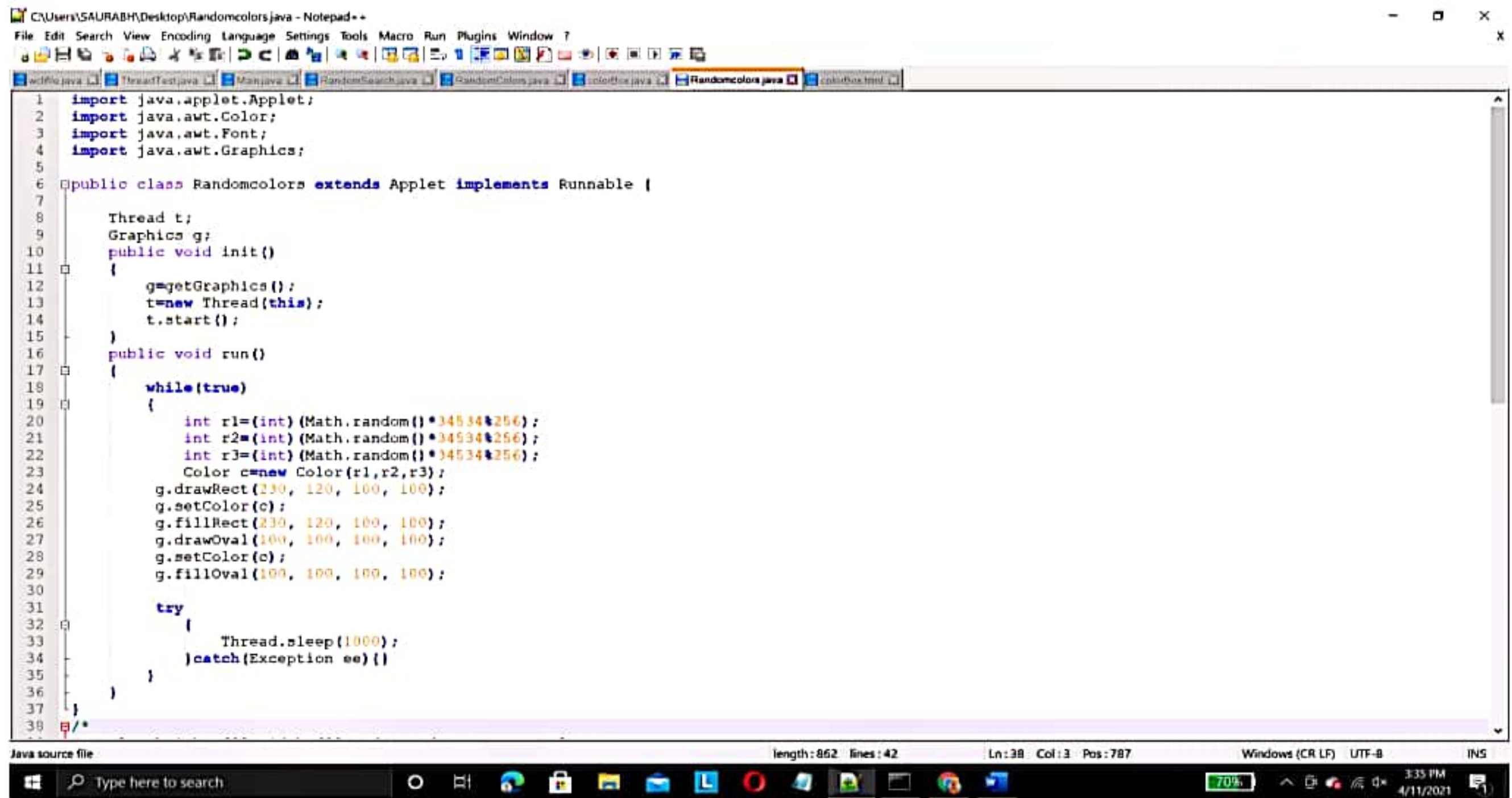


# QUESTION1

1) Construct a Java program to display two different shapes in an applet or a frame and fill these two different shapes with random colors at the same instance of time using multi-threading concept in java.

# ANSWER1



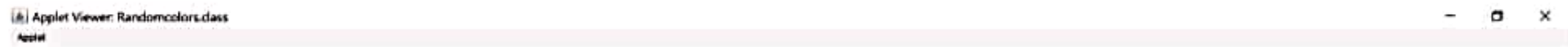
```
1 import java.applet.Applet;
2 import java.awt.Color;
3 import java.awt.Font;
4 import java.awt.Graphics;
5
6 public class RandomColors extends Applet implements Runnable {
7
8     Thread t;
9     Graphics g;
10    public void init()
11    {
12        g=getGraphics();
13        t=new Thread(this);
14        t.start();
15    }
16    public void run()
17    {
18        while(true)
19        {
20            int r1=(int) (Math.random()*255);
21            int r2=(int) (Math.random()*255);
22            int r3=(int) (Math.random()*255);
23            Color c=new Color(r1,r2,r3);
24            g.drawRect(230, 120, 100, 100);
25            g.setColor(c);
26            g.fillRect(230, 120, 100, 100);
27            g.drawOval(100, 100, 100, 100);
28            g.setColor(c);
29            g.fillOval(100, 100, 100, 100);
30
31            try
32            {
33                Thread.sleep(1000);
34            } catch (Exception ee) {}
35        }
36    }
37 }
38 /*
```

Java source file length: 862 lines: 42 Ln: 38 Col: 3 Pos: 787 Windows (CR LF) UTF-8 INS 70% 3:33 PM 4/11/2021

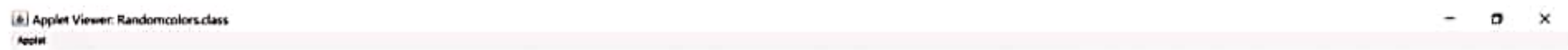
```
Command Prompt - appletviewer Randomcolors.html

C:\Users\SAURABH\Desktop>javac Randomcolors.java

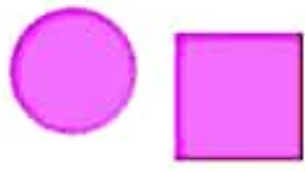
C:\Users\SAURABH\Desktop>appletviewer Randomcolors.html
```



Applet frame will show shape filled with different colors at the same instance of time



Applet Viewer: Randomcolors.class  
Applet



Applet started  
Type here to search  
Applet Viewer: Randomcolors.class  
Applet



Applet started  
Type here to search  
Applet Viewer: Randomcolors.class  
Applet

CODE =

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;

public class Randomcolors extends Applet implements Runnable {

    Thread t;
    Graphics g;

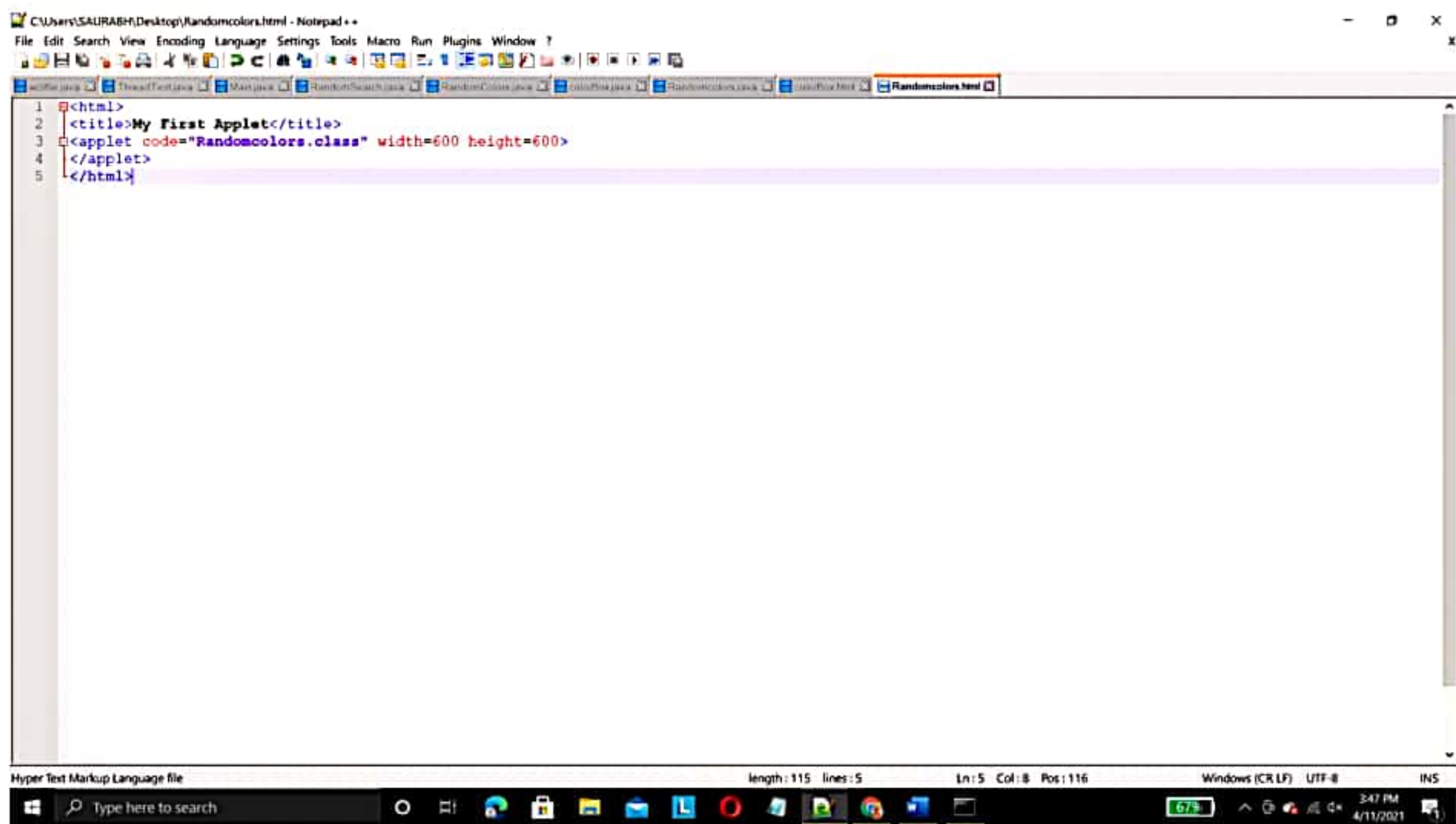
    public void init()
    {
        g=getGraphics();
        t=new Thread(this);
        t.start();
    }

    public void run()
    {
        while(true)
        {
            int r1=(int)(Math.random()*34534%256);
            int r2=(int)(Math.random()*34534%256);
            int r3=(int)(Math.random()*34534%256);
            Color c=new Color(r1,r2,r3);

            g.drawRect(230, 120, 100, 100);
            g.setColor(c);
            g.fillRect(230, 120, 100, 100);
            g.drawOval(100, 100, 100, 100);
            g.setColor(c);
            g.fillOval(100, 100, 100, 100);
        }
    }
}
```



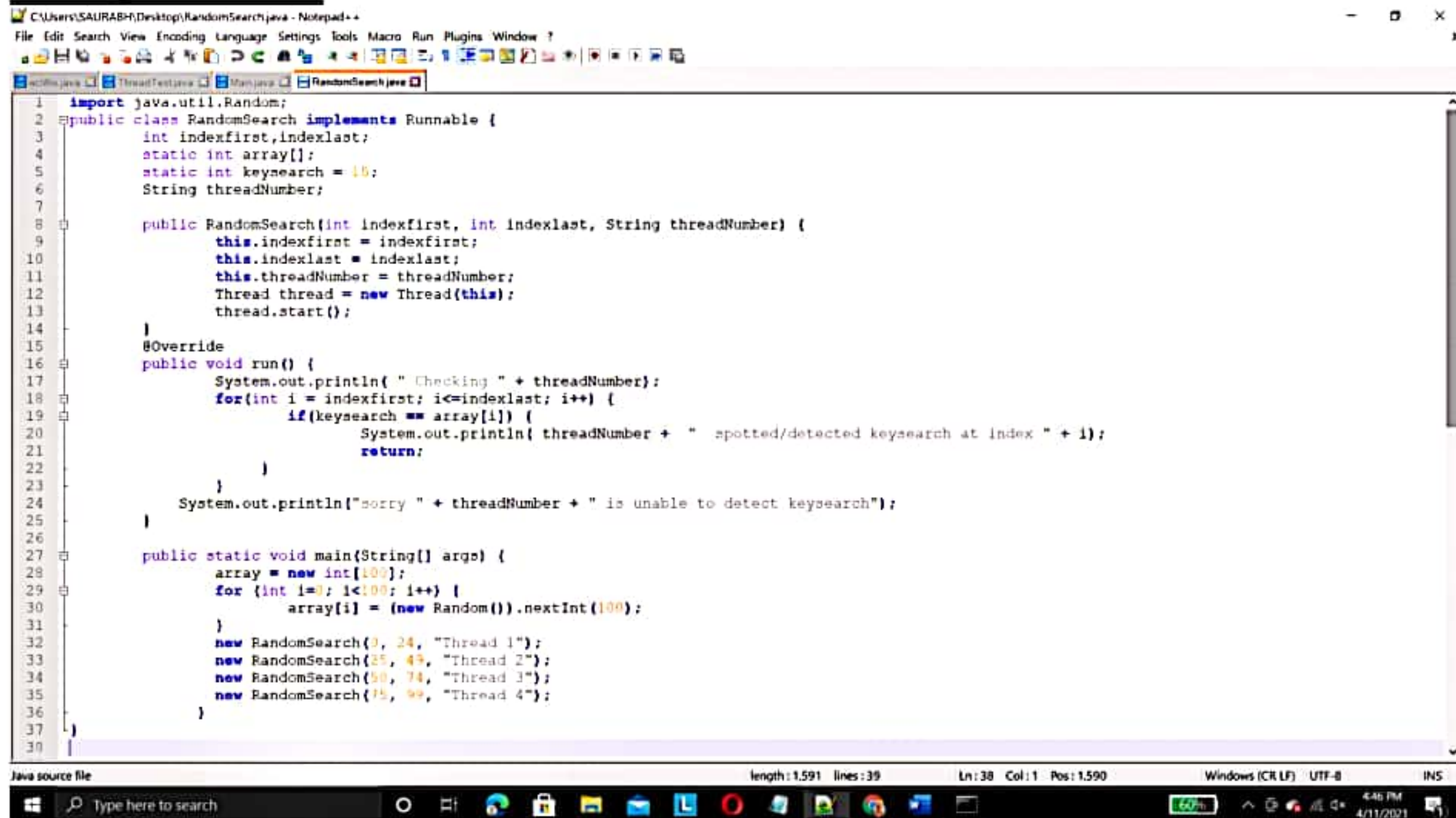
```
try
{
    Thread.sleep(1000);
}catch(Exception ee){}
}
}
}
/*
<applet height=600 width=600 code=Randomcolors>
</applet>
*/
```



## QUESTION -2

2) Construct a multi-threaded Java program to search for an element in the randomly initialized input array of size 100 elements and each thread is likely to search for an equally partitioned array. Suppose if you consider two threads, then each thread searches for 50 elements.

## ANSWER2



```
1  import java.util.Random;
2  public class RandomSearch implements Runnable {
3      int indexfirst, indexlast;
4      static int array[];
5      static int keysearch = 45;
6      String threadNumber;
7
8      public RandomSearch(int indexfirst, int indexlast, String threadNumber) {
9          this.indexfirst = indexfirst;
10         this.indexlast = indexlast;
11         this.threadNumber = threadNumber;
12         Thread thread = new Thread(this);
13         thread.start();
14     }
15     @Override
16     public void run() {
17         System.out.println(" Checking " + threadNumber);
18         for(int i = indexfirst; i<=indexlast; i++) {
19             if(keysearch == array[i]) {
20                 System.out.println(threadNumber + " spotted/detected keysearch at index " + i);
21                 return;
22             }
23         }
24         System.out.println("sorry " + threadNumber + " is unable to detect keysearch");
25     }
26
27     public static void main(String[] args) {
28         array = new int[100];
29         for (int i=0; i<100; i++) {
30             array[i] = (new Random()).nextInt(100);
31         }
32         new RandomSearch(0, 24, "Thread 1");
33         new RandomSearch(25, 49, "Thread 2");
34         new RandomSearch(50, 74, "Thread 3");
35         new RandomSearch(75, 99, "Thread 4");
36     }
37 }
38
```

Java source file length: 1,591 lines: 39 Ln: 38 Col: 1 Pos: 1,590 Windows (CRLF) UTF-8 INS



```
Command Prompt
C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 2
sorry Thread 2 is unable to detect keysearch
Checking Thread 4
sorry Thread 4 is unable to detect keysearch
Checking Thread 3
sorry Thread 3 is unable to detect keysearch
Checking Thread 1
sorry Thread 1 is unable to detect keysearch

C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 1
sorry Thread 1 is unable to detect keysearch
Checking Thread 4
Thread 4 spotted/detected keysearch at index 87
Checking Thread 2
sorry Thread 2 is unable to detect keysearch
Checking Thread 3
sorry Thread 3 is unable to detect keysearch

C:\Users\SAURABH\Desktop>

Type here to search
4:47 PM
4/11/2021

Command Prompt
C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 2
sorry Thread 2 is unable to detect keysearch
Checking Thread 4
sorry Thread 4 is unable to detect keysearch
Checking Thread 1
sorry Thread 1 is unable to detect keysearch
Checking Thread 3
Thread 3 spotted/detected keysearch at index 70

C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 2
Checking Thread 3
Thread 3 spotted/detected keysearch at index 59
Checking Thread 1
Checking Thread 4
sorry Thread 1 is unable to detect keysearch
Thread 2 spotted/detected keysearch at index 49
sorry Thread 4 is unable to detect keysearch

C:\Users\SAURABH\Desktop>

Type here to search
4:48 PM
4/11/2021
```

```
Command Prompt
C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 1
sorry Thread 1 is unable to detect keysearch
Checking Thread 2
sorry Thread 2 is unable to detect keysearch
Checking Thread 3
sorry Thread 3 is unable to detect keysearch
Checking Thread 4
Thread 4 spotted/detected keysearch at index 99

C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 2
Checking Thread 4
Thread 4 spotted/detected keysearch at index 77
Checking Thread 3
sorry Thread 3 is unable to detect keysearch
Checking Thread 1
sorry Thread 1 is unable to detect keysearch
sorry Thread 2 is unable to detect keysearch

C:\Users\SAURABH\Desktop>

Type here to search
Command Prompt
C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 2
sorry Thread 2 is unable to detect keysearch
Checking Thread 4
Thread 4 spotted/detected keysearch at index 95
Checking Thread 1
sorry Thread 1 is unable to detect keysearch
Checking Thread 3
Thread 3 spotted/detected keysearch at index 52

C:\Users\SAURABH\Desktop>java RandomSearch
Checking Thread 1
sorry Thread 1 is unable to detect keysearch
Checking Thread 3
sorry Thread 3 is unable to detect keysearch
Checking Thread 2
Checking Thread 4
sorry Thread 4 is unable to detect keysearch
sorry Thread 2 is unable to detect keysearch

C:\Users\SAURABH\Desktop>
```



CODE=

```
import java.util.Random;
public class RandomSearch implements Runnable {
    int indexfirst, indexlast;
    static int array[];
    static int keysearch = 15;
    String threadNumber;

    public RandomSearch(int indexfirst, int indexlast, String threadNumber) {
        this.indexfirst = indexfirst;
        this.indexlast = indexlast;
        this.threadNumber = threadNumber;
        Thread thread = new Thread(this);
        thread.start();
    }
    @Override
    public void run() {
        System.out.println(" Checking " + threadNumber);
        for(int i = indexfirst; i<=indexlast; i++) {
            if(keysearch == array[i]) {
                System.out.println( threadNumber + " spotted/detected keysearch at index " + i);
                return;
            }
        }
        System.out.println("sorry " + threadNumber + " is unable to detect keysearch");
    }

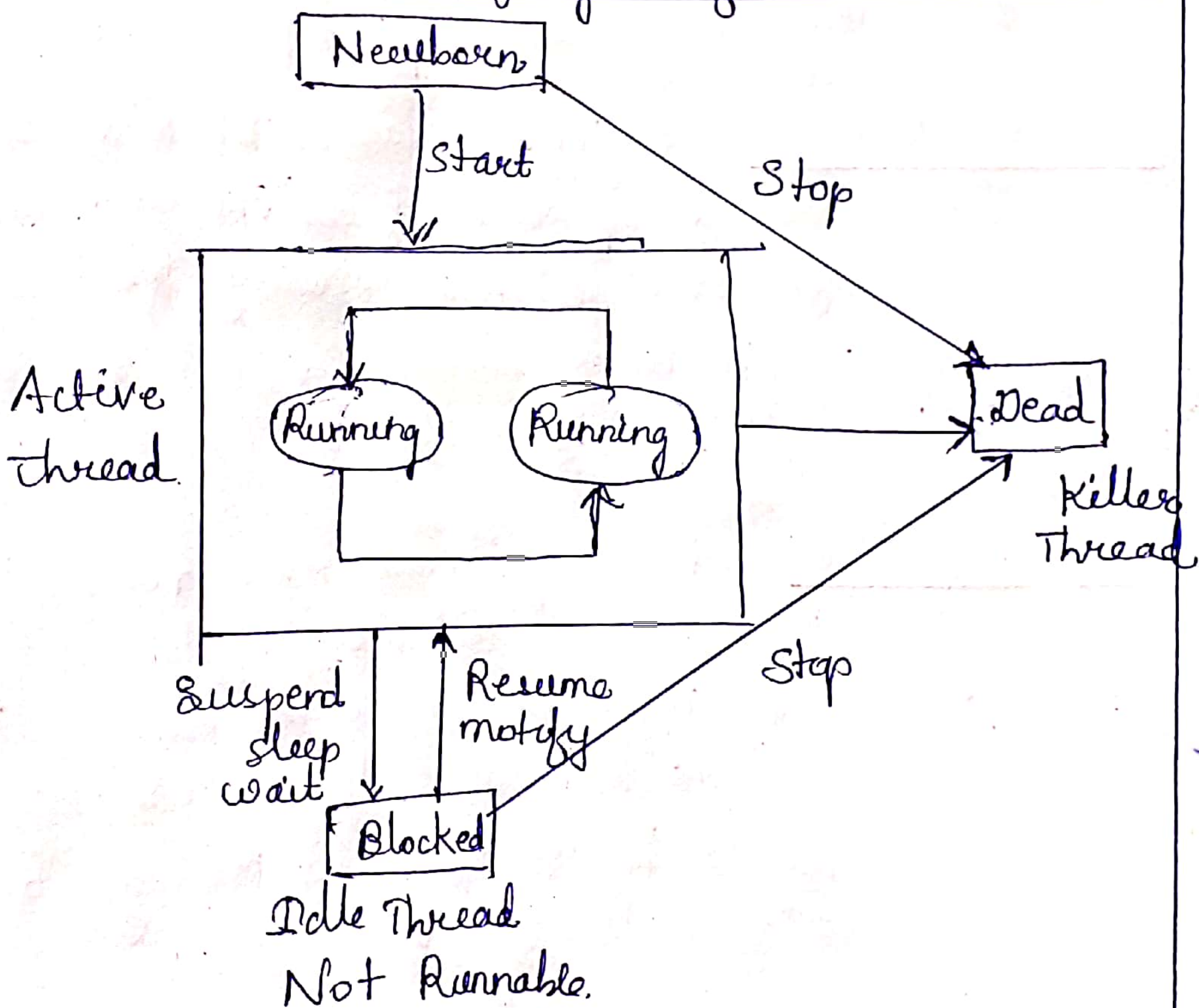
    public static void main(String[] args) {
        array = new int[100];
        for (int i=0; i<100; i++) {
            array[i] = (new Random()).nextInt(100);
        }
        new RandomSearch(0, 24, "Thread 1");
        new RandomSearch(25, 49, "Thread 2");
        new RandomSearch(50, 74, "Thread 3");
        new RandomSearch(75, 99, "Thread 4");
    }
}
```



There are mainly 5 stages of Thread -

- 1) Newborn state.
- 2) Runnable state.
- 3) Running state.
- 4) Blocked state.
- 5) Dead state.

### Life Cycle of Thread.



① Newborn state → When a thread object is created, thread is born & when thread is in newborn state it can be either scheduled for running using **start()** method or kill it using **stop method ()**.



② Runnable State → In this state of thread, it's ready for execution & waiting for availability of processor.

③ Running State → Here, the processor has given its time to the thread for its execution. The thread runs until it releases control on its own or is preempted by high order priority thread. It can be suspended or either made to sleep.

④ Blocked State → Thread can be blocked state when it's prevented from entering into runnable state & subsequently the running state. A blocked thread is not runnable but not dead.

⑤ Dead State - Running thread now reaches its end when it has completed executing its `run()` method. It is the natural death of the thread. Thread can be killed by sending stop message to it.



# Multi-Threading

Multithreading means multiple flow of control. Multithreading programming is conceptual paradigm for programming where one can divide a program into threads & run in parallel. Program with multithreads result in better utilization of system of resources.

Threads can be Created by -

- ① Extending Thread Keyword.
- ② Using Runnable Interface.

Daemon Threads - Low priority threads, & service oriented. They have MIN priority. These threads work intermittently in background for doing garbage collection.

Multithreading and Multiprocessing use multitasking. We use multithreading than multiprocessing because thread uses a shared memory area. They don't allocate separate memory area.

## Multitasking

Process  
Based

Multiprocessing

Thread  
Based

Multithreading



## Constructors for Thread class

- Thread ()
- Thread (String name)
- Thread (Runnable r)
- Thread (Runnable r, String name)

## Example Extending Thread

```
class ABC extends Thread {  
    public void run() {  
        S.o.pln("Thread is running.");  
    }  
    P.s.v.m (String args[]) {  
        ABC a = new ABC();  
        a.start();  
    }  
}
```

## Example Implementing Runnable Interface

```
class Multi implements Runnable {  
    public void run() {  
        S.o.pln("Start Thread");  
    }  
    P.s.v.m (String args[]) {  
        Multi m = new Multi();  
        Thread t1 = new Thread(m);  
        t1.start();  
    }  
}
```



If you are not extending Thread class, instead using Runnable then, your class object would not be treated as object. So you need to create explicitly Thread class object. We are passing the object of your class that implements Runnable so that run() method executes.





wclfile.java ThreadTest.java Main.java RandomSearch.java

```
1  class Ascending extends Thread
2  {
3      public void run()
4      {
5          System.out.println("\n\tAscending Order of numbers::");
6          for(int i=1;i<=10;i++)
7          {
8              System.out.println("\t"+i);
9          }
10     }
11 }
12 class Descending extends Thread
13 {
14     public void run()
15     {
16         System.out.println("\n\tDescending Order of numbers::");
17         for(int i=10;i>=1;i--)
18         {
19             System.out.println("\t"+i);
20         }
21     }
22 }
23 class ThreadTest
24 {
25     public static void main(String args[])
26     {
27         System.out.println("\n\tOne thread printing numbers in Ascending Order is started");
28         new Ascending().start();
29         System.out.println("\n\tOne thread printing numbers in Descending Order is started");
30         new Descending().start();
31     }
32 }
33 }
```



Ascending Order of numbers::

1  
2  
3  
4  
5  
6  
7  
8  
9

Descending Order of numbers::

10  
10  
9  
8  
7  
6  
5  
4  
3



Type here to search



52%



5:21 PM  
4/11/2021

