```java
package com.pack1;

public class ClassA
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("run() executed : "+i);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t=new Thread();
        t.start();
    }
}
```

In above thread class run method is executed

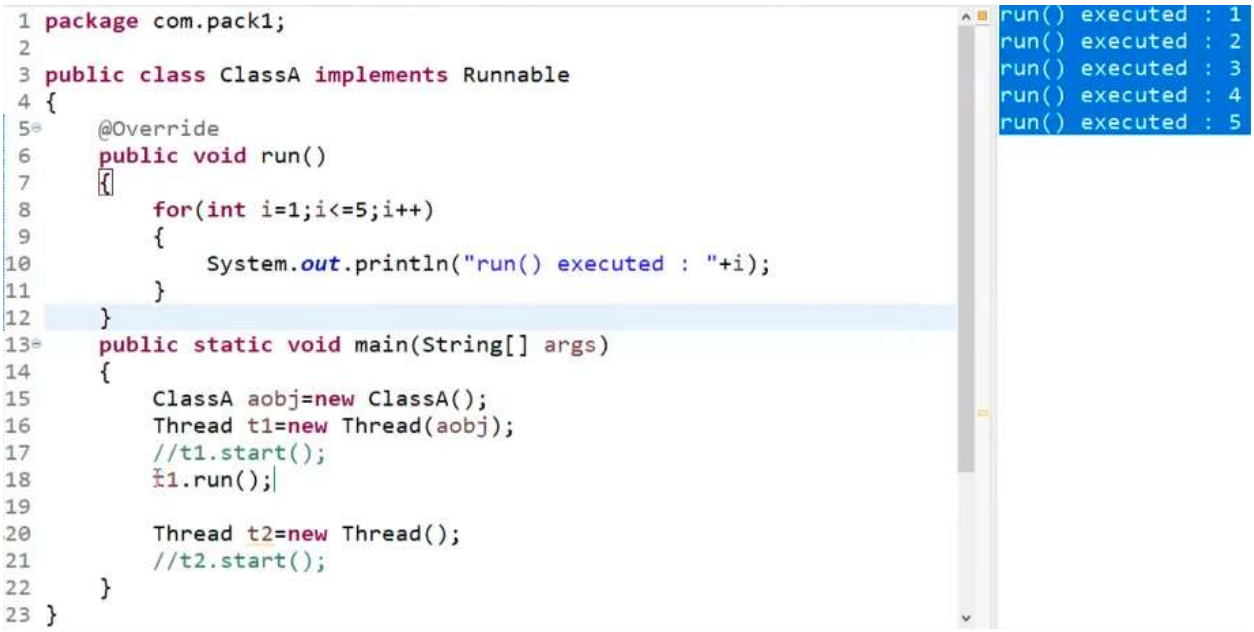
To execute our class run method

```java
package com.pack1;

public class ClassA
{
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("run() executed : "+i);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t=new Thread(aobj);
        t.start();
    }
}
```

```
1 package com.pack1;
2
3 public class ClassA implements Runnable
4 {
5     @Override
6     public void run()
7     {
8         for(int i=1;i<=5;i++)
9         {
10            System.out.println("run() executed : "+i);
11        }
12    }
13    public static void main(String[] args)
14    {
15        ClassA aobj=new ClassA();
16        Thread t=new Thread(aobj);
17        t.start();
18    }
19 }
```

```
run() executed : 1
run() executed : 2
run() executed : 3
run() executed : 4
run() executed : 5
```

```
1  package com.pack1;
2
3  public class ClassA implements Runnable
4  {
5      @Override
6      public void run()
7      {
8          for(int i=1;i<=5;i++)
9          {
10             System.out.println("run() executed : "+i);
11         }
12     }
13     public static void main(String[] args)
14     {
15         ClassA aobj=new ClassA();
16         Thread t1=new Thread(aobj);
17         //t1.start();
18
19         Thread t2=new Thread();
20         t2.start();
21     }
22 }
```



```
package com.pack1;

public class ClassA implements Runnable
{
    @Override
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("run() executed : "+i);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        Thread t1=new Thread(aobj);
        //t1.start();
        t1.run();

        Thread t2=new Thread();
        //t2.start();
    }
}
```

```
run() executed : 1
run() executed : 2
run() executed : 3
run() executed : 4
run() executed : 5
```

```java
public class ClassA implements Runnable
{
    @Override
    public void run()
    {
        for(int i=1;i<=5;i++)
        {
            System.out.println("run() executed : "+i);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        Thread t1=new Thread(aobj);
        //t1.start();
        /*
         Whenever we are calling start() a new thread is created and that
         new thread is responsible for running the run() which is present
         in ClassA because of overridding concept
         */

        Thread t1=new Thread(aobj);
        //t1.start();
        /*
         Whenever we are calling start() a new thread is created and that
         new thread is responsible for running the run() which is present
         in ClassA because of overridding concept
         */

        //t1.run();
        /*
         Whenever we are calling run() NO new thread is created. We are calling
         run() just like a normal method call & as we are performing
         method overridding ClassA run() will be executed.
         */

        Thread t2=new Thread();
        //t2.start();
        /*
         Whenever we are calling start() a new thread is created and that
         new thread is responsible for running the run() which is present
         in Thread Class

         */

        //t2.run();
        /*
         Whenever we are calling run() NO new thread is created. We are calling
         run() just like a normal method call.(Thread Class run() will be executed)
         */
    }
```

```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[]
6     {
7         System.out.println()
8     }
9 }
```

```
Exception in thread "main" java.lang.Error: Unresolved compilatior
        Syntax error, insert ";" to complete BlockStatements

        at Training/com.pack1.ClassB.main(ClassB.java:7)
```

Main method is a thread

Creating Thread by implementing Runnable interface

```
public class ClassA implements Runnable
{
   public void run()
    {
      for(int i=0;i<5;i++)
      System.out.println("Run method");
    }
   public static void main(String[] args)
    {
     ClassA a=new ClassA();
     Thread t1=new Thread(a);
     Thread t2=new Thread();
      System.out.println("Java is awesome");
    }
}
```

```
t1.start();
t1.run();
t2.start();
t2.run();
```

t1.start()

New Thread will be generated which is responsible for the execution of **ClassA** run() method.

t1.run()

No new Thread will be generated but **ClassA** run() method will be called just like a normal method call.

t2.start()

A new Thread will be generated which is responsible for the implementation of **Thread class** run()method

t2.run()

No new Thread will be generated but **Thread class** run() method will be called just like a normal method call.

Creating Thread by extending Thread class

```
public class ClassA extends Thread
        {
                public void run()
                {
                        for(int i=0;i<5;i++)
                        System.out.println("Run method");
                }
                public static void main(String[] args)
                {
                        ClassA a=new ClassA();
                        a.start();
                        System.out.println("Java is awesome");
                }
        }
```

Life Cycle of a Thread

New	Thread is created but not yet started.
Runnable	A thread in the Runnable state is executing in the Java virtual machine but it may be waiting for other resources from the operating system such as processor
Blocked	A thread in the blocked state is waiting to enter a synchronized block/method or reenter a synchronized block/method.
Waiting	A thread will be in waiting state for a unspecified period of time, due to calling one of the methods like **wait(),join()** etc
Timed_waiting	A thread will be in waiting state for another thread for a specified waiting time is in this state
Terminated	The thread has completed execution

A thread can be in only one state at a given point in time. **Thread.getState()**

Getting and setting name of a Thread:

- Every Thread in java has some name it may be provided explicitly by the programmer or automatically generated by JVM.
- Thread class defines the following methods to get and set name of a Thread.
  - ✓ public final String getName()
  - ✓ public final void setName(String name)

Understanding Thread Priorities

- In the Java programming language, every thread has a priority.
- We can increase or decrease the priority of any thread by using setPriority(int newPriority) method.
- We can get the priority of the thread by using getPriority() method
- Priority can either be given by JVM (5) while creating the thread or it can be given by programmer explicitly.
- Accepted value of priority for a thread is in range of 1 to 10.
- Thread priorities are highly system-dependent we should always keep in mind that underlying platform should provide support for scheduling based on thread priority.

- There are 3 static variables defined in Thread class for priority.

public static int MIN_PRIORITY --->1

public static int NORM_PRIORITY --->5

public static int MAX_PRIORITY --->10

Thread.currentThread(); is going to provide the instance(object) of the current executing Thread

```
2
3  public class ClassA extends Thread
4  {
5      @Override
6      public void run()
7      {
8          String name=Thread.currentThread().getName();
9          System.out.println(name+" has entered run()");
10         for(int i=1;i<=5;i++)
11         {
12             System.out.println(name+" : "+i);
13         }
14         System.out.println(name+" completed run() execution");
15     }
16     public static void main(String[] args)
17     {
18         ClassA aobj=new ClassA();
19
20         Thread t1=new Thread(aobj);
21         Thread t2=new Thread(aobj);
22
23         t1.setName("First-Thread");
24         t2.setName("Second-Thread");
```

```
First-Thread has entered run()
First-Thread : 1
First-Thread : 2
First-Thread : 3
First-Thread : 4
First-Thread : 5
First-Thread completed run() executior
```

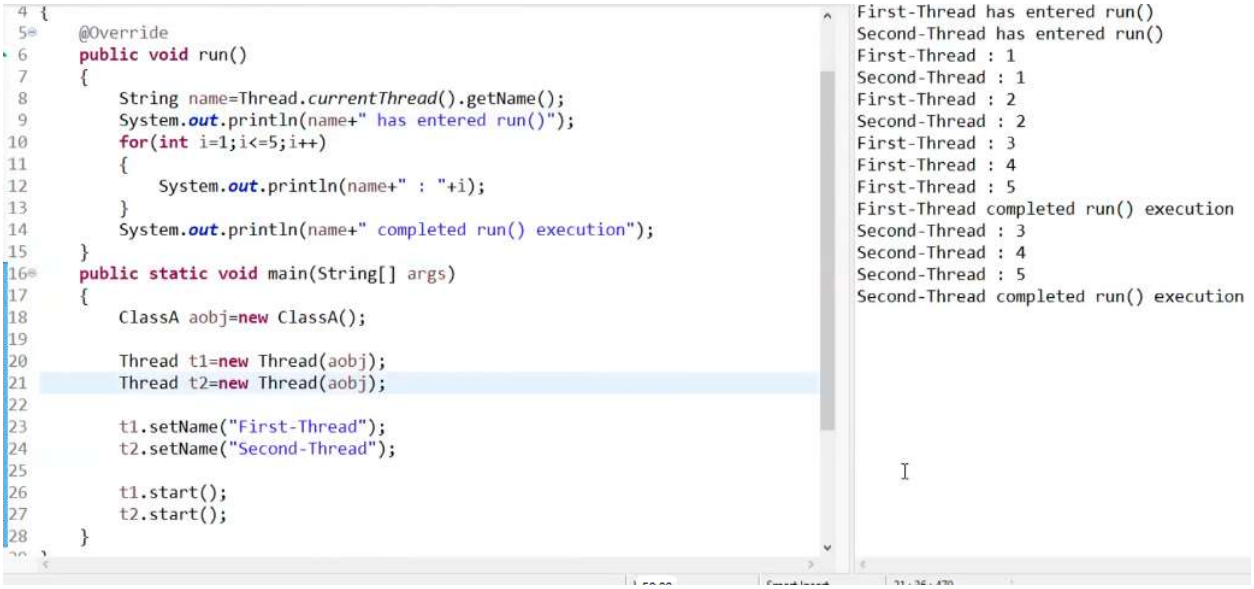
```
25
26             t1.start();
27             //t2.start();
28         }
```

```
4  {
5      @Override
6      public void run()
7      {
8          String name=Thread.currentThread().getName();
9          System.out.println(name+" has entered run()");
10         for(int i=1;i<=5;i++)
11         {
12             System.out.println(name+" : "+i);
13         }
14         System.out.println(name+" completed run() execution");
15     }
16     public static void main(String[] args)
17     {
18         ClassA aobj=new ClassA();
19
20         Thread t1=new Thread(aobj);
21         Thread t2=new Thread(aobj);
22
23         t1.setName("First-Thread");
24         t2.setName("Second-Thread");
25
26         //t1.start();
27         t2.start();
28     }
```

```
Second-Thread has entered run()
Second-Thread : 1
Second-Thread : 2
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
Second-Thread completed run() execution
```

```
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" completed run() execution");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("First-Thread");
        t2.setName("Second-Thread");

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

```
First-Thread has entered run()
Second-Thread has entered run()
First-Thread : 1
First-Thread : 2
First-Thread : 3
First-Thread : 4
First-Thread : 5
First-Thread completed run() execution
Second-Thread : 1
Second-Thread : 2
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
Second-Thread completed run() execution
```



```
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+" : "+i);
        }
        System.out.println(name+" completed run() execution");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("First-Thread");
        t2.setName("Second-Thread");

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

```
First-Thread has entered run()
Second-Thread has entered run()
First-Thread : 1
Second-Thread : 1
First-Thread : 2
Second-Thread : 2
First-Thread : 3
First-Thread : 4
First-Thread : 5
First-Thread completed run() execution
Second-Thread : 3
Second-Thread : 4
Second-Thread : 5
Second-Thread completed run() execution
```

No change in the code but why are we getting different types of answers (we cannot guess the output in threads)

```java
public class ClassA extends Thread
{
    @Override
    public void run()
    {
        String name=Thread.currentThread().getName();
        int priority=Thread.currentThread().getPriority();

        System.out.println(name+" has entered run()");
        for(int i=1;i<=5;i++)
        {
            System.out.println(name+"("+priority+")"+" : "+i);
        }
        System.out.println(name+" completed run() execution");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t1=new Thread(aobj);
        Thread t2=new Thread(aobj);

        t1.setName("First-Thread");
        t2.setName("Second-Thread");


        t1.setPriority(1);      //t1.setPriority(MIN_PRIORITY);
        t2.setPriority(10);     //t2.setPriority(MAX_PRIORITY);

        t1.start();
        t2.start();
    }
}

/*
  Thread Schedular :
  -----------------
  It is going to decide which Thread should start its execution 1st basing on '2' aspects

       =====> Thread Priorities
       =====> Underlying OS.
```

```
/*
  Thread Schedular :
  -----------------
  It is going to decide which Thread should start its execution 1st basing on '2' aspects

       =====> Thread Priorities
       =====> Underlying OS.

  Thread Priorities:    [1 to 10]
  ------------------

       ===> Minimum Priority : 1
       ===> Normal / Default Priority : 5
       ===> Maximum Priority : 10

*/
```