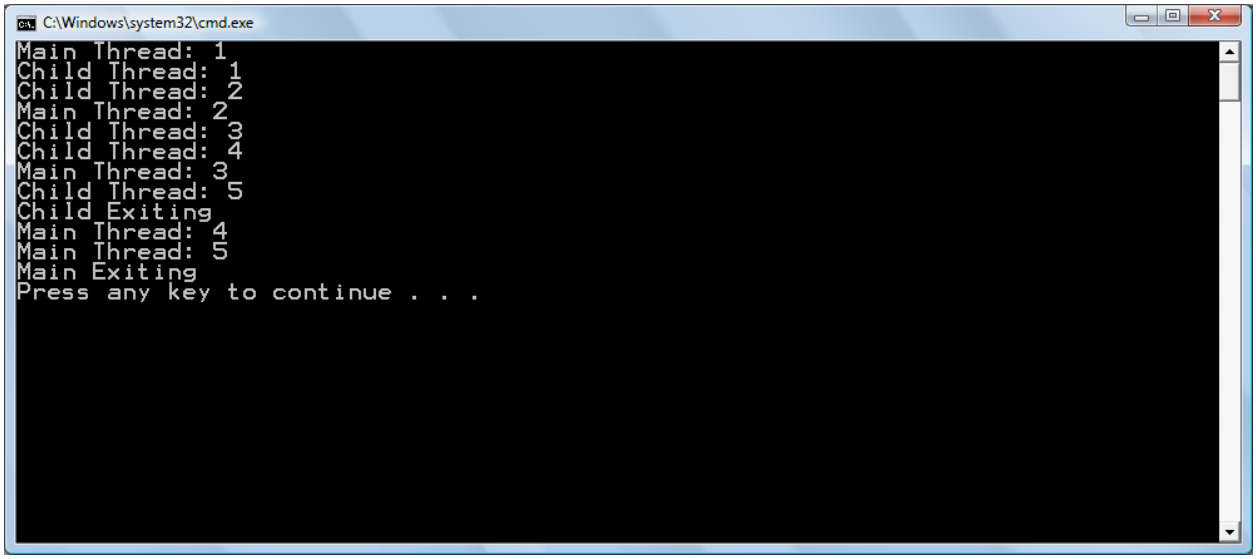
<u>Example # 01</u> (How can you create a simple thread?)

MyThreads.java

```
//Creating your own thread, You can create thread either by implementing the Runnable interface
//and defining the run() method (preferred approach) Or by extending the Thread class
//class implements the Runnable interface

class NewThread implements Runnable
{
 //defining the run method, (entry point for the thread)
 public void run()
 {
 try
 {
    //displaying the loop number with delay of half second
    for(int i=1; i<=5; i++)
    {
    System.out.println("Child Thread: "+ i);
    Thread.sleep(500);
    }
 }
 catch(InterruptedException e)
 {
    System.out.println("Child thread interrupted " + e.getMessage());
 }

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

}

}
class MyThreads
{
 public static void main(String []args)
 {
    //creating object of NewThread class
    NewThread ct = new NewThread();

    //creating new thread with the help of Thread class
    Thread t1 = new Thread(ct);

    //starting the thread, it will call the run() mehtod of NewThread class
    t1.start();

 try
 {
    //displaying the loop number with delay of 1 second
    for(int i=1; i<=5; i++)
    {
    System.out.println("Main Thread: "+ i);
    Thread.sleep(1000);
    }
 }

 catch(InterruptedException e)
 {
 System.out.println("Main thread interrupted " + e.getMessage());
 }
    System.out.println("Main Exiting");
 }

}
```

Code Example of Multi Threading

Output:



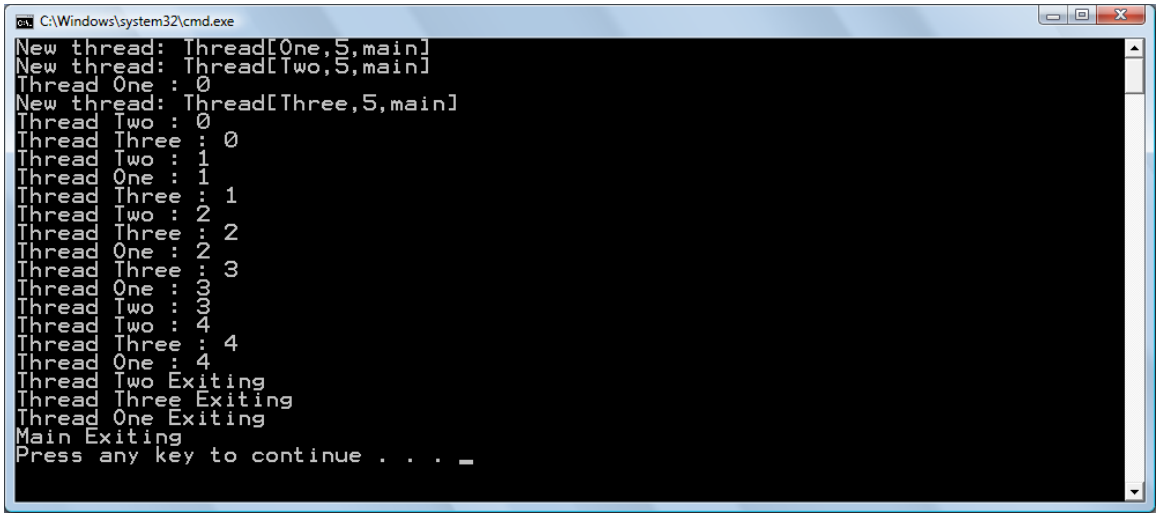
<u>Example # 02</u> (Create multiple threads…..?)

MultipleThreads.java

```
//Creating your multiple threads, You main thread must be the last thread to finish
//In this program we will handle it by using the sleep function of Thread class (Bad Approach)

class NewThread implements Runnable
 {
    String t_name;     //holds the thread name
    Thread t;      //holds the threa information

    //parameterized constructor
    NewThread(String t_name)
    {
     this.t_name = t_name; //intializing thread name
     //creating thread by calling the constructor of Thread class with thread name
     t = new Thread(this, t_name);
     System.out.println("New thread: " + t); //displaying thread information
     t.start();
    }

    //defining the run method, (entry point for the thread)
    public void run()
    {
    try
     {
      //displaying the loop number with delay of half second
      for(int i=0; i<5; i++)
       {
        System.out.println("Thread " + t_name + " : " + i);
        Thread.sleep(500);
       }
     }
    catch(InterruptedException e)
     {
      System.out.println("Child thread interrupted " + e.getMessage());
      }
     System.out.println("Thread "+ t_name + " Existing");
     }
 }

 class MultipleThreads
 {

  public static void main(String args[])
  {
   //creating object(s) of NewThread class
   NewThread t1 = new NewThread("One");
   NewThread t2 = new NewThread("Two");
   NewThread t3 = new NewThread("Three");

      try
      {
         Thread.sleep(10000);
      }
   catch(InterruptedException e)
   {
      System.out.println("Main interrupted " + e.getMessage());
   }
  System.out.println("Main Exiting");

   }
  }
```

Code Example of Multi Threading

Output:

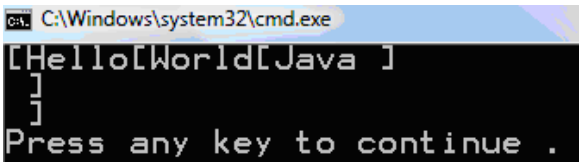


<u>Example # 03</u> (Multiple threads share the same piece of code without synchronization)

WithOutSync.java

```
//Un-Synchronized code, will mix up the out-put of the program
class Message
{
  void Go(String msg)
    {
     System.out.print("[" + msg);
    try
      {
       Thread.sleep(1000);
      }
    catch(InterruptedException e)
      {
        System.out.println("Interrupted " + e.getMessage());
      }
    System.out.println(" ]");
    }
}

class NewThread implements Runnable
{
 Thread t; //holds the thread information
 Message obj_m; //holds the Message class object
 String msg; //holds the message string


   //parameterized constructor
    NewThread(Message obj_m, String msg)
    {
     this.obj_m = obj_m; //initializing Message object
     this.msg = msg; //initializing the message
     //creating thread by calling the constructor of Thread class with thread name
     t = new Thread(this, msg);
     t.start(); //starting the thread
     }

    public void run()
    {
     obj_m.Go(msg); //calls the Go(String msg) method o fMessage class, with messag
    }
}


class WithOutSync
{

public static void main(String []args)
{
 //creating object of Message class
 Message obj_msg = new Message();

 //creating threads
 NewThread t1 = new NewThread(obj_msg, "Hello");
 NewThread t2 = new NewThread(obj_msg, "World");
 NewThread t3 = new NewThread(obj_msg, "Java");

 }
}
```

<u>Output:</u>

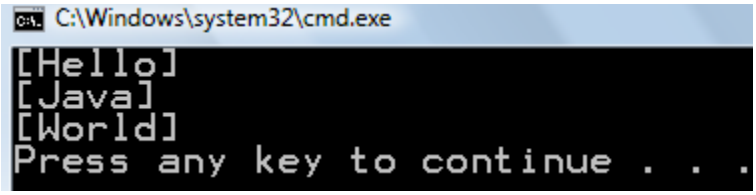


<u>Example # 04</u> (Multiple threads share the same piece of code with synchronization)

MethodSync.java

```
class Message
{
    synchronized void Go(String msg)
    {
     System.out.print("[" + msg);
    try
     {
      Thread.sleep(1000);
     }
    catch(InterruptedException e)
     {
      System.out.println("Interrupted " + e.getMessage());
     }
    System.out.println("]");
    }
}

class NewThread implements Runnable
 {
 Thread t; //holds the thread information
 Message obj_m; //holds the Message class object
 String msg; //holds the message string

 //parameterized constructor
    NewThread(Message obj_m, String msg)
    {
     this.obj_m = obj_m; //initializing Message object
     this.msg = msg; //initializing the message
     t = new Thread(this);
    t.start(); //starting the thread

    }

    public void run()
    {
     obj_m.Go(msg); //calls the Go(String msg) method
    }
}

class MethodSync
{

 public static void main(String []args)
 {
 //creating object of Message class
 Message obj_msg = new Message();

 //creating threads
 NewThread t1 = new NewThread(obj_msg, "Hello");
 NewThread t2 = new NewThread(obj_msg, "World");
 NewThread t3 = new NewThread(obj_msg, "Java");
 }
}
```

<u>Output:</u>

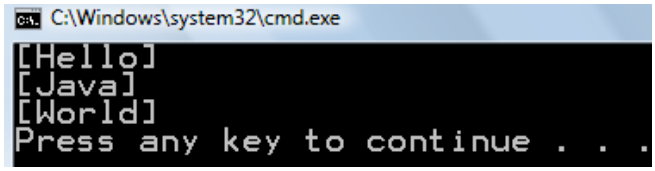


<u>Example # 05</u> (How can you synchronize a block instead of a method?)

BlockSync.java

```
//Synchronized code block synchronization provides a way to force the
//thread to not to enter in the shared code until other thread finishes its working
class Message
{

    void Go(String msg)
    {
     System.out.print("[" + msg);
     try
     {
      Thread.sleep(1000);
     }
     catch(InterruptedException e)
     {
      System.out.println("Interrupted " + e.getMessage());
     }
    System.out.println("]");
    }
}


//class implements the Runnable interface
 class NewThread implements Runnable
 {
  Thread t; //holds the thread information
  Message obj_m; //holds the Message class object
  String msg; //holds the message string

 //parameterized constructor

    NewThread(Message obj_m, String msg)
    {
     this.obj_m = obj_m; //initializing Message object
     this.msg = msg; //initializing the message
     t = new Thread(this);
     t.start(); //starting the thread
    }

 //defining the run method, (entry point for the thread)

     public void run()
    {
     //synchronizing the block for Message object
     synchronized(obj_m)
        {
        obj_m.Go(msg);
        }
    }
}


class BlockSync
{

 public static void main(String []args)
 {
 //creating object of Message class
 Message obj_msg = new Message();

 //creating threads
 NewThread t1 = new NewThread(obj_msg, "Hello");
 NewThread t2 = new NewThread(obj_msg, "World");
 NewThread t3 = new NewThread(obj_msg, "Java");
 }
}
```

<u>Output:</u>

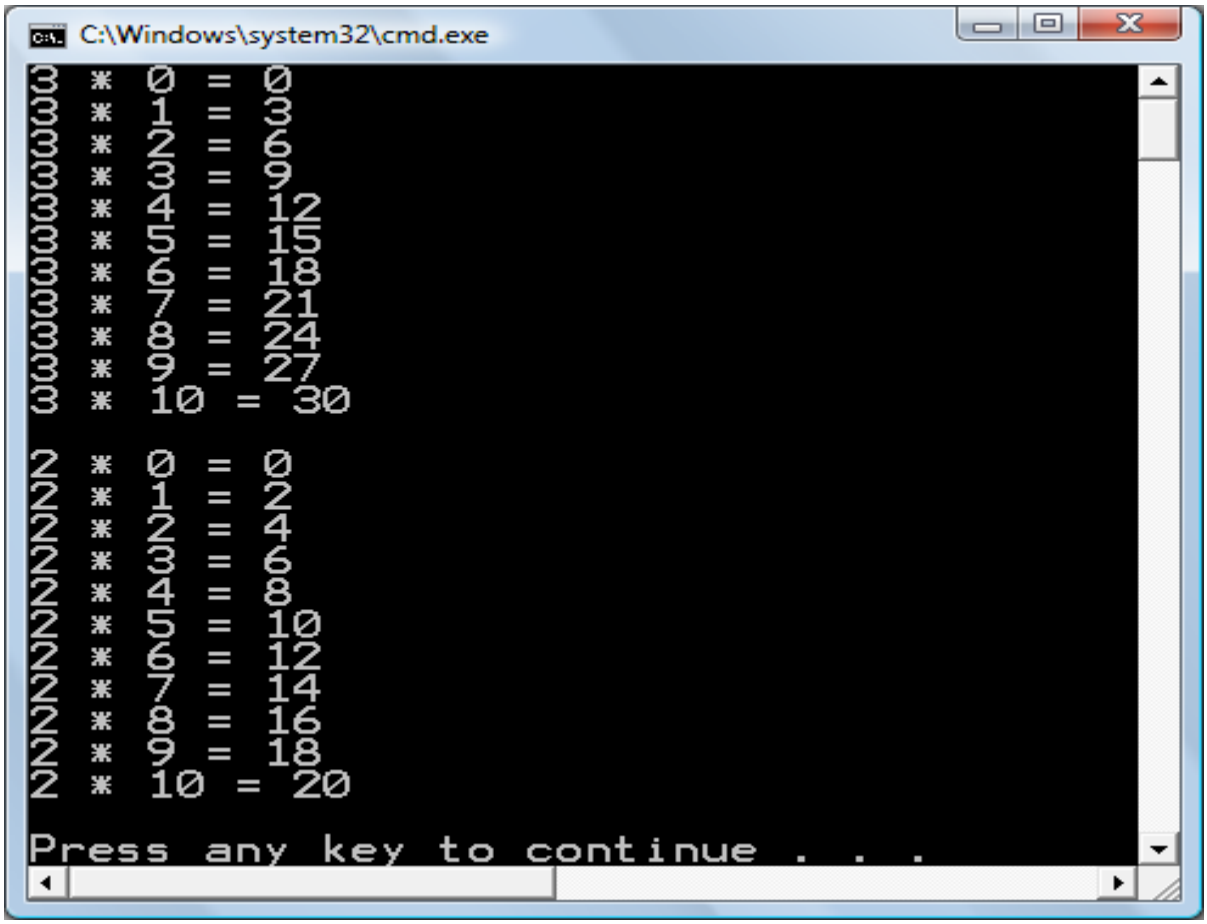


Example # 06

TableSync.java

```
class Table
{
 synchronized void GenerateTable(int table, int s_limit, int e_limit)
 {
  for(int i=s_limit; i<=e_limit; i++)
  {
   System.out.println(table + " * " + i + " = " + (i * table));
  }
 System.out.println();
 }
}

class NewThread implements Runnable
 {
 Thread t;
 Table obj_t;
 int table;
 int s_limit;
 int e_limit;

 //parameterized constructor
    NewThread(Table obj_t, int table, int s_limit, int e_limit)
    {
     this.obj_t = obj_t;
     this.table = table;
     this.s_limit = s_limit;
     this.e_limit = e_limit;
     t = new Thread(this);
     t.start();
    }
 //defining the run method, (entry point for the thread)
    public void run()
    {
     obj_t.GenerateTable(table, s_limit, e_limit);
    }
}

class TableSync
{
    public static void main(String []args)
    {
     //creating object of Table class
    Table obj_t = new Table();

    //creating threads
    NewThread t1 = new NewThread(obj_t, 2, 0, 10);
    NewThread t2 = new NewThread(obj_t, 3, 0, 10);
 }
}
```

Output:


C:\Windows\system32\cmd.exe
3 * 0 = 0
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
2 * 0 = 0
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
Press any key to continue . . .

<u>Example # 07</u> (Use of *Join()*)

UsingJoin.java

```
//Creating your multiple threads, You main thread must be the last thread to finish.
//A better approach to do this is the use of join() mehtod of Thread class
//This method waits until the thread on which it is called terminates.
class NewThread implements Runnable
{
 //data members
 String t_name; //holds the thread name
 Thread t; //holds the threa information

 //parameterized constructor
    NewThread(String t_name)
     {
      this.t_name = t_name; //intializing thread name
      t = new Thread(this, t_name);
      System.out.println("New thread: " + t); //displaying thread information
      t.start(); //starting the thread
     }

 //defining the run method, (entry point for the thread)
    public void run()
    {
     try
       {
        //displaying the loop number with delay of half second
        for(int i=1; i<=5; i++)
        {
         System.out.println("Thread " + t_name + " : " + i);
         Thread.sleep(500);
        }
       }
       catch(InterruptedException e)
       {
        System.out.println("Child thread interrupted " + e.getMessage());
       }
    System.out.println("Thread "+ t_name + " Existing");
    }
}

class UsingJoin
{

public static void main(String []args)
{
 //creating object(s) of NewThread class
 NewThread t1 = new NewThread("One");
 NewThread t2 = new NewThread("Two");
 NewThread t3 = new NewThread("Three");

 //Checking either the threads are alive
 System.out.println("Thread " + t1.t_name + " is alive: " +  t1.t.isAlive());
 System.out.println("Thread " + t2.t_name + " is alive: " +  t2.t.isAlive());
 System.out.println("Thread " + t3.t_name + " is alive: " +  t3.t.isAlive());

 //wait for threads to finish
try
{
 t1.t.join();
 t2.t.join();
 t3.t.join();
}
catch(InterruptedException e)
{
 System.out.println("Interrupted " + e.getMessage());
}

 //Checking either the threads are alive
 System.out.println("Thread " + t1.t_name + " is alive: " +  t1.t.isAlive());
 System.out.println("Thread " + t2.t_name + " is alive: " +  t2.t.isAlive());
 System.out.println("Thread " + t3.t_name + " is alive: " +  t3.t.isAlive());
System.out.println("Main Exiting");

 }
}
```

Code Example of Multi Threading

Output: