Threads

By

Dr M. Senthilkumar

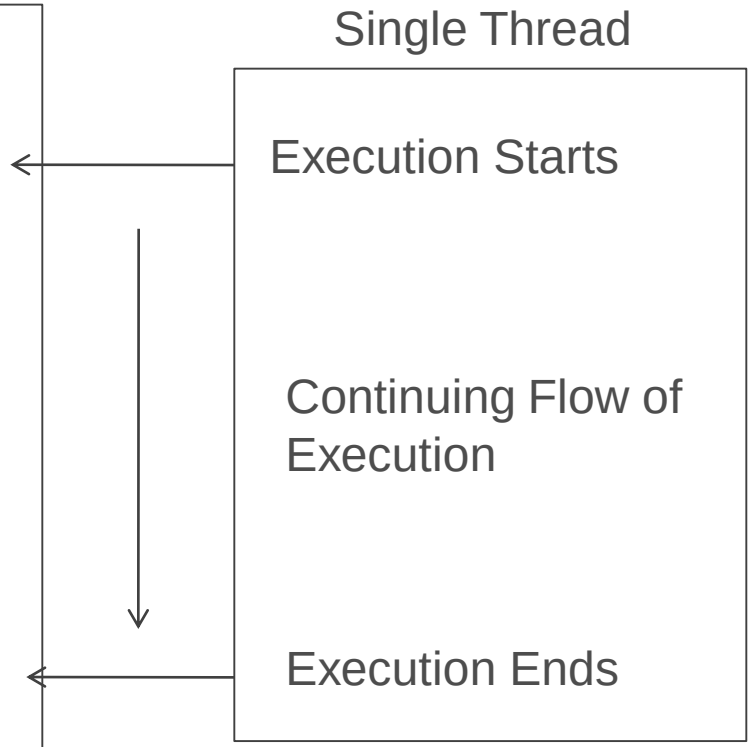
Assistant Professor

Department of Computer Science

Government Arts and Science College, Avinashi - 641654

Threads

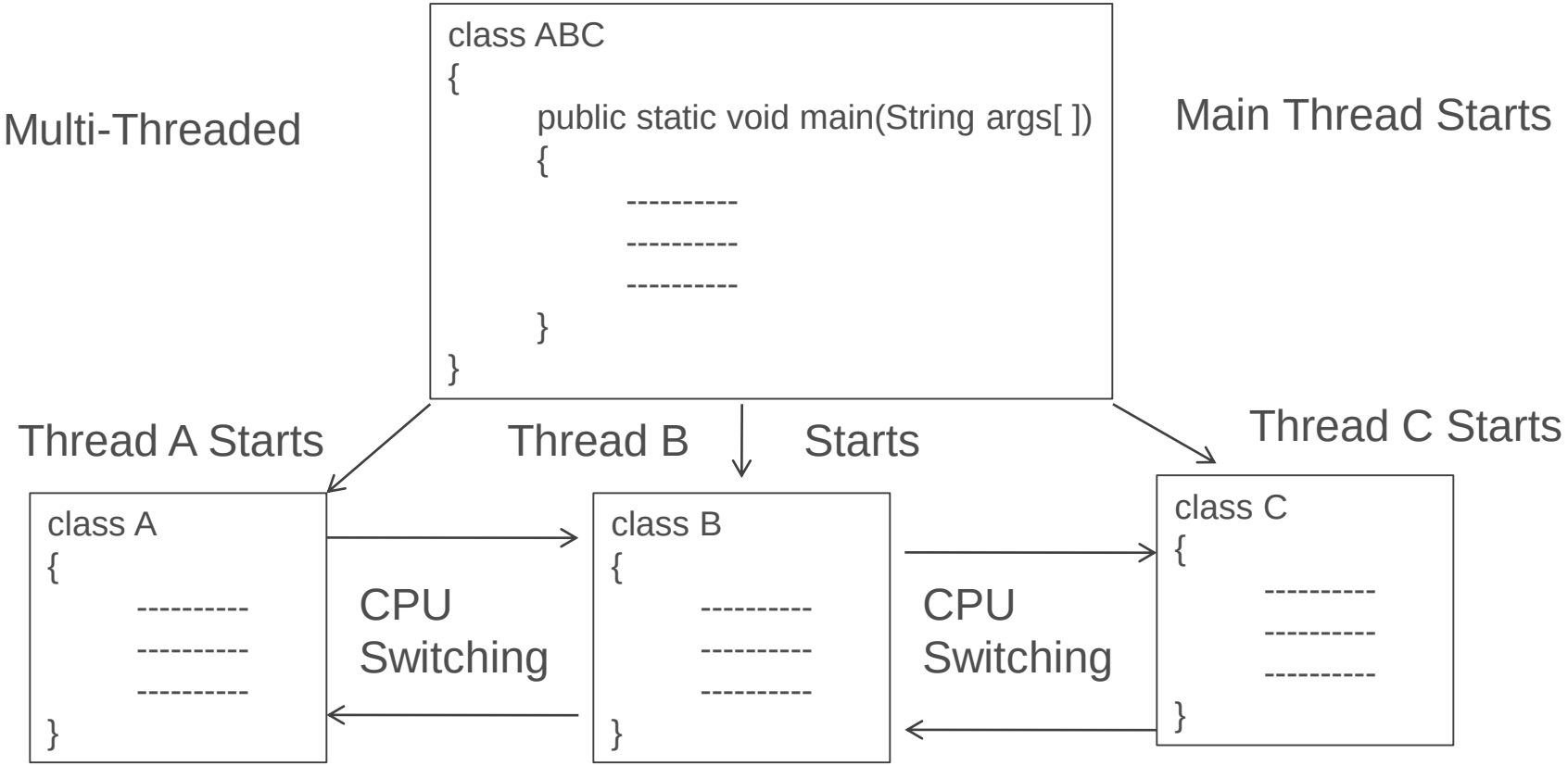
```
class ABC
{
        public static void main(String args[ ])
        {
                Statement – 1;
                Statement – 2;
                Statement – 3;
                Statement – 4;
                ----------------
                ----------------
                Statement – n;
        }
}
```



Threads

- ✓ A Java program is executed in a particular sequence. The program begins, runs through a sequence of executions, and finally ends.
- ✓ At a given point of time, only one statement is under execution
- ✓ So, a Single flow of control is required to execute a Java program
- ✓ Thread refers to a flow of control for a Program

Threads


Multi-Threaded
class ABC
{
public static void main(String args[ ])
{

}
}
Main Thread Starts
Thread A Starts
Thread B Starts
Thread C Starts
class A
{

}
CPU Switching
class B
{

}
CPU Switching
class C
{

}

Threads

- ✓ A Java program can be divided into sub-programs.
- ✓ Each sub program must have a separate sequence of execution i.e., a Separate flow control is required
- ✓ Each sub-program must be independently executed
- ✓ All sub-programs can be parallely executed
- ✓ So Multiple flow of controls are found i.e., Multiple threads are required

Threads

- ✓ Java program contains either a single flow of control or multiple flow of control for execution
- ✓ If a Java program contains single flow of control, then it is known as Single Threaded Program
- ✓ If a Java program contains multiple flows of control then it is known as Multithreaded Program

Creating Threads – run( ) method

- ✓ Threads are created in the form of Objects
- ✓ Thread contains a method called run( )

```
public void run( )
{
        Statements for implementing Thread
}
```

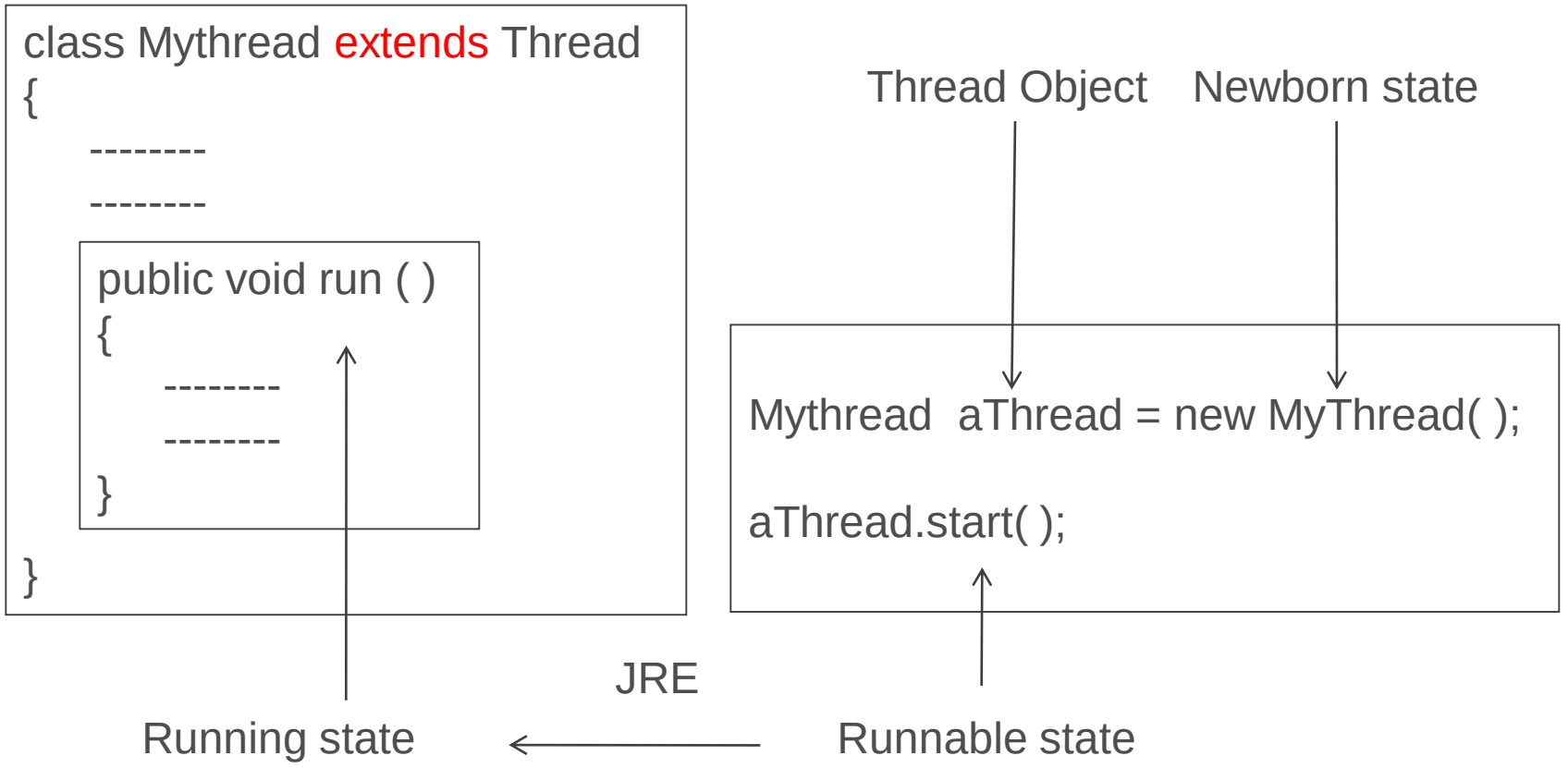
Creating Threads - run( ) method

- ✓ The run( ) method is the heart and soul of any thread
- ✓ The run( ) method makes the entire body of Thread
- ✓ The run( ) method is the only method that implements Thread’s behavior
- ✓ The run( ) method is invoked by an Object of Thread

Creating Threads - start( ) method

- ✓ Thread is created. It is initiated using start( ) method
- ✓ The start( ) is also a method of Thread
- ✓ Thread can be created using Two methods
- ✓ Extending Thread Class and overriding the run( )
- ✓ Implementing Runnable Interface which has run( )

Extending Thread Class - overriding the run( )



Extending Thread Class - overriding the run( )

- ✓ Declare a Class which extends java.lang.Thread class
- ✓ Implement the run( )
- ✓ Create Thread object and call start( ) to initiate Thread

Extending Thread Class – Example 1

```
class A extends Thread
{
      public void run ( )
      {
        for ( int i = 1; i <= 5; i++ )
        {
            System.out.println("From Thread A : i =  " + i);
        }
        System.out.println("Exit From Thread A");
      }
}
```

```
class B extends Thread
{
      public void run ( )
      {
        for ( int j = 1; j <= 5; j++ )
        {
           System.out.println("From Thread B : j =  " + j);
        }
        System.out.println("Exit From Thread B");
      }
}
```

Extending Thread Class – Example 1

```
class C extends Thread
{
      public void run ( )
      {
        for ( int k = 1; k <= 5; k++ )
        {
          System.out.println("From Thread C : k =  " + k);
        }
        System.out.println("Exit From Thread C");
      }
}
```

```
class ThreadTest
{
      public static void main(String args[ ])
      {
                new A( ).start( );
                new B( ).start( );
                new C( ).start( );
      }
}
```

D:\jdk1.8.0_111\jdk1.8.0_111\bin>javac ThreadTest.java

Extending Thread Class – Example 1

```
D:\jdk1.8.0_111\jdk1.8.0_111\bin>java ThreadTest
From Thread A : i =  1
From Thread A : i =  2
From Thread A : i =  3
From Thread A : i =  4
From Thread A : i =  5
Exit From Thread A
From Thread B : j =  1
From Thread B : j =  2
From Thread B : j =  3
From Thread B : j =  4
From Thread B : j =  5
Exit From Thread B
From Thread B : k =  1
From Thread B : k =  2
From Thread B : k =  3
From Thread B : k =  4
From Thread B : k =  5
Exit From Thread C
```

Run 1

```
D:\jdk1.8.0_111\jdk1.8.0_111\bin>java ThreadTest
From Thread A : i =  1
From Thread A : i =  2
From Thread A : i =  3
From Thread A : i =  4
From Thread A : i =  5
Exit From Thread A
From Thread B : j =  1
From Thread B : j =  2
From Thread B : j =  3
From Thread B : j =  4
From Thread B : j =  5
Exit From Thread B
From Thread B : k =  1
From Thread B : k =  2
From Thread B : k =  3
From Thread B : k =  4
From Thread B : k =  5
Exit From Thread C
```

Run 2

Extending Thread Class – Example 1

```
D:\jdk1.8.0_111\jdk1.8.0_111\bin>java ThreadTest
From Thread A : i =  1
From Thread A : i =  2
From Thread A : i =  3
From Thread A : i =  4
From Thread A : i =  5
Exit From Thread A
From Thread B : j =  1
From Thread B : j =  2
From Thread B : j =  3
From Thread B : j =  4
From Thread B : j =  5
Exit From Thread B
From Thread B : k =  1
From Thread B : k =  2
From Thread B : k =  3
From Thread B : k =  4
From Thread B : k =  5
Exit From Thread C
```

Run 3

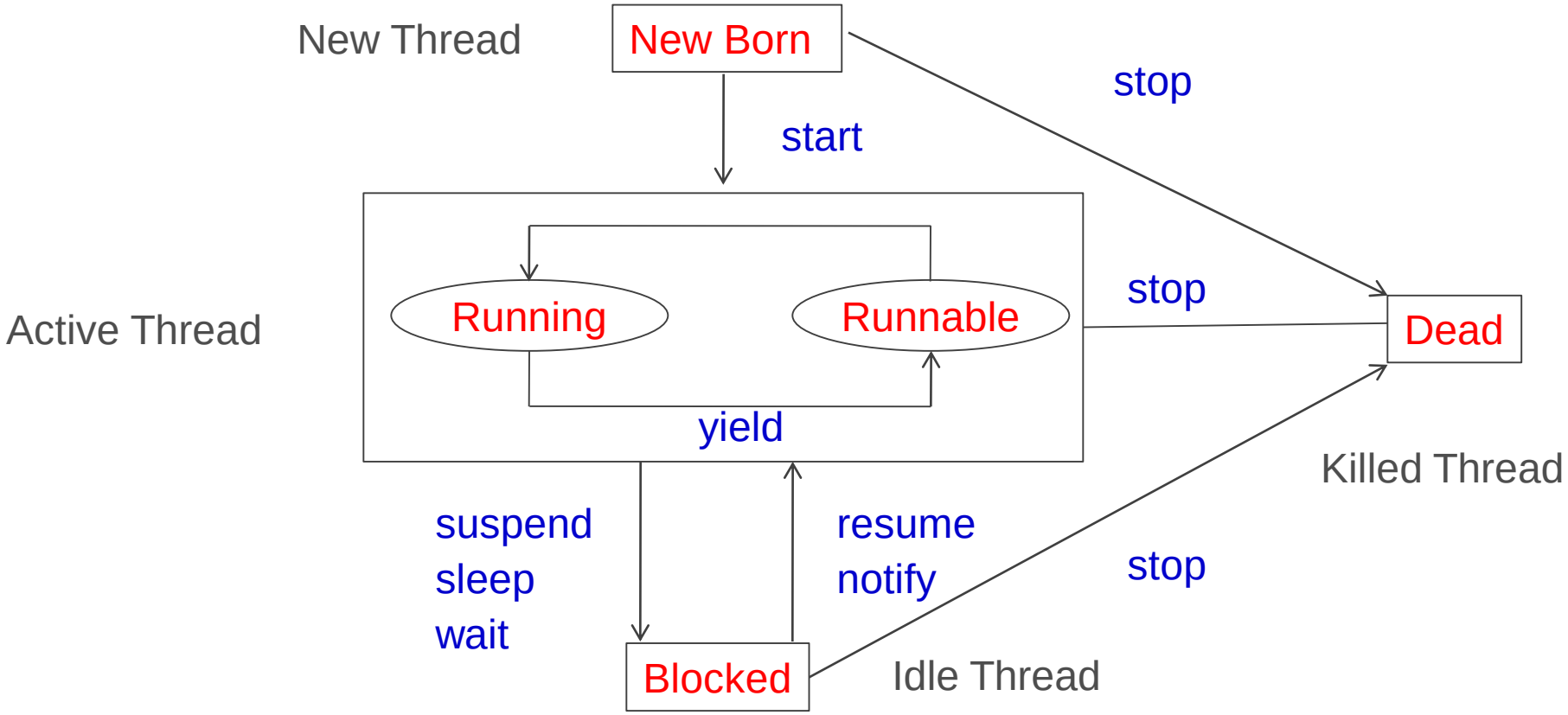
```
D:\jdk1.8.0_111\jdk1.8.0_111\bin>java ThreadTest
From Thread A : i =  1
From Thread A : i =  2
From Thread A : i =  3
From Thread B : j =  1
From Thread B : j =  2
From Thread B : j =  3
From Thread B : j =  4
From Thread B : j =  5
Exit From Thread B
From Thread A : i =  4
From Thread A : i =  5
Exit From Thread A
From Thread B : k =  1
From Thread B : k =  2
From Thread B : k =  3
From Thread B : k =  4
From Thread B : k =  5
Exit From Thread C
```

Run 4

Threads – stop( ), sleep( ), suspend( ), wait( )

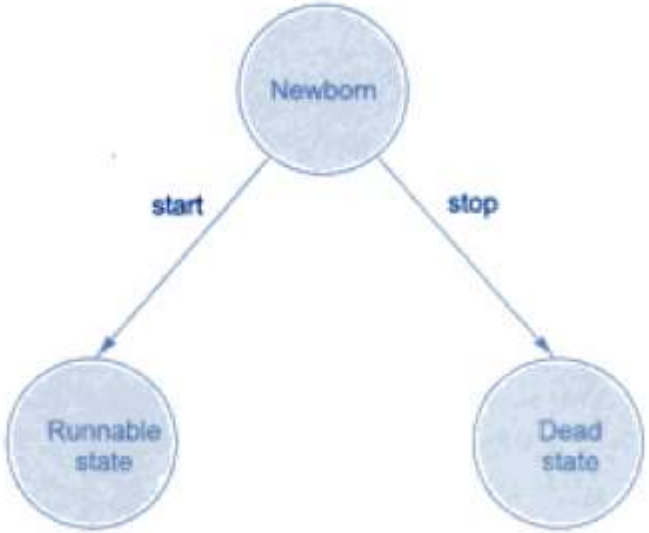
aThread.stop( );	Moving a Thread to Dead state
aThread.sleep( );	Block a Thread to Specified Time
aThread.suspend( );	Block a Thread until further Orders
aThread.wait( );	Block a Thread until a Condition occurs

Life Cycle of a Thread



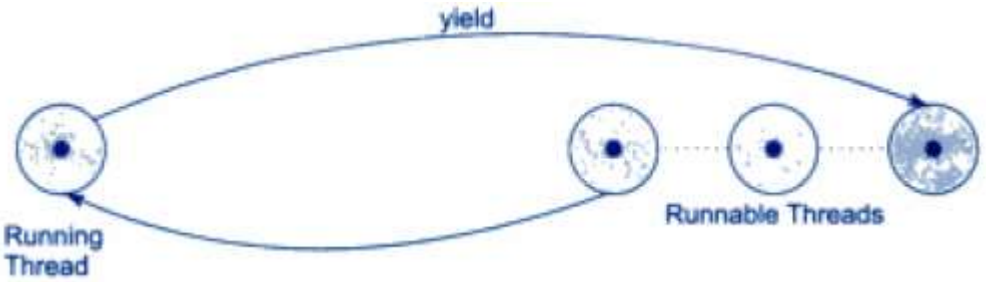
Life Cycle of a Thread – New Born State

- ✓ New Born State: Thread is created
- ✓ Thread is invoked using
  - ✓ Thread.start( );
  - ✓ Moving a Thread to Runnable state
- ✓ Thread Object is killed using
  - ✓ Thread.stop( );
  - ✓ Moving a Thread to Dead state



Life Cycle of a Thread – Runnable State

- ✓ Runnable State:
  - ✓ Thread is ready for execution
  - ✓ Thread is waiting in Queue to get CPU

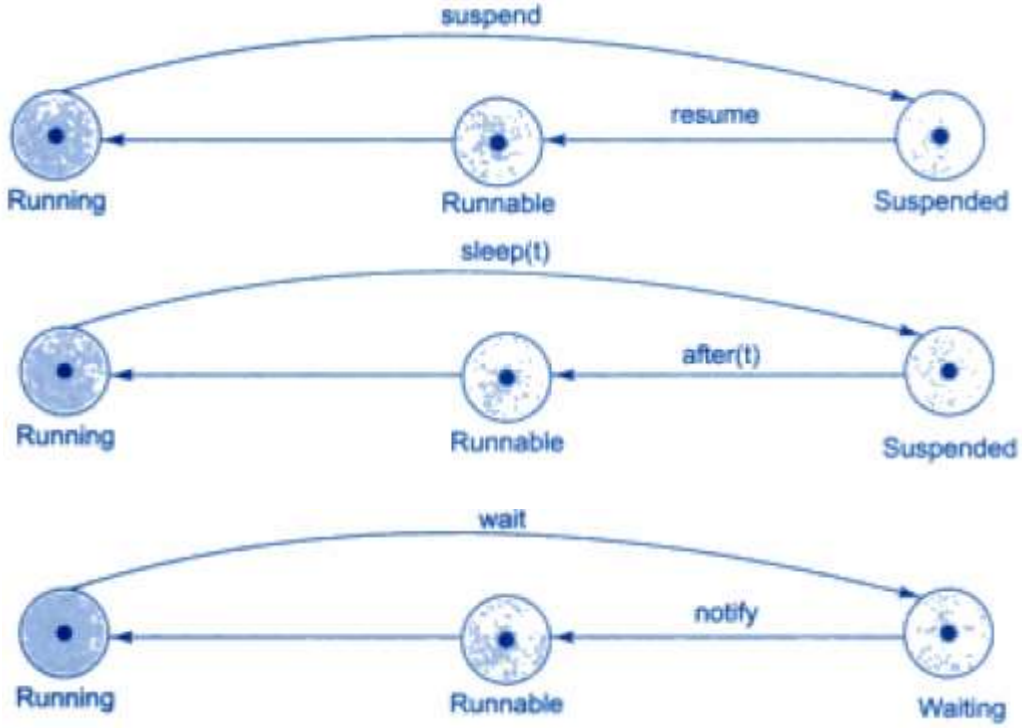


Life Cycle of a Thread – Runnable State

- ✓ Runnable State:
  - ✓ Round Robin: Equal Time Slots for Threads with same Priorities
  - ✓ Thread.yield( ): Relinquish Control from Running Thread and move it to Runnable State
  - ✓ Threads may be assigned Priorities
  - ✓ Threads are arranged based on Priorities

Life Cycle of a Thread – Running State

- ✓ Running State: Thread is executed in CPU



Life Cycle of a Thread – Running State

Thread.stop( );	Moving a Thread to Dead state
Thread.suspend( );	Block a Thread until further Orders Resumed
Thread.sleep( );	Block a Thread to Specified Time (ms)
Thread.wait( );	Block a Thread until a Condition occurs Notified

Life Cycle of a Thread – Blocked State

- ✓ Blocked State: To satisfy the requirements
  - ✓ Thread is suspended
  - ✓ Thread is sleeping
  - ✓ Thread is waiting

Life Cycle of a Thread – Dead State

- ✓ Dead State:
  - ✓ Life of the Thread Ends
  - ✓ Thread Completes its execution
  - ✓ Thread is killed after
    - ✓ New born
    - ✓ Running/ Runnable
    - ✓ Blocked

Thread Methods – Example 2

```
class A extends Thread
{
      public void run ( )
      {
        for ( int i = 1; i <= 5; i++ )
        {
            if ( i == 1) yield( );
            System.out.println("From Thread A : i =  " + i);
        }
        System.out.println("Exit From Thread A");
      }
}
```

```
class B extends Thread
{
      public void run ( )
      {
        for ( int j = 1; j <= 5; j++ )
        {
            System.out.println("From Thread B : j =  " + j);
            if ( j == 3 ) stop( );
        }
        System.out.println("Exit From Thread B");
      }
}
```

Thread Methods – Example 2

```
class C extends Thread
{
      public void run ( )
      {
        for ( int k = 1; k <= 5; k++ )
        {
          System.out.println("From Thread C : k =  " + k);
          if ( k == 1 )
                try
                {
                    sleep(1000);
                }
                catch( Exception  e)   {          }

        }
        System.out.println("Exit From Thread C");
      }
}
```

```
class ThreadTest2
{
  public static void main(String args[ ])
  {
    A  threadA   = new A( );
    B  threadB  = new B( );
    C  threadC  = new C( );
    System.out.println(" Start Thread A  ");
    threadA.start( );
    System.out.println(" Start Thread B  ");
    threadB.start( );
    System.out.println(" Start Thread C  ");
    threadC.start( );
    System.out.println(" End of Main Thread  ");
  }
}
```

Thread Methods – Example 2

```
Start thread A
Start thread B
Start thread C
    From Thread B : j = 1
    From Thread B : j = 2
    From Thread A : i = 1
    From Thread A : i = 2
End of main thread
    From Thread C : k = 1
    From Thread B : j = 3
    From Thread A : i = 3
    From Thread A : i = 4
    From Thread A : i = 5
Exit from A
    From Thread C : k = 2
    From Thread C : k = 3
    From Thread C : k = 4
    From Thread C : k = 5
Exit from C
```

Thread Priorities

- ✓ setPriority(intNumber) – Sets priority for a Thread
- ✓ intNumber may take values from 1 to 10
- ✓ Thread class contains several priority constants

  MIN_PRIORITY = 1
  NORM_PRIORITY = 5
  MAX_PRIORITY = 10

- ✓ NORM_PRIORITY is default – Many user level processes uses it with +/- 1)

Thread Priorities – Example 3

```
import java.io.*;
class A extends Thread
{
   public void run()
   {
      for(int i=1;i<=5;i++)
      {
        System.out.println(i + "*" +3+ "=" +(i*3));
       }
       System.out.println("End of the 1st Thread");
    }
}
```

```
class B extends Thread
{
   public void run()
   {
      for(int j=1;j<=5;j++)
      {
        System.out.println(j + "*" +5+ "=" +(j*5));
       }
       System.out.println("End of the 2nd Thread");
    }
}
```

Thread Priorities – Example 3

```
class C extends Thread
{
   public void run()
   {
      for(int k=1;k<=5;k++)
      {
        System.out.println(k + "*" +7+ "=" +(k*7));
       }
       System.out.println("End of the 3rd Thread");
    }
}
```

Thread Priorities – Example 3

```
public class Multithread
{
	public static void main(String args[ ])throws IOException
	{
		A ThreadA=new A();
		B ThreadB=new B();
		C ThreadC=new C();
		ThreadA.setPriority(Thread.NORM_PRIORITY);
		ThreadB.setPriority(Thread.MAX_PRIORITY);
		ThreadC.setPriority(Thread.MIN_PRIORITY);
		System.out.println("The priority of Thread A is "+ThreadA.getPriority());
		System.out.println("The priority of Thread B is "+ThreadB.getPriority());
		System.out.println("The priority of Thread C is "+ThreadC.getPriority());
		ThreadA.start();
		ThreadB.start();
		ThreadC.start();
	}
}
```

Thread Priorities – Example 3

```
D:\jdk1.8.0_111\jdk1.8.0_111\bin>java Multithread
The priority of Thread A is 5
The priority of Thread B is 10
The priority of Thread C is 1
1*5=5
2*5=10
3*5=15
4*5=20
5*5=25
End of the 2nd Thread
1*3=3
2*3=6
3*3=9
4*3=12
5*3=15
End of the 1st Thread
1*7=7
2*7=14
3*7=21
4*7=28
5*7=35
End of the 3rd Thread
```

Implementing Runnable Interface

- ✓ Runnable interface declares run( ) method
- ✓ Declare a class that implements Runnable interface
- ✓ Implement run( )
- ✓ Create a Thread by defining an object that is instantiated from this “runnable” class of as the target of the Thread
- ✓ Call the Thread’s start( )

Implementing Runnable Interface – Example 4

```
class X implements Runnable
{
    public void run( )
    {
        for(int i = 1;  i <= 10; i++)
        {
            System.out.println("\t ThreadX  :  " + i );
        }
         System.out.println("End of  ThreadX  " );
    }
}
```

Implementing Runnable Interface – Example 4

```
Class RunnableTest
{
    public static void main(String args[  ])
    {
            X runnable = new X( );
            Thread threadX  =  new Thread(runnable);
            threadX.start( );
            System.out.println("End of  main Thread" );
    }
}
```

Implementing Runnable Interface – Example 4

```
End of main Thread
     ThreadX : 1
     ThreadX : 2
     ThreadX : 3
     ThreadX : 4
     ThreadX : 5
     ThreadX : 6
     ThreadX : 7
     ThreadX : 8
     ThreadX : 9
     ThreadX : 10
End of ThreadX
```

References

- ✓ Programming with Java – A Primer - E. Balagurusamy, 3rd Edition, TMH

Thank You