Multithreading Applications

```
using System;
using System.Threading;
```

Application 1:
Multithreaded application with Static Thread procedure.

```
public class SingleTask1
{
    public static void DisplayF()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
    }
}


public class Marvellous
{
    public static void Main()
    {
       Console.WriteLine("Multiple threads as a static method performing same task");
        Thread t1 = new Thread(new ThreadStart(SingleTask1.DisplayF));
        Thread t2 = new Thread(new ThreadStart(SingleTask1.DisplayF));

        t1.Name = "Marvellous First";
        t2.Name = "Marvellous Second";

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

Application 2:

Multithreaded application with Non Static Thread procedure.

```
public class SingleTask2
{
    public void DisplayF()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
          Console.WriteLine("Multiple threads as a non static method performing same task");

        SingleTask2 S1 = new SingleTask2();

        Thread t3 = new Thread(new ThreadStart(S1.DisplayF));
        Thread t4 = new Thread(new ThreadStart(S1.DisplayF));

        t3.Name = "Marvellous Third";
        t4.Name = "Marvellous Fourth";

        t3.Start();
        t4.Start();
    }
}
```

Application 3:

Multithreaded application with Non Static Thread procedure which performs different task.

```
public class MultipleTask
{
    public void DisplayF()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
    }

    public void DisplayB()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 10; i >= 0; i--)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
         Console.WriteLine("Multiple threads as a non static method performing different
task");
        MultipleTask Mobj = new MultipleTask();

        Thread t5 = new Thread(new ThreadStart(Mobj.DisplayF));
        Thread t6 = new Thread(new ThreadStart(Mobj.DisplayB));
        t5.Name = “Marvellous Fifth";
        t6.Name = "Marvellous Sixth";
    }
}
```

Application 4:

Multithreaded application with Non Static Thread procedure. Main thread abort the execution of child thread. (Abort() method)

```
public class ThreadSleep
{
    public void Display()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);

            Thread.Sleep(1000);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
       ThreadSleep tsobj = new ThreadSleep();
        Thread t7 = new Thread(new ThreadStart(tsobj.Display));
        t7.Name = "Seventh";

        t7.Start();

        try
        {
            t7.Abort();
        }
        catch (ThreadAbortException tae)
        {
            Console.WriteLine(tae.ToString());
        }
    }
}
```

Application 5:

Multithreaded application with Non Static Thread procedure. Main thread waits till the execution of child thread gets completed. (Join() method)

```
public class ThreadSleep
{
    public void Display()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);

            Thread.Sleep(1000);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
        ThreadSleep tsobj1 = new ThreadSleep();
        Thread t8 = new Thread(new ThreadStart(tsobj1.Display));

        t8.Name = "Marvellous Eight";

        t8.Start();
        t8.Join();
        Console.WriteLine("Continue from main thread..");
    }
}
```

Application 6:

Multithreaded application with Non Static Thread procedure. In this application we change default priority of thread.

```
public class ThreadSleep
{
    public void Display()
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);

            Thread.Sleep(1000);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
        ThreadSleep tsobj2 = new ThreadSleep();
        Thread t9 = new Thread(new ThreadStart(tsobj2.Display));
        Thread t10 = new Thread(new ThreadStart(tsobj2.Display));

        t9.Name = "Marvellous Ninth";
        t10.Name = "Marvellous Tenth";

        t9.Priority = ThreadPriority.Lowest;
        t10.Priority = ThreadPriority.Highest;

        t9.Start();
        t10.Start();
    }
}
```

Application 7:

Multithreaded application which accept parameter from main thread using ParameterizedThreadStart.

```
using System;
using System.Threading;

public class SingleTask2
{
    public void DisplayF(Object no)
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < (int)no; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
           Console.WriteLine("Threads which accept parameter using
ParameterizedThreadStart");

                SingleTask2 S1 = new SingleTask2();

                Thread t3 = new Thread(new ParameterizedThreadStart(S1.DisplayF));
                Thread t4 = new Thread(new ParameterizedThreadStart(S1.DisplayF));

                t3.Name = "Marvellous Third";
                t4.Name = "Marvellous Fourth";

                t3.Start(11);
                t4.Start(21);
    }
}
```

Application 8:

Multithreaded application which accept parameter from main thread using Lambda Expression.

```
using System;
using System.Threading;

public class SingleTask2
{
    public void DisplayF(int no)
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < no; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
    }
}

public class Marvellous
{
    public static void Main()
    {
            Console.WriteLine("Threads which accept parameter using Lambda
Expression");

            SingleTask2 S1 = new SingleTask2();

            Thread t = new Thread (() => S1.DisplayF(11));

            t.Name = “Marvellous”;

            t.Start ();
    }
}
```

Application 9:

Multithreaded application which returns value from thread procedure using Lambda Expression.

```
using System;
using System.Threading;

public class SingleTask2
{
    public int DisplayF(int no)
    {
        Thread t = Thread.CurrentThread;

        for (int i = 0; i < no; i++)
        {
            Console.WriteLine(t.Name + " is running");

            Console.WriteLine(i);
        }
                return 1;
    }
}

public class Marvellous
{
    public static void Main()
    {
           Console.WriteLine("Thread which returns the value from thread procedure using
Lambda Expression");

                SingleTask2 S1 = new SingleTask2();
                int value = 0;

                Thread t = new Thread(() => { value = S1.DisplayF(5);});
                t.Name = "Marvellous";

                t.Start();
                t.Join();

                Console.WriteLine("Return value from thread is {0}",value);
    }
}
```