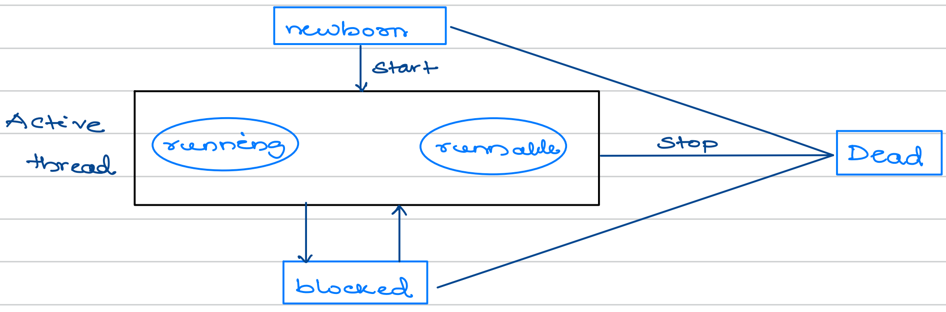
Multithreading :-

* 4 major stages of thread



Newborn → when thread is created

Every thread that user creates is child class of thread class

Start → method to send new thread to active state

$$\text{Running} \underset{\text{yield}}{\overset{\text{run}}{\rightleftarrows}} \text{Running}$$

$$\text{Active} \underset{\text{resume}}{\overset{\text{suspend}}{\rightleftarrows}} \text{Blocked}$$

sleep( ) argument is in millisecond time.

Thread creation

2 ways of thread creation :-

1> extending thread class

2> implementing runnable interface.

(2) better than (1)

* can implement as well as extend also

* multiple inheritance.

1> Extending thread class

```
class Five extends thread
 {
      public void run()
        {
           for ( int i = 1 ; i<=10 ; i++)
             Sopln ( 5* i) ;
          }
      }
   class seven extends thread
        {
           public void sum()
             {
                for ( i=1 ; i<=10 ; i++)
                  super (7* i);
               }
           }
        Class MT1
           {
              psvm ( string args[])
                {
                  Five t1 = new Five();
        Seven t2 = new Seven ( );
        t1. start ( );
        t2. start ( );
    }
}
```

O/P

again run

```
5          5
10         10
15         7
25         14
30         21
7          28
17         15
21         .
.          .
.          .
.          .
.
```

Threads shows asynchronous.

↳ non sequential behavior

Therefore thread synchronization reqd.

↓

done using "synchronized" keyword.

```
abstract class A
{
   A()
    {
       Sopln ("I am constructor of abstract class A");
     }
     abstract void show ();
     abstract void add (int a, int b);
  }
   abstract class B extends A
    {
       B();
        {
          Sopln ("I am constructor of abstract class B");
        }
    }
 class C extends B
    {
     C();
      {
          Sopln ("I am constructor of class C");
        }
        void show ()
         {
           Sopln (" I have completed the show");
         }
         void add (int a, int b)
          {
            int c;
            c = a+b;
            Sopln (" Sum = "+c) ;
          }
          class Main
           {
```

```
class Five implements Runnable
  {
      public void run()
        {
          for ( int i = 1 ; i<=10 ; i++)
            Sopln ( 5* i) ;
         }
      }
   class seven implements runnable
      {
         public void sum()
           {
              for (i=1 ; i<=10 ; i++)
                super (7* i);
           }
         }
      class MT 5
         {
            psvm (string args[])
              {
                 Five t1 = new Five();
                 Seven s = new Seven ();
                 thread t1 = new thread (t);
                 thread t2= new thread (s);
                 t1.start();
                 t2.start() ;
             }
          }
```

* runnable interface is not thread class.

O/P

```
5                 5
10     again      10
     <-------->
15      run       7
7                 14
14                21
21                .
28                .
20                .
.                 ,
.                 ,
.
```

In case of thread class :

run() method

⇓

run() method is abstract

In case of thread extended by class, we creati the object of child class which is also a thread object.

In case of runnable interface, we have to create object of thread class because runnable is not thread class.

```
abstract class A
{
    A()
     {
        Sopln (" I am constructor of abstract class A");
     }
      abstract void show ();
      abstract void add (int a, int b);
  }
  abstract class B extends A
   {
      B();
       {
         Sopln ("I am constructor of abstract class B");
       }
   }
 class C extends B
   {
    C( );
     {
        Sopln ("I am constructor of class C");
      }
      void show ()
       {
          Sopln (" I have completed the show");
       }
       void add (int a , int b)
        {
           int c;
           c=a+b;
           Sopln (" Sum= "+c) ;
        }
        class Main
```

Thread Synchronization

3 approaches :-

1) Synchronized block
2) Synchronized Method
3) Static Synchronization

MT9.java

```
class Share implements Thread
 {
    public void run()
      {
        for ( int i = 1 ; i<=10 ; i++)
          Sopln ( 5* i) ;
      }
    }
  class  seven  implements  runnable
       {
          public  void  sum()
             {
               for  (i=1  ; i<=10  ; i++)
                 super  (7* i);
             }
          }
       class   MT 5
          {
            psvm  ( string  args[])
              {
                Five t1 = new Five();
                Seven  s = new  Seven ();
                thread  t1  =  new  thread  (t);
                thread  t2 = new  thread  (s);
                t1. setPriority(2);
                t2. setPriority(7);
                Sopln ("Priority of t1 :   " t1.getPriority();
                Sopln ("Priority of t2 :   " t2.getPriority();
                t2.start();
                t1. start() ;
            }
    }
```

Output :-

```
Priority of t1  =  5
   "      "  t2  :  5
   "      "  t1  =  2
   "      "  t2  =  7
```

Thread Priorities

↳ ranging from 1 to 10.

1 → least

5 → avg

10 → highest

higher priority → higher chance of being executed.

```
public final int getPriority ()

public final void setPriority (int newPriority);
```

Static Synchronization

* Synchronize is a keyword provided by java.

* we can make a static method synchronization.

```
class A
 {
   synchronized static void (int m)
       {
          for (int i = 1 ; i = 10 ; i++)
            {
               sopln( m* i);
               try
                {
                   thread.sleep(1000);
                }
                catch ( Exception e)
                 {
                 }
              }

            class MyThread1 extends Thread
             {
                 pv sum()
                 {
                  A.ptable (5);
                 }
             class Mythread2 extends Thread
                 {
                   public void sum ()
```

```
        {
            A.ptable(7);
        }
class MT3
  {
      psvm ( string args[] )
        {
            MyThread1 t1 = new Mythread1();
            MyThread2 t2 = new Mythread2();
            t1.start();
            t2.start();
        }
    }
```

* when we call sleep() method to suspend / block the thread, call it within try block

  ⇓

  otherwise sleep() won't work.

  May use empty catch block.

* While writing code of synchro. ; not necessary to write sleep() method.

* separate method of bubble sort → call that within() run() method.

———— × ————