```
        m1.setPriority(Thread.MIN_PRIORITY);
        m2.setPriority(Thread.MAX_PRIORITY);
        m1.start();
        m2.start();
    }
}
```

Output:

```
Running thread name is: Thread-1
Running thread priority is: 10
Running thread name is: Thread-0
Running thread priority is: 1
```

2.6 EXECUTION OF THREAD APPLICATION : RUNNING MULTIPLE THREADS

- The first way is to create a class which extends thread class and then create the instance of that class. This extended class must override method run().
- It must also call start() method. The Thread class is defined in package java.lang; so we have to import it.
- The following code segment tells the definition:

```
import java.lang.*;
public class Coun extends Thread
{
    public void run()
    {
        _______________
        _______________
        _______________
    }
}
```

- This will create a new class Coun and overrides method run(). The program 2.8 shows how to write Thread program.

Program 2.8: Program for multiple threads.

```
import java.lang.*;
class Cons extends Thread
{
    //constructor
    Cons()
    {
        start();  //starts the thread
    }
    public void run()
    {
    try
    {
       for (int k=1;k<=5;++k)
        {
           System.out.println("mythread" + k);
           Thread.sleep(500);
        }
    }
    catch(InterruptedException ob)
    {
    }
    System.out.println("Thread exists");
    } // end run
 } //end Cons
 class Mainthread
 {
    public static void main(String args[])
    {
       Cons c = new Cons();
       try
       {
          for (int i=1;i<=5;++i)
          {
             System.out.println("Main Thread" +i);
             Thread.sleep(1000);
          } //end for
       } //end try
       catch (InterruptedException ob)
       {
       }
       System.out.println("main Thread Exists");
    } //end main
 } // end mainthread
```

Output:

```
 mythread1
 Main Thread1
 mythread2
```

```
mythread3
Main Thread2
mythread4
mythread5
Main Thread3
Thread exists
Main Thread4
Main Thread5
main Thread Exists
```

- This output may change PC to PC. In this program, two threads main thread and mythread (coun) runs simultaneously.
- In the above example, mythread suspends threads for 500 millisecond by calling its sleep method. The mainthread first get control of CPU and count as 1 and then suspended for 1000 milliseconds.
- The Program 2.9 illustrates the calling of the start method from main method and not from constructor. In a constructor, we will use a super method. It gets one argument as string which is name of a thread. In this program, we will create three different objects.

Program 2.9: Program to use of super in threading.

```
class mythread extends Thread
{
    mythread(String name)
    {
        super(name);
    }
    public void run()
    {
        try
        {
            for(int k=5;k>0;k--)
            {
                System.out.println(getName() + k);
                Thread.sleep(500);
            }
        }
        catch(InterruptedException e)
        {
           System.out.println("Thread interrupted");
        }
        System.out.println("Thread exists");
      } //end run()
  } //end mythread
  //Main class for thread object
  class Mainthread1
  {
     public static void main(String args[])
     {
         mythread ob1 = new mythread ("First");
         mythread ob2 = new mythread ("Second");
         mythread ob3 = new mythread ("Third");
         ob1.start();
         ob2.start();
         ob3.start();
         try
         {
            for(int k=5;k>0;k--)
            {
                System.out.println("main thread" +k);
                Thread.sleep(500);
             }
        }
        catch(InterruptedException e)
        {
           System.out.println("interrupted thread");
        }
           System.out.println("main thread exists");
      } //end main
  } //end class mainthread
```

utput:

```
main thread5
Third5
First5
Second5
```

```
Third4
First4
Second4
main thread4
Third3
Second3
First3
main thread3
Third2
First2
Second2
main thread2
Third1
Second1
First1
main thread1
Thread exists
main thread exists
Thread exists
Thread exists
```

- In above Program 2.9, the constructor is parameterized which takes one argument of type string. This argument is the name of thread.
- First, second and third are the names of thread. So in this program, four threads are running simultaneously. This is shown in Fig. 2.7.

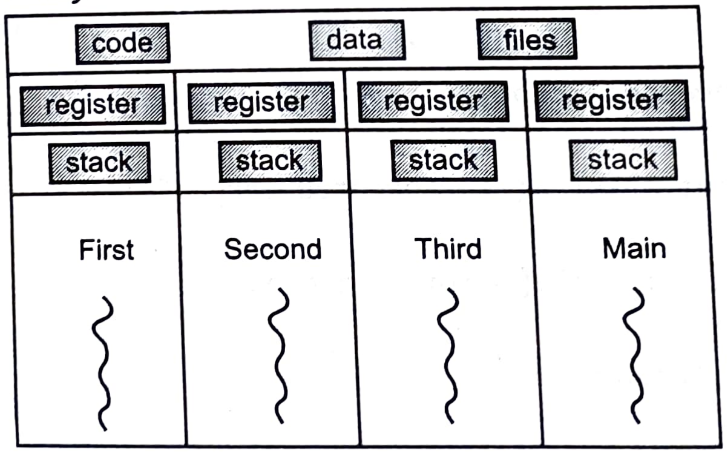

Fig. 2.7: Four threads with CPU

2.7 SYNCHRONIZATION

- In this section we will study synchronization and inter-thread communication in Java.

2.7.1 Synchronization

- Multithreading introduces asynchronous behavior to the programs. If a thread is writing some data another thread may be reading the same data at that time. This may bring inconsistency.
- When two or more threads wants to access a shared resource, then it must ensure that the resource will be used by only one thread at an instant of time. The mechanism of this process is called synchronization.
- Synchronization is the concept of the monitor or semaphore. Monitor works as mutex and restrict to one thread to own a monitor at a given time.
- As the thread acquires the lock, all the threads that want to acquire the monitor will be suspended.
- As the first thread exits from the monitor, one thread will acquire monitor from the waiting list of threads.
- Java programming language provides a very handy way of creating threads and synchronizing their task by using synchronized blocks.
- We keep shared resources within this block. Following is the general form of the synchronized statement:

```
synchronized(objectidentifier) {
   // Access shared variables and other shared resources
}
```

- Here, the objectidentifier is a reference to an object whose lock associates with the monitor that the synchronized statement represents.
- Synchronization in java is the capability to control the access of multiple threads to any shared resource.
- Java Synchronization is better option where we want to allow only one thread to access the shared resource.

Program 2.10: Program for synchronization.

```
class mythread extends Thread
{
   String msg[]={"Java", "Supports", "Multithreading", "Concept"};
   mythread(String name)
   {
      super(name);
   }
   public void run()
   {
      display(getName());
      System.out.println("Exit from "+getName());
   }
```

```
                catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
            }
            System.out.println(msg);
            flag = false;
            notify();
            }
    }
    class T1 implements Runnable
    {
        Chat m;
        String[] s1 = { "Hi", "How are you?", "I am also doing fine!" };
        public T1(Chat m1)
        {
            this.m = m1;
            new Thread(this, "Question").start();
        }
        public void run()
        {
            for (int i = 0; i < s1.length; i++)
            {
                m.Question(s1[i]);
            }
        }
    }
    class T2 implements Runnable
    {
        Chat m;
        String[] s2 = { "Hi", "I am good, what about you?", "Great!" };
        public T2(Chat m2)
        {
            this.m = m2;
            new Thread(this, "Answer").start();
        }
    public void run()
    {
        for (int i = 0; i < s2.length; i++)
        {
            m.Answer(s2[i]);
        }
    }
}
public class MyDemoThread
{
    public static void main(String[] args)
    {
        Chat m = new Chat();
        new T1(m);
        new T2(m);
    }
}
```

Output:

```
Hi
Hi
How are you?
I am good, what about you?
I am also doing fine!
Great!
```

Additional Programs

Program 2.12: Program to display the 100, 99, 98...... 1 using thread.

```
class MyThredDemo1
{
    public static void main(String args[])
    {
        Thread t = Thread.currentThread();
        System.out.println("Current thread is: " + t);
        t.setName("Demo Thread");
        System.out.println("After changing the name thread is: " + t);
        try
        {
            for(int n = 100; n> 0; n--)
            {
              System.out.println(n);
              Thread.sleep(1000);
            }
          }
          catch (InterruptedException e)
          {
            System.out.println("Thread interrupted");
          }
        }
      }
```

Output:

```
Current thread is: Thread[main,5,main]
After changing the name thread is: Thread[Demo Thread,5,main]
100
99
98
97
96
95
94
93
92
91
90
89
88
87
86
85
84
83
82
81
80
79
78
```

```
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
```

```
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
```

Program 2.13: Program to use of sleep() method.

```
class NewThread implements Runnable
{
    Thread t;
    NewThread()
    {
        t = new Thread(this, "Demo Thread");
        System.out.println("Child Thread: " + t);
        t.start();
    }
    public void run()
    {
        try
        {
            for (int i = 0; i < 5;  i++)
            {
                System.out.println("Child Thread: " + i);
                Thread.sleep(1000);
            }
        }
        catch (InterruptedException e)
        {
            System.out.println("Child Thread Interrupted");
        }
        System.out.println("Exiting child thread");
    }
}
class DemoMyThread2
{
    public static void main(String args[])
    {
        new NewThread();
        try
        {
            for (int i = 0;  i < 5; i++)
            {
                System.out.println("Main Thread: " + i);
                Thread.sleep(500);
            }
        }
        catch (InterruptedException e)
        {
            System.out.println("Main Thread Interrupted");
        }
        System.out.println("Exiting main thread");
    }
}
```

Output:

```
Child Thread: Thread[Demo Thread,5,main]
Child Thread: 0
Main Thread: 0
Main Thread: 1
Child Thread: 1
Main Thread: 2
Main Thread: 3
Child Thread: 2
Main Thread: 4
Exiting main thread
Child Thread: 3
Child Thread: 4
Exiting child thread
```

Program 2.14: Program to use super().

```
class NewThread extends Thread
{
    NewThread()
    {
        super( "Demo Thread");
        System.out.println("Child Thread : " + this);
        start();
    }
    public void run()
    {
        try
        {
            for (int i = 0; i < 5;  i++)
            {
                System.out.println("Child Thread: " + i);
                Thread.sleep(1000);
            }
        }
        catch (InterruptedException e)
        {
            System.out.println("Child Thread Interrupted");
        }
        System.out.println("Exiting child thread");
    }
}
class DemoMyThread3
{
    public static void main(String args[])
    {
        new NewThread();
        try
        {
            for (int i = 0;  i < 5; i++)
            {
                System.out.println("Main Thread: " + i);
                Thread.sleep(500);
            }
        }
        catch (InterruptedException e)
        {
            System.out.println("Main Thread Interrupted");
        }
        System.out.println("Exiting main thread");
    }
}
```

Output:

```
Child Thread: Thread[Demo Thread,5,main]
Main Thread: 0
Child Thread: 0
Main Thread: 1
Child Thread: 1
Main Thread: 2
Main Thread: 3
Child Thread: 2
Main Thread: 4
Exiting main thread
Child Thread: 3
Child Thread: 4
Exiting child thread
```

Program 2.15: Java program to display "BYE CORONA..." message n times using Runnab Interface.

```
import java.io.*;
public class DemoMyThread4 implements Runnable
{
   int i, no;
   DemoMyThread4(int n)
   {
      no = n;
   }
   public void run()
   {
      for(i = 1; i<=no; i++)
      {
         System.out.println("BYE CORONA...");
         try
         {
            Thread.sleep(50);
         }
         catch(Exception e)
         {
            System.out.println(e);
         }
      }
   }
```

```
    public static void main(String args[])
    {
      try
      {
         int n;
         System.out.println("\nHow many time you want? ");
         BufferedReader br = new BufferedReader(new
                                       InputStreamReader(System.in));
         String str = br.readLine();
         n = Integer.parseInt(str);
         Thread t = new Thread(new DemoMyThread4(n));
         t.start();
      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

utput:

```
How many time you want?
5
BYE CORONA...
BYE CORONA...
BYE CORONA...
BYE CORONA...
BYE CORONA...
```

rogram 2.16: Program to define a thread for printing text on output screen for 'n' umber of times. Create 3 threads and run them. Pass the text 'n' parameters to the hread constructor.

xample:

(a) First thread prints "COVID19" 10 times.

(b) Second thread prints "LOCKDOWN2020" 20 times

(c) Third thread prints "VACCINATED2021" 30 times