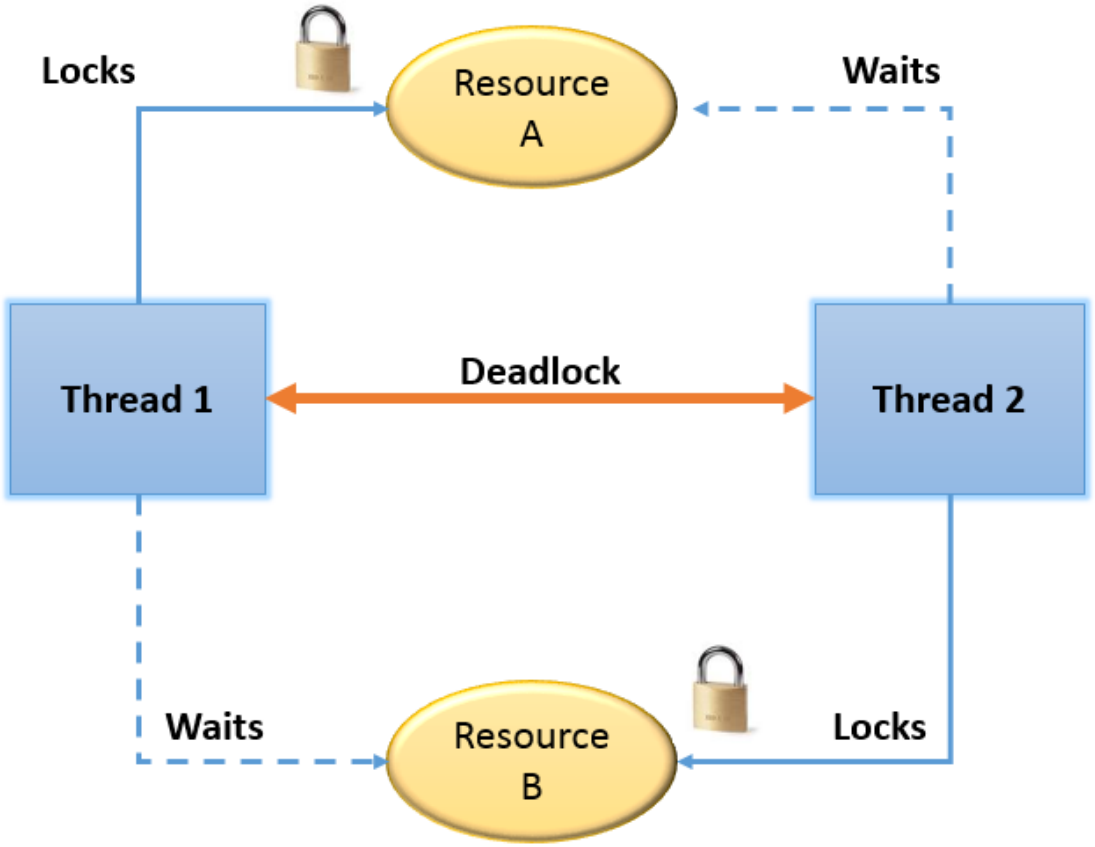
DEADLOCK

Deadlock in java is a situation where **two or more threads are blocked forever,** waiting for each other. Deadlock occurs when multiple threads need the same locks but obtain them in different order. A Java multithreaded program may suffer from the deadlock condition because the **synchronized** keyword causes the executing thread to block while waiting for the lock, or monitor, associated with the specified object.

Consider the below diagram, thread1 is waiting for an object lock, that is acquired by another thread2 and second thread2 is waiting for an object lock that is acquired by first thread1. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.

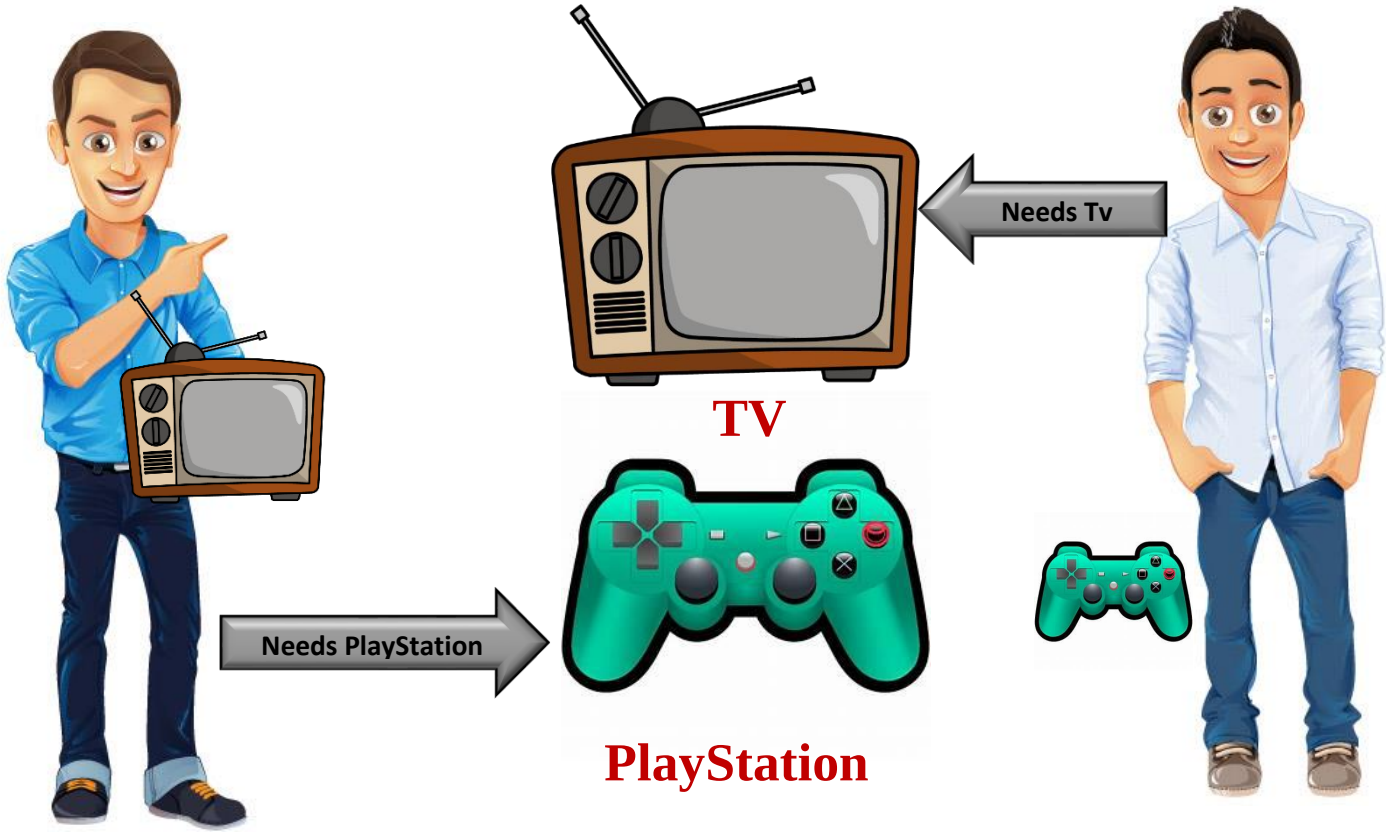


The QWERTY keyboard was designed to slow you down. If you want to type faster, try the Dvorak Keyboard.

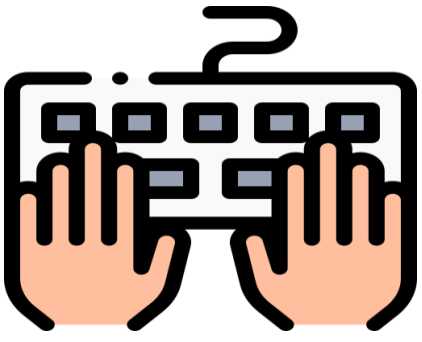
Let us consider a scenario to understand deadlocks in detail

Scenario:

There are two threads rohit and vijay and there are two resources TV and PlayStation. Both the threads now want to acquire both the resources. Rohit acquires resource2 first which is PlayStation. Vijay acquires resource1 first which is TV. Now vijay wants playstation which is already acquired by rohit and rohit wants Tv which is already acquired by vijay. In this way, they enter into a state where both the resources are locked and threads enter into deadlock.



Let us start coding the above scenario



```
class Family implements Runnable
{
    String resource1 = "Tv";
    String resource2 = "Playstation";
    public void run()
    {
        String name = Thread.currentThread().getName();
        if(name.equals("Rohit") == true)
        {
            rohitAccquiredResource();
        }
        else
        {
            vijayAccquiredResource();
        }
    }
    void rohitAccquiredResource()
    {
        synchronized(resource2)
        {
            try
            {
                System.out.println("Rohit accquired Platstation");
                Thread.sleep(1000);
                synchronized(resource1)
                {
                    System.out.println("Rohit accquired Tv");
                    Thread.sleep(1000);
                }
            }
            catch (Exception e)
            {
                System.out.println("Rohit failed");
            }
        }
    }
    void vijayAccquiredResource()
    {
        synchronized(resource1)
        {
```

```
            try
            {
                System.out.println("Vijay accquired Tv");
                Thread.sleep(1000);
                synchronized(resource2)
                {
                    System.out.println("Vijay accquired Playstation");
                    Thread.sleep(1000);
                }
            }
            catch (Exception e)
            {
                System.out.println("Vijay failed");
            }
        }
    }
}
class  Launch1
{
    public static void main(String[] args)
    {
        Family f = new Family();
        Thread t1 = new Thread(f);
        Thread t2 = new Thread(f);
        t1.setName("Rohit");
        t2.setName("Vijay");
        t1.start();
        t2.start();
    }
}
```

OUTPUT

Rohit acquired PlayStation

Vijay acquired Tv

Cursor keeps blinking as both the threads are locked and execution never proceeds.