//Program on ITC using notifyAll() :

```java
package com.ravi.itc;

class Resource
{
    private boolean flag = false;

    public synchronized void waitMethod()
    {
        System.out.println("Wait");

        while (!flag)   //Infinite loop
        {
          try
         {
             System.out.println(Thread.currentThread().getName() + " is waiting...");
             System.out.println(Thread.currentThread().getName()+" is Waiting for 
Notification");
             wait(); //child1  child2  child3
          }
         catch (InterruptedException e)
         {
                e.printStackTrace();
          }
        }

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


    public synchronized void setMethod()
    {
        System.out.println("notifyAll");
        this.flag = true;
         System.out.println(Thread.currentThread().getName() + " has make flag value as a 
true");
         notifyAll(); // Notify all waiting threads that the signal is set
    }
}

public class ITCDemo6
{
    public static void main(String[] args)
        {

         Resource r1 = new Resource(); //lock is created

         Thread t1 = new Thread(() -> r1.waitMethod(), "Child1");
        Thread t2 = new Thread(() -> r1.waitMethod(), "Child2");
        Thread t3 = new Thread(() -> r1.waitMethod(), "Child3");

        t1.start();
         t2.start();
         t3.start();


        Thread setter = new Thread(() -> r1.setMethod(), "Setter_Thread");

           try
           {
                Thread.sleep(2000);
            }
           catch (InterruptedException e)
           {
                e.printStackTrace();
            }

           setter.start();
    }
}
```

ThreadGroup :

It is a predefined class available in java.lang Package.

By using ThreadGroup class we can put 'n' number of threads into a single group to perform some common/different operation.

By using ThreadGroup class constructor, we can assign the name of group under which all the thread will be executed.

ThreadGroup tg = new ThreadGroup(String groupName);

ThreadGroup class has provided the following methods :

public String getName() : To get the name of the Group

pubic int activeCount() : How many threads are alive and running under that particular group.

Thread class has provided constructor to put the thread into particular group.

Thread t1 = new Thread(ThreadGroup tg, Runnable target, String name);

By using ThreadGroup class, multiple threads will be executed under single group.

```java
package com.ravi.thread_group;

class UserThread implements Runnable
{
    @Override
    public void run()
    {
        String name = Thread.currentThread().getName();

        for(int i=1; i<=10; i++)
        {
            System.out.println(i+" by "+name+" thread");
        }
    }
}

public class ThreadGroupDemo1
{
    public static void main(String[] args) throws InterruptedException
    {
       ThreadGroup group = new ThreadGroup("Batch_46");

       Thread t1 = new Thread(group, new UserThread(), "Child1");
       Thread t2 = new Thread(group, new UserThread(), "Child2");
       Thread t3 = new Thread(group, new UserThread(), "Child3");
       Thread t4 = new Thread(group, new UserThread(), "Child4");

       t1.start(); t2.start();  t3.start();  t4.start();


       //Group Information
       System.out.println("Group Name is :"+group.getName());
       System.out.println("Active threads are :"+group.activeCount());

    }
}
```

```java
package com.ravi.thread_group;

class Tatkal implements Runnable
{
    @Override
    public void run()
    {
    String name = Thread.currentThread().getName();
    System.out.println(name+" has booked ticket under Tatkal Scheme");

    }

}

class PremimumTatkal implements Runnable
{
    @Override
    public void run()
    {
    String name = Thread.currentThread().getName();
    System.out.println(name+" has booked ticket under Premimum Tatkal Scheme");

    }

}
public class ThreadGroupDemo2 {

    public static void main(String[] args)
    {
        ThreadGroup tatkal = new ThreadGroup("Tatkal");
        ThreadGroup premimumTatkal = new ThreadGroup("PremimumTatkal");


        Thread scott = new Thread(premimumTatkal, new PremimumTatkal(), "Mr Scott");
        Thread smith = new Thread(premimumTatkal, new PremimumTatkal(), "Mr Smith");

        Thread alen = new Thread(tatkal, new Tatkal(), "Mr Alen");
        Thread john = new Thread(tatkal, new Tatkal(), "Mr John");

        scott.start();  smith.start();  alen.start(); john.start();
    }

}
```

```java
package com.ravi.thread_group;

public class ThreadGroupDemo3
{
    public static void main(String[] args)
    {
        Thread t = Thread.currentThread();
        System.out.println(t.toString());
    }

}
```

Note : From the above program IT is clear that main thread is running under main group

---

Daemon Thread [Service Level Thread]

Daemon thread is a low- priority thread which is used to provide background maintenance.

The main purpose of of Daemon thread to provide services to the user thread.

JVM can't terminate the program till any of the non-daemon (user) thread is active, once all the user thread will be completed then JVM will automatically terminate all Daemon threads, which are running in the background to support user threads.

The example of Daemon thread is Garbage Collection thread, which is running in the background for memory management.

In order to make a thread as a Daemon thread as well as to verify whether a thread is daemon thread or not, Java software people has provided the following two methods

1) public final void setDaemon(boolean on) : If the boolean value
   is true the thread will work as a Daemon thread.

2) public final boolean isDaemon() : Will verify whether the
   thread is daemon thread OR user thread.

```java
package com.ravi.daemon;

public class DaemonThreadDemo1
{
  public static void main(String[] args)
    {
        System.out.println("Main Thread Started...");

         Thread daemonThread = new Thread(() ->
        {
             while (true)
            {
                System.out.println("Daemon Thread is running...");
                try
              {
                    Thread.sleep(1000);
                }
              catch (InterruptedException e)
              {
                    e.printStackTrace();
                }
            }
         });

         daemonThread.setDaemon(true);
         daemonThread.start();


         Thread userThread = new Thread(() ->
        {
             for (int i=1; i<=15; i++)
            {
                System.out.println("User Thread: " + i);
                try
              {
                    Thread.sleep(2000);
                }
              catch (InterruptedException e)
              {
                    e.printStackTrace();
                }
            }
         });

         userThread.start();

         System.out.println("Main Thread Ended...");
    }
}
```

```java
package com.ravi.daemon;

public class DaemonThread2
{
    public static void main(String[] args)
    {
        Thread t = Thread.currentThread();
        System.out.println(t.isDaemon()); //false

        t.setDaemon(true);
        System.out.println(t.isDaemon()); //java.lang.IllegalThreadStateException

    }

}
```

---

public void interrupt() Method of Thread class :
