```java
// A Java program to demonstrate working of
// synchronized.

import java.io.*;
import java.util.*;

// A Class used to send a message
class Sender
{
    public void send(String msg)
    {
        System.out.println("Sending\t"  + msg );
        try
        {
            Thread.sleep(1000);
        }
        catch (Exception e)
        {
            System.out.println("Thread  interrupted.");
        }
        System.out.println("\n" + msg + "Sent");
    }
}

// Class for send a message using Threads
class ThreadedSend extends Thread
{
    private String msg;
    Sender  sender;

    // Receives a message object and a string
    // message to be sent
    ThreadedSend(String m,  Sender obj)
    {
        msg = m;
        sender = obj;
    }

    public void run()
    {
        // Only one thread can send a message
        // at a time.
        synchronized(sender)
        {
            // synchronizing the send object
            sender.send(msg);
        }
    }
}

// Driver class
class SyncDemo
```

```
{
    public static void main(String args[])
    {
        Sender send = new Sender();
        ThreadedSend S1 =
            new ThreadedSend( " Hi " , send );
        ThreadedSend S2 =
            new ThreadedSend( " Bye " , send );

        // Start two threads of ThreadedSend type
        S1.start();
        S2.start();

        // wait for threads to end
        try
        {
            S1.join();
            S2.join();
        }
        catch(Exception e)
        {
            System.out.println("Interrupted");
        }
    }
}
```

Output

```
Sending     Hi

 Hi Sent
Sending     Bye

 Bye Sent
```

The output is the same every time we run the program.

In the above example, we choose to synchronize the Sender object inside the run() method of the ThreadedSend class. Alternately, we could define the **whole send() block as synchronized**, producing the same result. Then we don’t have to synchronize the Message object inside the run() method in ThreadedSend class.

```
// An alternate implementation to demonstrate
// that we can use synchronized with method also.
```

```
class Sender {
   public synchronized void send(String msg)
   {
       System.out.println("Sending\t" + msg);
       try {
           Thread.sleep(1000);
       }
       catch (Exception e) {
           System.out.println("Thread interrupted.");
       }
       System.out.println("\n" + msg + "Sent");
   }
}
```

We do not always have to synchronize a whole method. Sometimes it is preferable to **synchronize only part of a method**. Java synchronized blocks inside methods make this possible.

```
// One more alternate implementation to demonstrate
// that synchronized can be used with only a part of
// method

class Sender
{
   public void send(String msg)
   {
       synchronized(this)
       {
           System.out.println("Sending\t" + msg );
           try
           {
               Thread.sleep(1000);
           }
           catch (Exception e)
           {
               System.out.println("Thread interrupted.");
           }
           System.out.println("\n" + msg + "Sent");
       }
   }
}
```