Understanding Synchronization

- Synchronization in java controls multiple threads from accessing the same shared resource in order to prevent an inconsistent state.
- Java Synchronization is done when we want to allow only one thread to access the shared resource.
- In other words Synchronization is a process of making only one thread access a resource, where multiple threads are trying to access the same resource, and moving all the remaining threads in to waiting state.

Advantage:- Resolves Thread Interference & Memory Consistency problems

Disadvantage:- Increases Thread waiting time.

- We can use Synchronization in two ways, a method can be synchronized and a block can be synchronized.
- We can't synchronize a complete class.

```
3  public class ClassA extends Thread
4  {
5      @Override
6      public void run()
7      {
8          String name=Thread.currentThread().getName();
9          System.out.println(name+" has entered run()");
10         for(int i=1;i<=5;i++)
11         {
12             System.out.println(name+" : "+i);
13         }
14         System.out.println(name+" Completed executing run()");
15     }
16     public static void main(String[] args)
17     {
18         ClassA aobj=new ClassA();
19
20         Thread t1=new Thread(aobj);
21         Thread t2=new Thread(aobj);
22
23         t1.setName("Tom-Thread");
24         t2.setName("Jerry-Thread");
25
26         t1.start();
27         //t2.start();
```

```
Tom-Thread has entered run()
Tom-Thread : 1
Tom-Thread : 2
Tom-Thread : 3
Tom-Thread : 4
Tom-Thread : 5
Tom-Thread Completed executing run()
```

```java
public class ClassA extends Thread
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("Tom-Thread");
        t2.setName("Jerry-Thread");

        //t1.start();
        t2.start();
```

```
Jerry-Thread has entered run()
Jerry-Thread : 1
Jerry-Thread : 2
Jerry-Thread : 3
Jerry-Thread : 4
Jerry-Thread : 5
Jerry-Thread Completed executing run()
```

```java
public class ClassA extends Thread
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("Tom-Thread");
        t2.setName("Jerry-Thread");

        t1.start();
        t2.start();
```

```
Tom-Thread has entered run()
Jerry-Thread has entered run()
Jerry-Thread : 1
Jerry-Thread : 2
Jerry-Thread : 3
Jerry-Thread : 4
Jerry-Thread : 5
Jerry-Thread Completed executing run()
Tom-Thread : 1
Tom-Thread : 2
Tom-Thread : 3
Tom-Thread : 4
Tom-Thread : 5
Tom-Thread Completed executing run()
```

```java
public class ClassA extends Thread
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("Tom-Thread");
        t2.setName("Jerry-Thread");

        t1.start();
        t2.start();
```

```
Tom-Thread has entered run()
Jerry-Thread has entered run()
Tom-Thread : 1
Jerry-Thread : 1
Tom-Thread : 2
Jerry-Thread : 2
Tom-Thread : 3
Tom-Thread : 4
Jerry-Thread : 3
Tom-Thread : 5
Jerry-Thread : 4
Jerry-Thread : 5
Jerry-Thread Completed executing run()
Tom-Thread Completed executing run()
```

```java
{
    @Override
    public void run()
    {
        criticalResource();
    }
    synchronized void criticalResource()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("Tom-Thread");
        t2.setName("Jerry-Thread");
```

```
Tom-Thread has entered run()
Tom-Thread : 1
Tom-Thread : 2
Tom-Thread : 3
Tom-Thread : 4
Tom-Thread : 5
Tom-Thread Completed executing run()
Jerry-Thread has entered run()
Jerry-Thread : 1
Jerry-Thread : 2
Jerry-Thread : 3
Jerry-Thread : 4
Jerry-Thread : 5
Jerry-Thread Completed executing run()
```

```java
{
    @Override
    public void run()
    {
        criticalResource();
    }
    synchronized void criticalResource()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("Tom-Thread");
        t2.setName("Jerry-Thread");
```

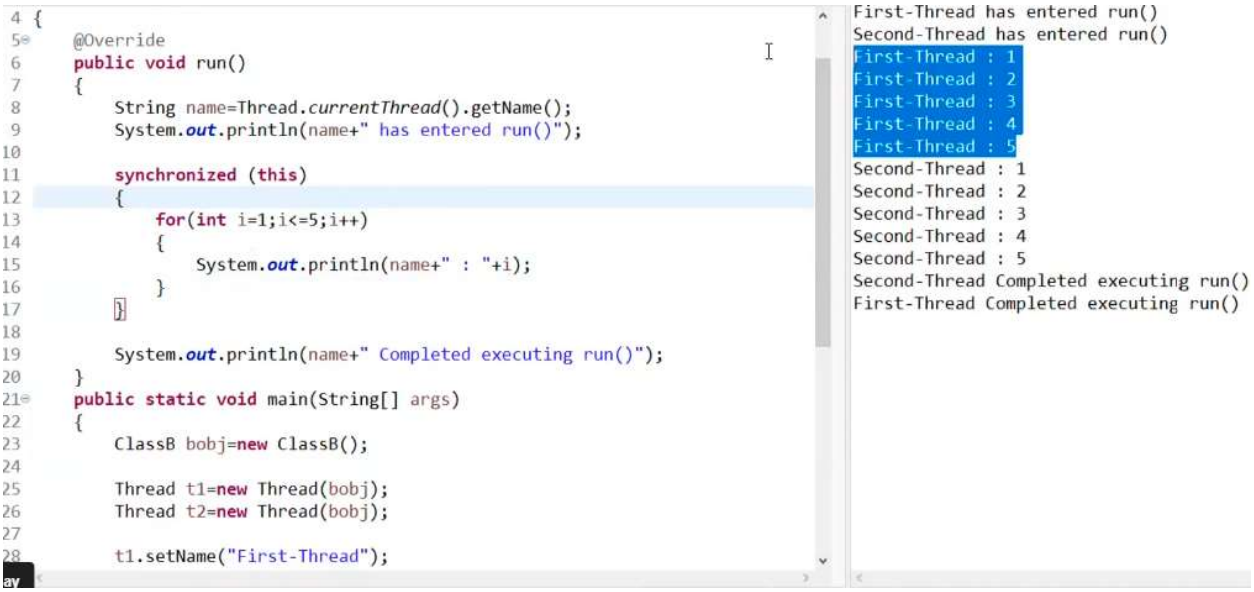
```
Jerry-Thread has entered run()
Jerry-Thread : 1
Jerry-Thread : 2
Jerry-Thread : 3
Jerry-Thread : 4
Jerry-Thread : 5
Jerry-Thread Completed executing run()
Tom-Thread has entered run()
Tom-Thread : 1
Tom-Thread : 2
Tom-Thread : 3
Tom-Thread : 4
Tom-Thread : 5
Tom-Thread Completed executing run()
```

```
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassB bobj=new ClassB();

        Thread t1=new Thread(bobj);
        Thread t2=new Thread(bobj);

        t1.setName("First-Thread");
        t2.setName("Second-Thread");

        t1.start();
        t2.start();
    }
```

```
Second-Thread has entered run()
First-Thread has entered run()
Second-Thread : 1
Second-Thread : 2
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
Second-Thread Completed executing run()
First-Thread : 1
First-Thread : 2
First-Thread : 3
First-Thread : 4
First-Thread : 5
First-Thread Completed executing run()
```

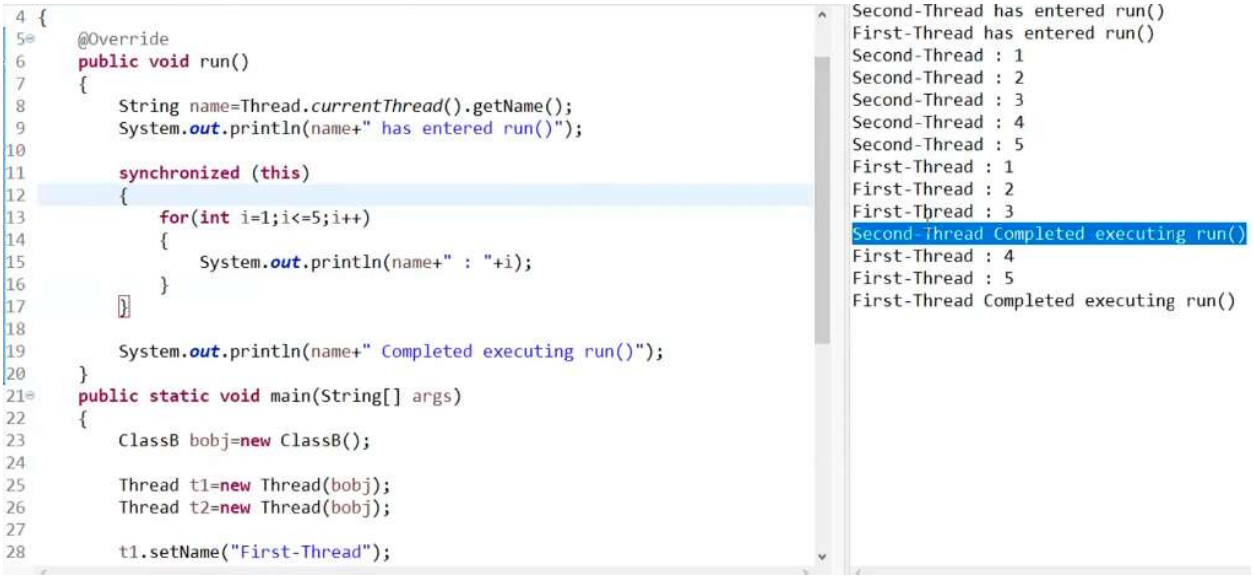
```
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");

        synchronized (this)
        {
            for(int i=1;i<=5;i++)
            {
                System.out.println(name+" : "+i);
            }
        }

        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassB bobj=new ClassB();

        Thread t1=new Thread(bobj);
        Thread t2=new Thread(bobj);

        t1.setName("First-Thread");
```

```
Second-Thread has entered run()
First-Thread has entered run()
Second-Thread : 1
Second-Thread : 2
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
First-Thread : 1
First-Thread : 2
First-Thread : 3
First-Thread : 4
First-Thread : 5
First-Thread Completed executing run()
Second-Thread Completed executing run()
```



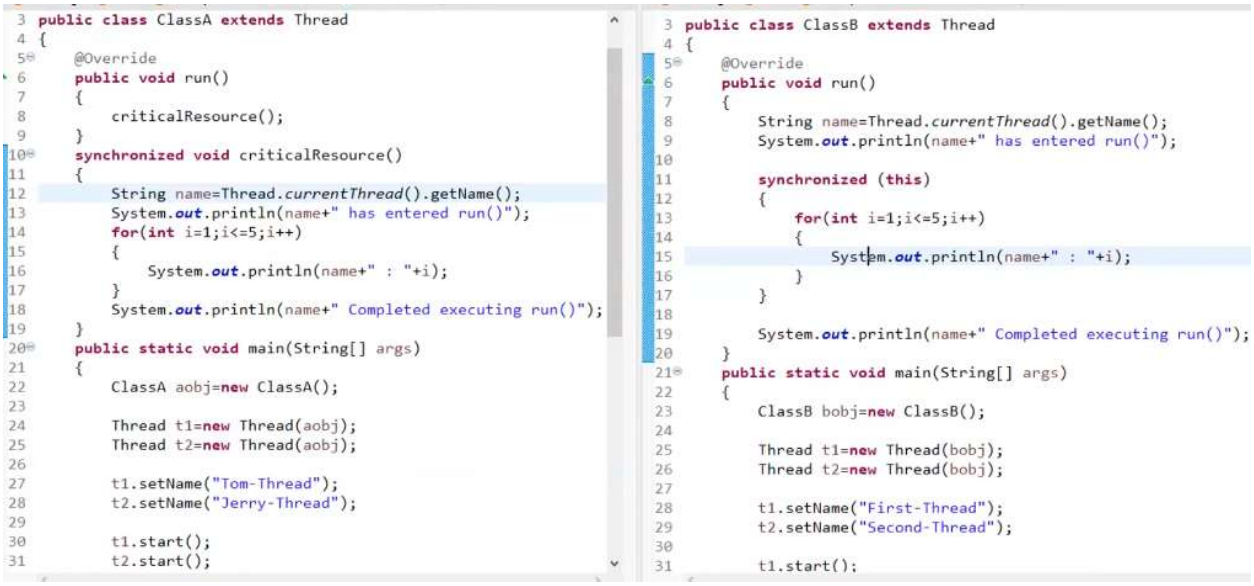
```
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");

        synchronized (this)
        {
            for(int i=1;i<=5;i++)
            {
                System.out.println(name+" : "+i);
            }
        }

        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassB bobj=new ClassB();

        Thread t1=new Thread(bobj);
        Thread t2=new Thread(bobj);

        t1.setName("First-Thread");
```

```
First-Thread has entered run()
Second-Thread has entered run()
First-Thread : 1
First-Thread : 2
First-Thread : 3
First-Thread : 4
First-Thread : 5
Second-Thread : 1
Second-Thread : 2
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
Second-Thread Completed executing run()
First-Thread Completed executing run()
```



```
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");

        synchronized (this)
        {
            for(int i=1;i<=5;i++)
            {
                System.out.println(name+" : "+i);
            }
        }

        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassB bobj=new ClassB();

        Thread t1=new Thread(bobj);
        Thread t2=new Thread(bobj);

        t1.setName("First-Thread");
```

```
Second-Thread has entered run()
First-Thread has entered run()
Second-Thread : 1
Second-Thread : 2
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
First-Thread : 1
First-Thread : 2
First-Thread : 3
Second-Thread Completed executing run()
First-Thread : 4
First-Thread : 5
First-Thread Completed executing run()
```



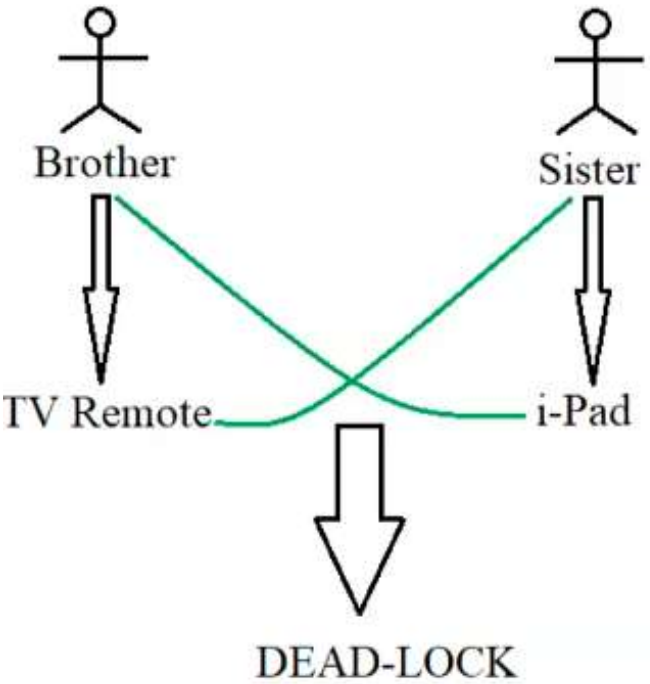
```java
public class ClassA extends Thread
{
    @Override
    public void run()
    {
        criticalResource();
    }
    synchronized void criticalResource()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("Tom-Thread");
        t2.setName("Jerry-Thread");

        t1.start();
        t2.start();
```

```java
public class ClassB extends Thread
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");

        synchronized (this)
        {
            for(int i=1;i<=5;i++)
            {
                System.out.println(name+" : "+i);
            }
        }

        System.out.println(name+" Completed executing run()");
    }
    public static void main(String[] args)
    {
        ClassB bobj=new ClassB();

        Thread t1=new Thread(bobj);
        Thread t2=new Thread(bobj);

        t1.setName("First-Thread");
        t2.setName("Second-Thread");

        t1.start();
```

```java
    public static void main(String[] args)
    {
        ClassB bobj=new ClassB();

        Thread t1=new Thread(bobj);
        Thread t2=new Thread(bobj);

        t1.setName("First-Thread");
        t2.setName("Second-Thread");

        t1.start();
        t2.start();
    }
}
```

Understanding Deadlocks

- Deadlock describes a situation where two or more threads are blocked forever, & waiting for each other.
- In other words it is a condition which occurs when two or more threads get blocked, waiting for each other for an infinite period of time to release the resources they hold.



```java
public class ClassD
{
    public static void main(String[] args)
    {
        final String A="Java";
        final String B="Python";

        Thread t1=new Thread()
        {  // Anononmyous Inner Class starts here
            @Override
            public void run()
            {
                synchronized (A)// Thread 1 is holding Java
                {
                    System.out.println("Thread 1 locked on A");
                    try
                    {
                        Thread.sleep(100);
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                    synchronized ( B)// Thread1 will be waiting for B (Python)
                    {

                        System.out.println("Thread 1 locked on B");
                    }

                    System.out.println("no dead lock");
                }
            }
        }// Anononmyous Inner Class Ends here
        ;
        Thread t2=new Thread()
        {
            @Override
            public void run()
            {
                synchronized (B)// Thread 2 is holding Python
                {
                    System.out.println("Thread 2 locked on B");
                    try
                    {
                        Thread.sleep(100);
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
```

```
52                      synchronized (A)// Thread2 will be waiting for A (Java)
53                      {
54                          System.out.println("Thread 2 locked on A");
55                      }
56                  }
57                  System.out.println("no dead lock");
58              }
59          }
60          ;
61          t1.start();
62          t2.start();
63      }
64  }
```

Refer to next class about above program