-muhn) is a **program that runs continuously and exists for the purpose of handling periodic service requests that a computer system expects to receive**.
- The daemon program forwards the requests to other programs (or processes) as appropriate.

Daemon thread in Java

Daemon thread is a low priority thread that runs in background to perform tasks such as garbage collection.

Properties:

- They can not prevent the JVM from exiting when all the user threads finish their execution.
- JVM terminates itself when all user threads finish their execution
- If JVM finds running daemon thread, it terminates the thread and after that shutdown itself.
- JVM does not care whether Daemon thread is running or not.
- It is an utmost low priority thread.

Methods:

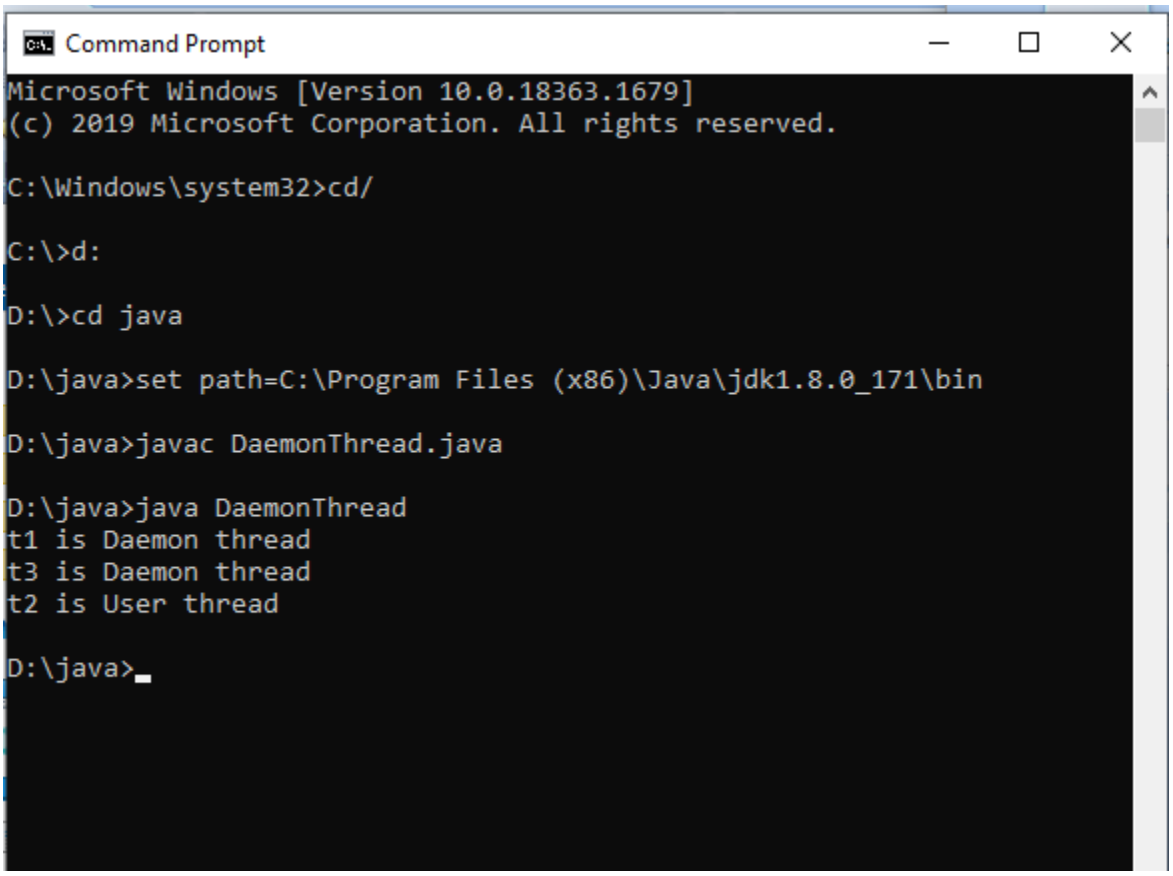
1. **void setDaemon(boolean status):** This method is used to mark the current thread as daemon thread or user thread. For example if I have a user thread tU then tU.setDaemon(true) would make it Daemon thread.
2. On the other hand if I have a Daemon thread tD then by calling tD.setDaemon(false) would make it user thread.
   Syntax:
3. **public final void setDaemon(boolean on)**
4. **parameters:**
5. **on :** if true, marks this thread as a daemon thread.
6. **exceptions:**
7. **IllegalThreadStateException**: if only this thread is active.
   SecurityException: if the current thread cannot modify this thread.
8. **booleanisDaemon():**
   This method is used to check that current is daemon. It returns true if the thread is Daemon else it returns false.
   Syntax:
   public final booleanisDaemon()
   returns:
   This method returns true if this thread is a daemon thread;
   false otherwise

```
// Java program to demonstrate the usage of
// setDaemon() and isDaemon() method.
public class DaemonThread extends Thread
{
        public DaemonThread(String name){
                super(name);
        }
        public void run()
        {
                // Checking whether the thread is Daemon or not
                if(Thread.currentThread().isDaemon())
```

```
		{
			System.out.println(getName() + " is Daemon thread");
		}
		else
		{
			System.out.println(getName() + " is User thread");
		}
	}
	public static void main(String[] args)
	{
		DaemonThread t1 = new DaemonThread("t1");
		DaemonThread t2 = new DaemonThread("t2");
		DaemonThread t3 = new DaemonThread("t3");
		// Setting user thread t1 to Daemon
		t1.setDaemon(true);
		// starting first 2 threads
		t1.start();
		t2.start();
		// Setting user thread t3 to Daemon
		t3.setDaemon(true);
		t3.start();
	}
}
```

Output:

t1 is Daemon thread
t3 is Daemon thread
t2 is User thread



Exceptions in Daemon thread

If you call the setDaemon() method after starting the thread, it would throw **IllegalThreadStateException**

```
// Java program to demonstrate the usage of
// exception in Daemon() Thread
public class DaemonThread extends Thread
{
	public void run()
	{
		System.out.println("Thread name: " + Thread.currentThread().getName());
		System.out.println("Check if its DaemonThread: "
					+ Thread.currentThread().isDaemon());
	}
	public static void main(String[] args)
	{
		DaemonThread t1 = new DaemonThread();
		DaemonThread t2 = new DaemonThread();
		t1.start();
		// Exception as the thread is already started
		t1.setDaemon(true);
		t2.start();
	}
}
```

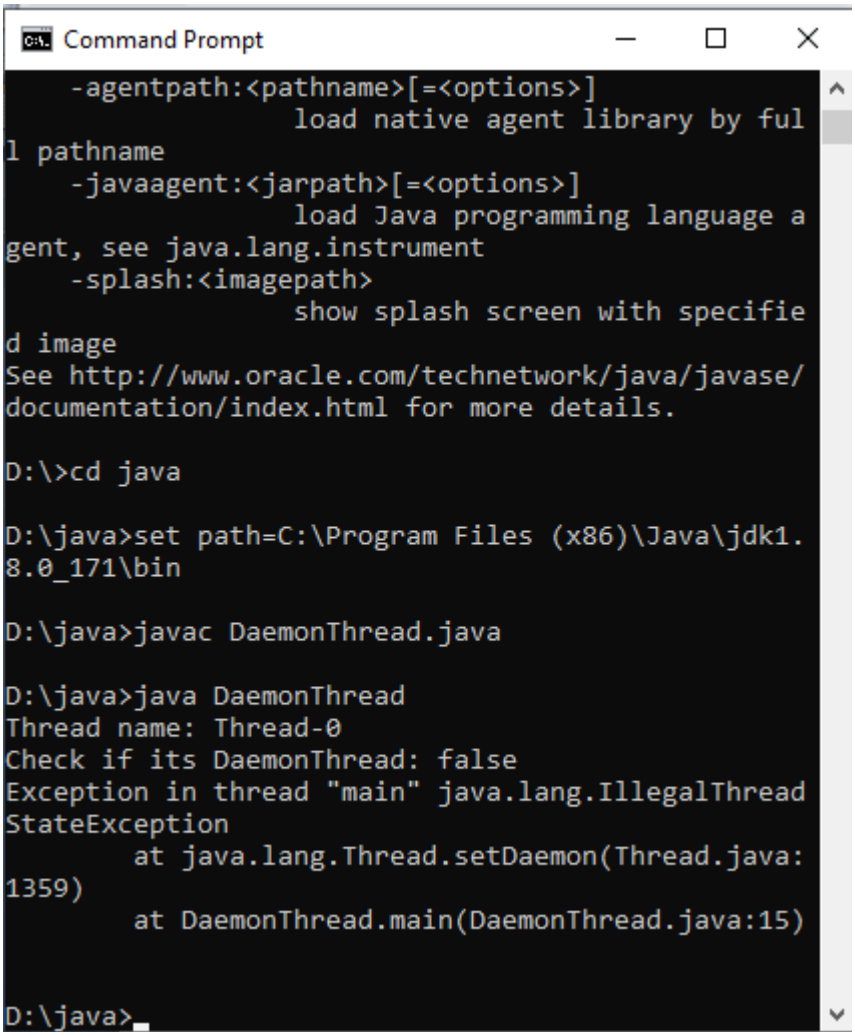
Runtime exception:

```
Exception in thread "main" java.lang.IllegalThreadStateException
   at java.lang.Thread.setDaemon(Thread.java:1352)
   at DaemonThread.main(DaemonThread.java:19)
```

Output:

```
Thread name: Thread-0
Check if its DaemonThread: false
```

This clearly shows that we cannot call the setDaemon() method after starting the thread.



Daemon vs User Threads

1. **Priority:** When the only remaining threads in a process are daemon threads, the interpreter exits. This makes sense because when only daemon threads remain, there is no other thread for which a daemon thread can provide a service.
2. **Usage:** Daemon thread is to provide services to user thread for background supporting task.