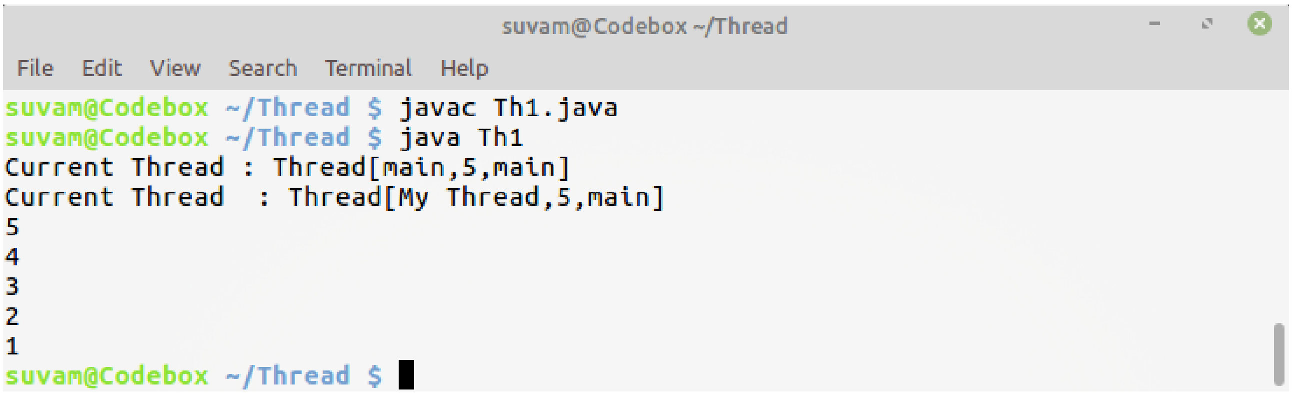
81. Controling the Main Thread.

Code:

```
class Th1{
     public static void main(String[] args){
          Thread t = Thread.currentThread();
          System.out.println("Current Thread : " + t);

          t.setName("My Thread");
          System.out.println("Current Thread  : " + t);

          try{
               for(int i=5; i>0; i--){
                    System.out.println(i);
                    Thread.sleep(1000);
               }
          }catch(InterruptedException e){
                    System.out.println("Child Exception");
               }
     }
}
```

Output:



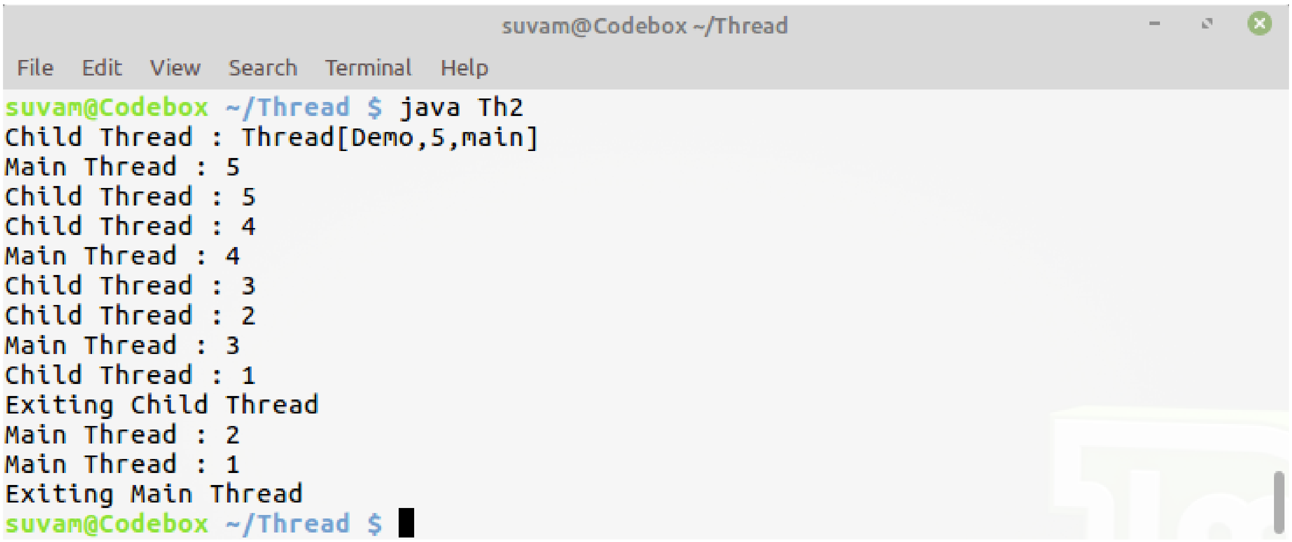
```
suvam@Codebox ~/Thread $ javac Th1.java
suvam@Codebox ~/Thread $ java Th1
Current Thread : Thread[main,5,main]
Current Thread  : Thread[My Thread,5,main]
5
4
3
2
1
suvam@Codebox ~/Thread $
```

82. Implementing Runnable.

Code:

```
class NewThread implements Runnable{
     Thread t;

     NewThread(){
          t = new Thread(this, "Demo");
          System.out.println("Child Thread : " + t);
          t.start();
     }

     public void run(){
               try{
                    for(int i=5;i>0;i--){
                         System.out.println("Child Thread : " + i);
                         Thread.sleep(500);
                    }
               }catch(InterruptedException e){
                    System.out.println("Child Exception");
               }

               System.out.println("Exiting Child Thread");
          }
}


class Th2{
     public static void main(String[] args){
          new NewThread();

          try{
               for(int i=5;i>0;i--){
                    System.out.println("Main Thread : " + i);
                    Thread.sleep(1000);
               }
          }catch(InterruptedException e){
                    System.out.println("Main Exception");
          }
          System.out.println("Exiting Main Thread");

     }
}
```

Output:



```
suvam@Codebox ~/Thread $ java Th2
Child Thread : Thread[Demo,5,main]
Main Thread : 5
Child Thread : 5
Child Thread : 4
Main Thread : 4
Child Thread : 3
Child Thread : 2
Main Thread : 3
Child Thread : 1
Exiting Child Thread
Main Thread : 2
Main Thread : 1
Exiting Main Thread
suvam@Codebox ~/Thread $
```

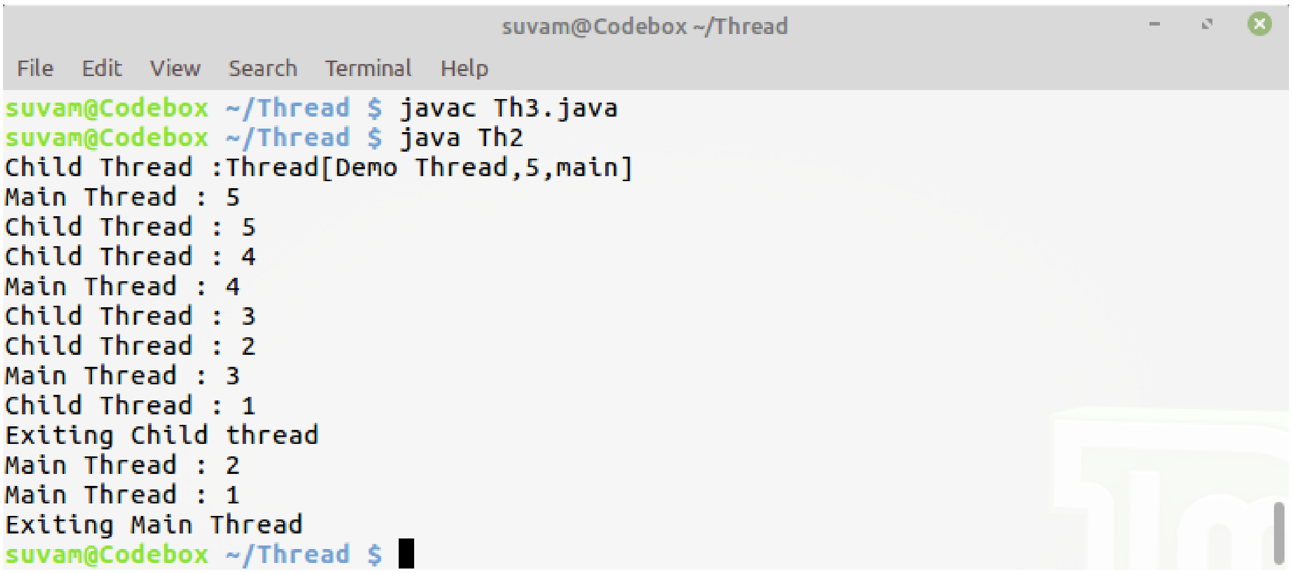
83. Entending Thread.

Code:

```
class NewThread extends Thread{
      NewThread(){
            super("Demo Thread");
            System.out.println("Child Thread :" + this);
            start();
      }

      public void run(){
            try{
                  for(int i=5;i>0;i--){
                        System.out.println("Child Thread : " + i);
                        Thread.sleep(500);
                  }
            }catch(InterruptedException e){
                        System.out.println("Child Exception");
            }

            System.out.println("Exiting Child thread");
      }
}
```

```
class Th3{
     public static void main(String[] args){
          new NewThread();

          try{
               for(int i=5; i>0; i--){
                    System.out.println("Main Thread : " + i);
                    Thread.sleep(1000);
               }
          }catch(InterruptedException e){
                    System.out.println("Main Exception");
          }

          System.out.println("Exiting Main thread");
     }
}
```

Output:



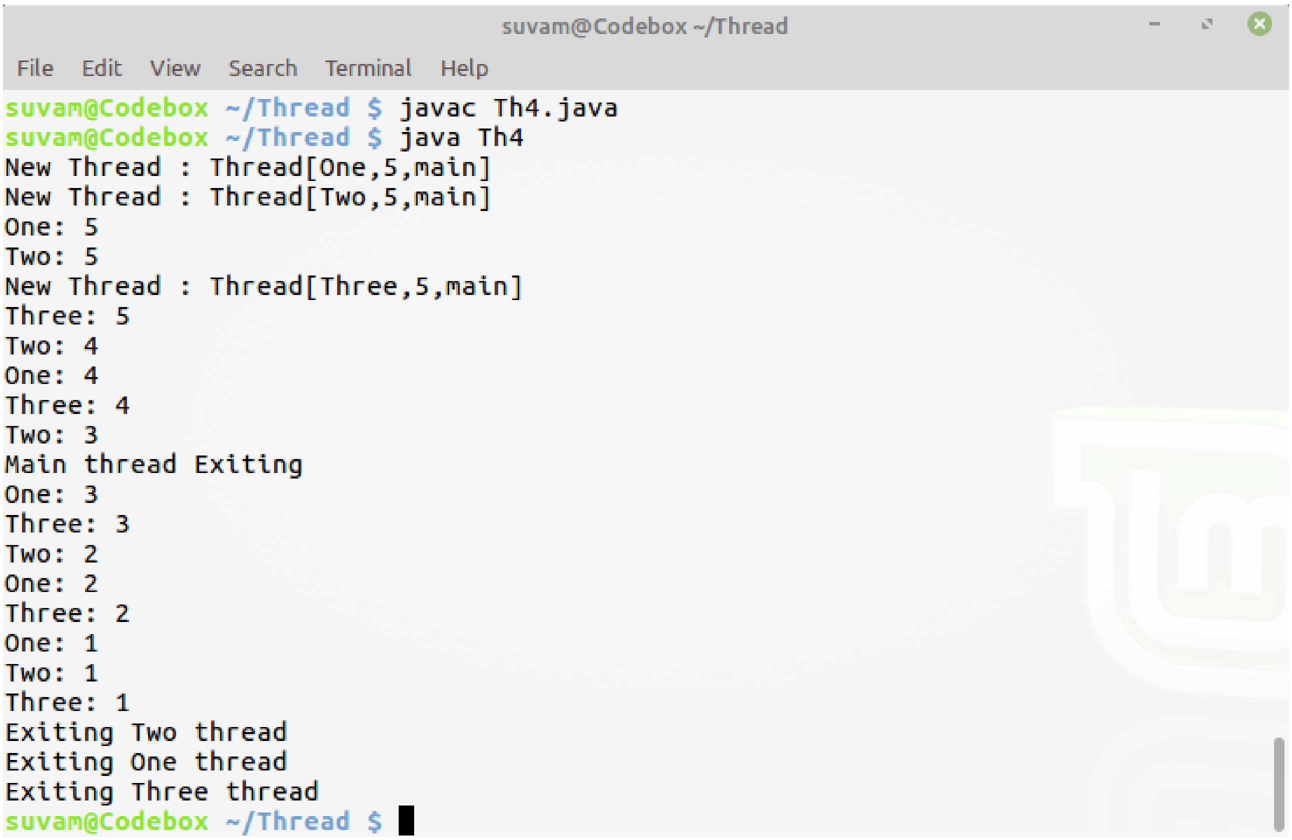
```
suvam@Codebox ~/Thread $ javac Th3.java
suvam@Codebox ~/Thread $ java Th2
Child Thread :Thread[Demo Thread,5,main]
Main Thread : 5
Child Thread : 5
Child Thread : 4
Main Thread : 4
Child Thread : 3
Child Thread : 2
Main Thread : 3
Child Thread : 1
Exiting Child thread
Main Thread : 2
Main Thread : 1
Exiting Main Thread
suvam@Codebox ~/Thread $
```

84. Creating Multiple Thread.

Code:

```
class NewThread implements Runnable{
     String name;
     Thread t;

     NewThread(String name){
          this.name = name;
          t = new Thread(this,name);
          System.out.println("New Thread : " + t);
          t.start();
     }

     public void run(){
          try{
               for(int i=5;i>0;i--){
                    System.out.println(name + ": " + i);
                    Thread.sleep(500);
               }
          }catch(InterruptedException e){
                    System.out.println(name  + " Exception");
          }

          System.out.println("Exiting "+ name +" thread");
     }
}

class Th4{
     public static void main(String[] args){
          new NewThread("One");
          new NewThread("Two");
          new NewThread("Three");

          try{
               Thread.sleep(1000);
          }catch(InterruptedException e){
                    System.out.println("Main thread Exception");
          }

          System.out.println("Main thread Exiting");
     }
}
```

Output:



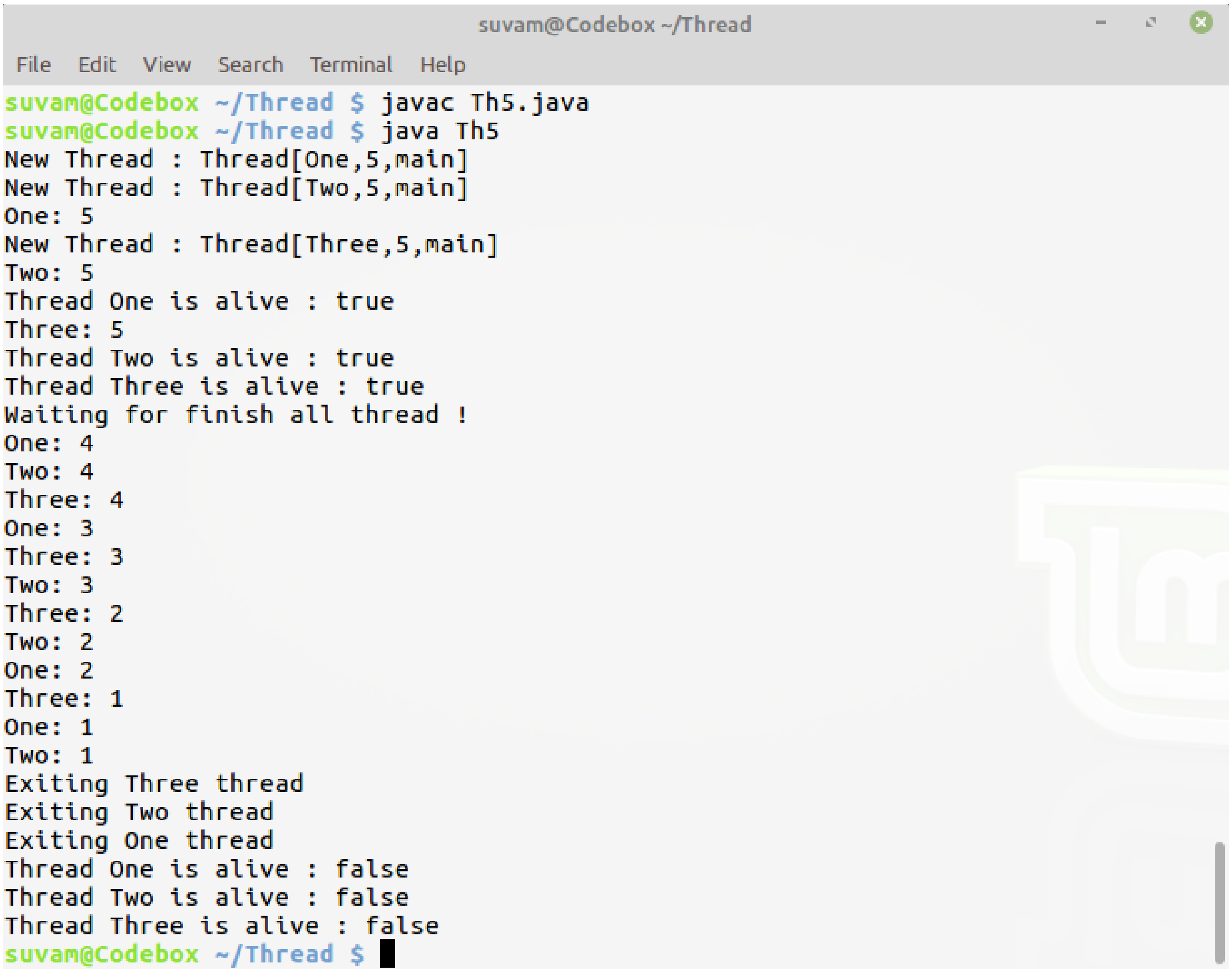
```
suvam@Codebox ~/Thread $ javac Th4.java
suvam@Codebox ~/Thread $ java Th4
New Thread : Thread[One,5,main]
New Thread : Thread[Two,5,main]
One: 5
Two: 5
New Thread : Thread[Three,5,main]
Three: 5
Two: 4
One: 4
Three: 4
Two: 3
Main thread Exiting
One: 3
Three: 3
Two: 2
One: 2
Three: 2
One: 1
Two: 1
Three: 1
Exiting Two thread
Exiting One thread
Exiting Three thread
suvam@Codebox ~/Thread $
```

85. Using join() to wait for Threads to finish.

Code:

```
class NewThread implements Runnable{
     String name;
     Thread t;

     NewThread(String name){
          this.name = name;
          t = new Thread(this,name);
          System.out.println("New Thread : " + t);
          t.start();
     }

     public void run(){
          try{
               for(int i=5;i>0;i--){
                    System.out.println(name + ": " + i);
                    Thread.sleep(1000);
               }
          }catch(InterruptedException e){
                    System.out.println(name  + " Exception");
          }

          System.out.println("Exiting "+ name +" thread");
     }
}

class Th5{
     public static void main(String[] args){
          NewThread ob1 = new NewThread("One");
          NewThread ob2 = new NewThread("Two");
          NewThread ob3 = new NewThread("Three");

          System.out.println("Thread One is alive : " +
ob1.t.isAlive());
          System.out.println("Thread Two is alive : " +
ob2.t.isAlive());
          System.out.println("Thread Three is alive : " +
ob3.t.isAlive());


          try{
               System.out.println("Waiting for finish all
thread !");
               ob1.t.join();
               ob2.t.join();
               ob3.t.join();
          }catch(InterruptedException e){
               System.out.println("Main Thread Interrupted");
          }

          System.out.println("Thread One is alive : " +
ob1.t.isAlive());
          System.out.println("Thread Two is alive : " +
ob2.t.isAlive());
          System.out.println("Thread Three is alive : " +
ob3.t.isAlive());

     }
}
```

Output:



```
suvam@Codebox ~/Thread $ javac Th5.java
suvam@Codebox ~/Thread $ java Th5
New Thread : Thread[One,5,main]
New Thread : Thread[Two,5,main]
One: 5
New Thread : Thread[Three,5,main]
Two: 5
Thread One is alive : true
Three: 5
Thread Two is alive : true
Thread Three is alive : true
Waiting for finish all thread !
One: 4
Two: 4
Three: 4
One: 3
Three: 3
Two: 3
Three: 2
Two: 2
One: 2
Three: 1
One: 1
Two: 1
Exiting Three thread
Exiting Two thread
Exiting One thread
Thread One is alive : false
Thread Two is alive : false
Thread Three is alive : false
suvam@Codebox ~/Thread $
```

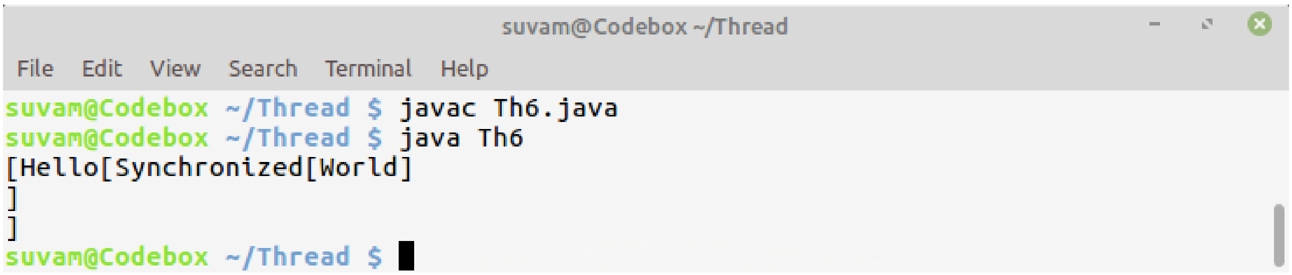
86. This Program is not Synchronized.

Code:

```
class Callme{
     void call(String msg){
          System.out.print("[" + msg);
          try{
               Thread.sleep(500);
          }catch(InterruptedException e){
               System.out.println("Inturrepted");
          }
          System.out.println("]");
     }
}

class Caller implements Runnable{
     String msg;
     Callme target;
     Thread t;

     Caller(Callme trg, String s){
          target = trg;
          msg = s;
          t = new  Thread(this);
          t.start();
     }

     public void run(){
          target.call(msg);
     }
}

class Th6{
     public static void main(String[] args){
          Callme target = new Callme();
          Caller ob1 = new Caller(target,"Hello");
          Caller ob2 = new Caller(target,"Synchronized");
          Caller ob3 = new Caller(target,"World");

          try{
               ob1.t.join();
               ob2.t.join();
               ob3.t.join();
          }catch(InterruptedException e){
               System.out.println("Interrupted");
          }
     }
```

}

Output:



```
suvam@Codebox ~/Thread $ javac Th6.java
suvam@Codebox ~/Thread $ java Th6
[Hello[Synchronized[World]
]
]
suvam@Codebox ~/Thread $ 
```

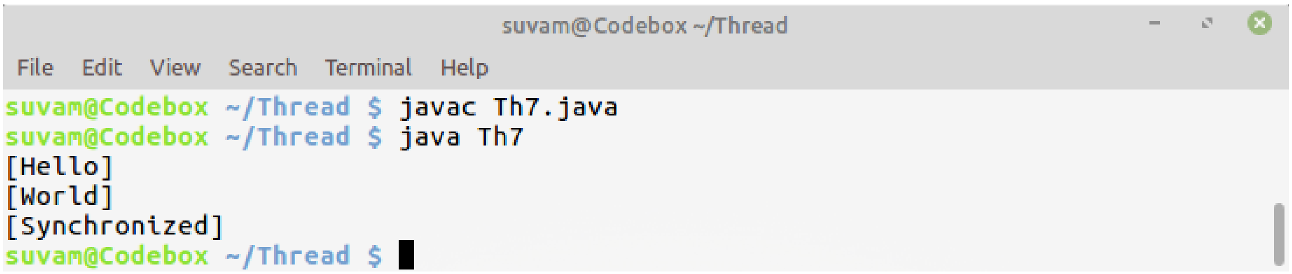
87. The Synchronized statement.

Code:

```
class Callme{
     void call(String msg){
          System.out.print("[" + msg);
          try{
               Thread.sleep(500);
          }catch(InterruptedException e){
               System.out.println("Inturrepted");
          }
          System.out.println("]");
     }
}

class Caller implements Runnable{
     String msg;
     Callme target;
     Thread t;

     Caller(Callme trg, String s){
          target = trg;
          msg = s;
          t = new  Thread(this);
          t.start();
     }

     public void run(){
          synchronized(target){
               target.call(msg);
```

```
            }
        }
}

class Th7{
        public static void main(String[] args){
                Callme target = new Callme();
                Caller ob1 = new Caller(target,"Hello");
                Caller ob2 = new Caller(target,"Synchronized");
                Caller ob3 = new Caller(target,"World");

                try{
                        ob1.t.join();
                        ob2.t.join();
                        ob3.t.join();
                }catch(InterruptedException e){
                        System.out.println("Interrupted");
                }
        }
}
```

Output:



```
suvam@Codebox ~/Thread $ javac Th7.java
suvam@Codebox ~/Thread $ java Th7
[Hello]
[World]
[Synchronized]
suvam@Codebox ~/Thread $ 
```