12.5 Life Cycle of a Thread

A thread can be in one of the five states as shown in figure 12.2.

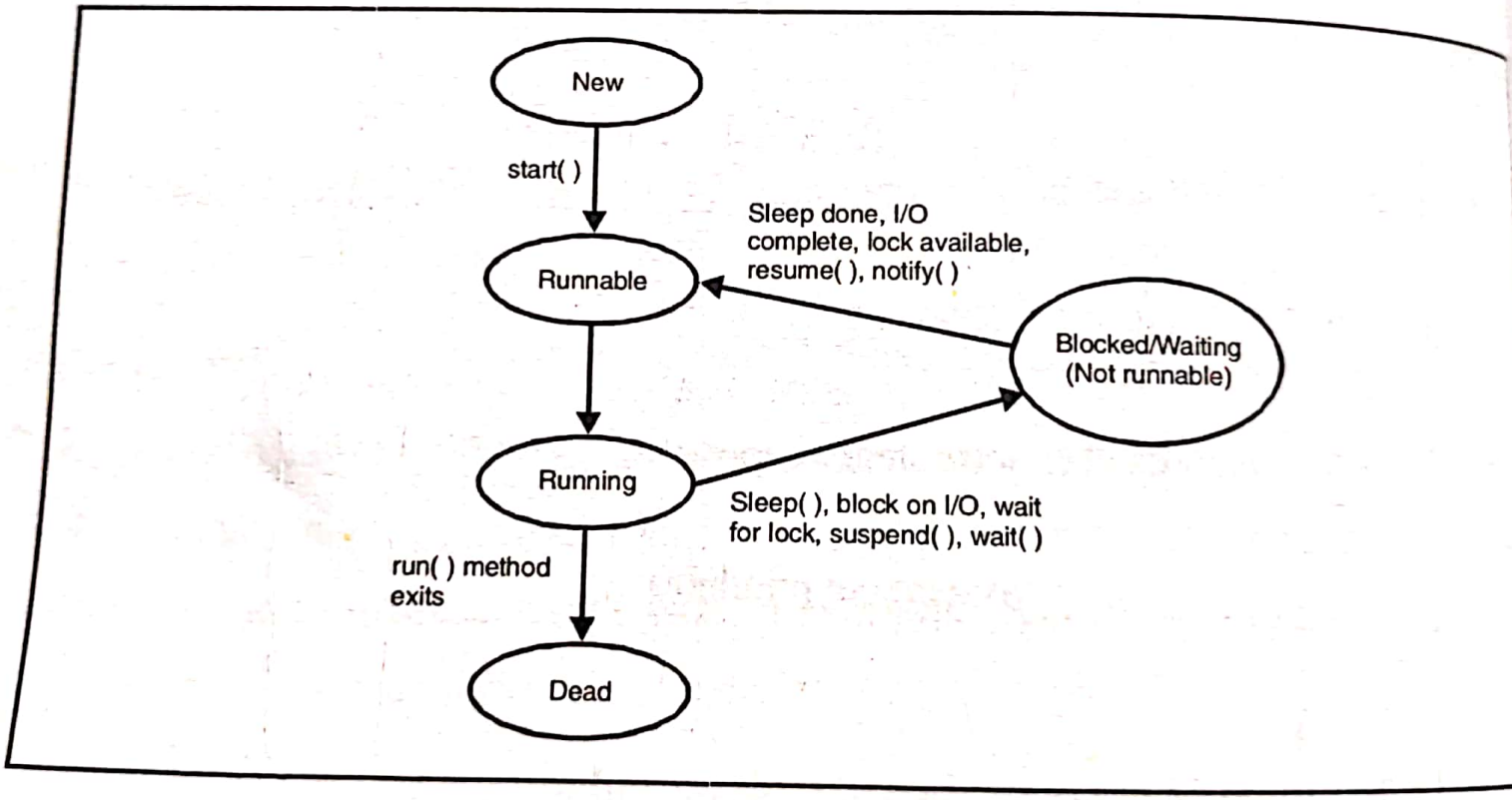

FIGURE 12.2

The life cycle of the thread in java is controlled by JVM. The java thread states are :

1. New
2. Runnable
3. Running
4. Blocked/Waiting
5. Dead (or Terminated)

1. New State

When we create a new Thread object using **new operator,** thread state in New Thread. At this point, thread is not alive. At this state, we can do only one of the following with it :

- Schedule it for running using start( ) method.
- Kill it using stop( ) method.

2. Runnable

The thread is in runnable state after invocation of start( ) method, but the thread scheduler has not selected it to be the running thread. Therefore, the runnable state means that the thread is ready for execution and is waiting for the availability of the processor.

3. Running

When thread is executing, it's state is changed to Running. Thread scheduler picks one of the thread from the runnable thread pool and change it's state to Running and CPU starts executing this thread. A running thread may relinquish (change) its control in one of the following situations :

(i) suspend( ) and resume( ) Methods

We can suspend a thread for some time due to certain reason, but do not want to kill it. The thread can be scheduled to run again using resume( ) method.

(ii) sleep( ) Method

We can put a thread to sleep for a specified time period using the sleep method. The thread re-enter the runnable state as soon as time period is elapsed.

(iii) wait( ) and notify( ) Methods

We can put a thread in wait( ) mode until some event occurs by using wait( ) method. The thread can be scheduled to run again by using notify( ) method.

4. **Blocked/Waiting**

A thread can be waiting for other thread to finish using thread join or it can be waiting for some resources to available, for example I/O resources, then it's state is changed to Waiting. Once the thread wait state is over, it's state is changed to Runnable and it's moved back to runnable thread pool.

5. **Dead**

Once the thread finished executing, it's state is changed to Dead and it is considered to be not alive.

Example 12.4

Program to illustrates the use of yield( ) and sleep( ) method

```
class Thread1 extends Thread
{
        public void run( )
        {
                for (int i=1; i<5; i++)
                {
                        if (i==3)
                        yield( ); // Put thread1 in Runnable state
                        System.out.println("Value of i = " + i);
                }
        }
}

class Thread2 extends Thread
{
        public void run( )
        {
                for (int j=1; j<5; j++)
                {
                        try
                        {
                                if (j==2)
                                sleep(1000); // pause execution of Thread2 for 1 second
                        }

                        catch (InterruptedException e)
```

```
                {
                        System.out.println(e);
                }
                System.out.println("Value of j = " +j);

            }
        }
    }

class Tlifecycle
{
    public static void main(String args[])
    {
            Thread1 t1 = new Thread1( );
            Thread2 t2 = new Thread2( );
            t1.start();
            t2.start();
    }
}
```

Output

```
Value of i = 1
Value of j = 1
Value of i = 2
Value of i = 3
Value of i = 4
Value of j = 2
Value of j = 3
Value of j = 4
```