

NAME – MAYANK BAGAULI

ROLL_NO. -200186

1. Write a java program for Client Server communication using UDP Datagram Socket Programming.

Client side :

```
import java.net.* ;
```

```
import java.io.* ;
```

```
public class client{
```

```
public static void main(String[] args) throws  
IOException
```

```
{
```

```
Socket s = new Socket("localhost", 4999);
```

```
PrintWriter pr = new PrintWriter(s.getOutputStream());
```

```
pr.println("is it working");
```

```
pr.println(" UDP is created by Mayank");
```

```
pr.flush();
```

```
InputStreamReader in = new  
InputStreamReader(s.getInputStream());  
BufferedReader bf = new BufferedReader(in);
```

```
String str = bf.readLine();  
System.out.println("server : "+ str);  
    }  
}
```

Server side :

```
import java.net.* ;
```

```
import java.io.*;
```

```
public class server{

    public static void main(String[] args) throws
    IOException
    {

        ServerSocket ss = new ServerSocket(4999);

        Socket s= ss.accept();

        System.out.println("client connected");

        InputStreamReader in = new InputStreamReader
        (s.getInputStream());

        BufferedReader bf = new BufferedReader(in);

        String str = bf.readLine();

        System.out.println("client : "+ str);
```

```
PrintWriter pr = new PrintWriter  
(s.getOutputStream());
```

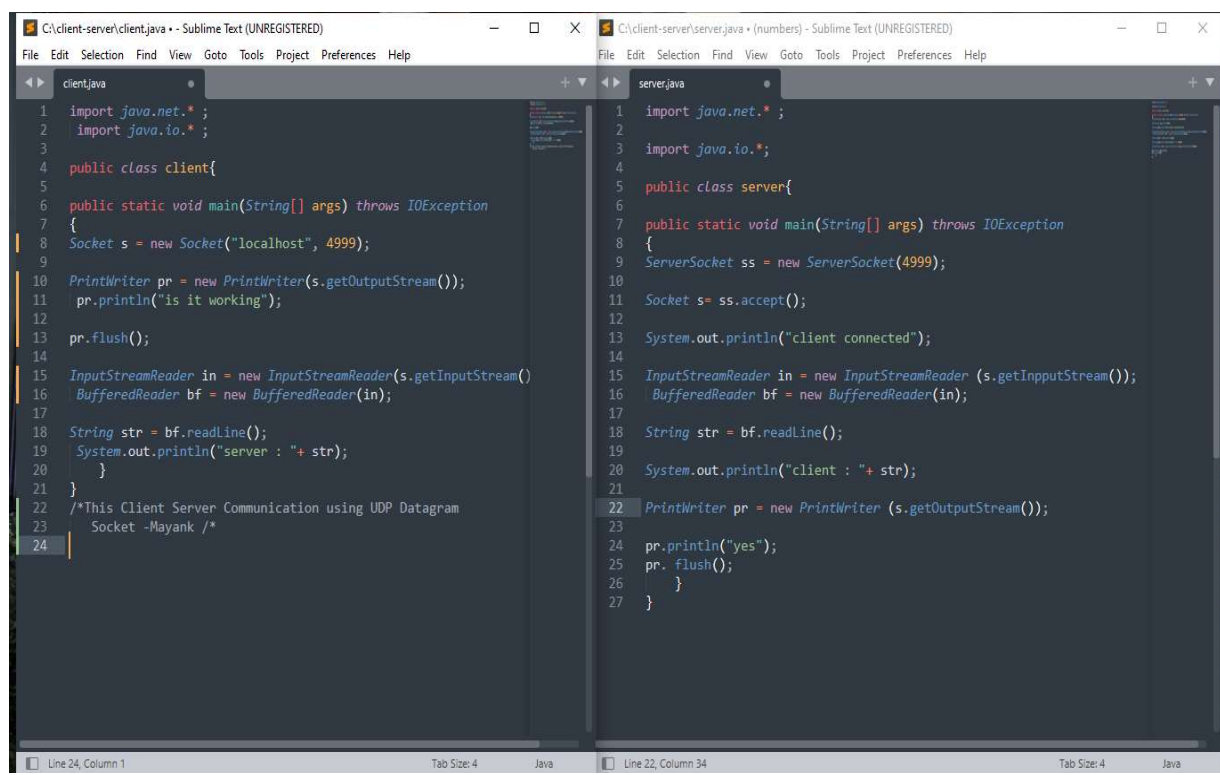
```
pr.println("yes");
```

```
pr. flush();
```

```
}
```

```
}
```

OUTPUT :



The screenshot displays two side-by-side Sublime Text editor windows. The left window, titled 'C:\client-server\client.java - Sublime Text (UNREGISTERED)', shows the client code. The right window, titled 'C:\client-server\server.java - (numbers) - Sublime Text (UNREGISTERED)', shows the server code. Both files are Java programs for a client-server communication using sockets.

```
client.java
1  import java.net.*;
2  import java.io.*;
3
4  public class client{
5
6  public static void main(String[] args) throws IOException
7  {
8  Socket s = new Socket("localhost", 4999);
9
10 PrintWriter pr = new PrintWriter(s.getOutputStream());
11 pr.println("is it working");
12
13 pr.flush();
14
15 InputStreamReader in = new InputStreamReader(s.getInputStream());
16 BufferedReader bf = new BufferedReader(in);
17
18 String str = bf.readLine();
19 System.out.println("server : "+ str);
20 }
21
22 /*This Client Server Communication using UDP Datagram
23 Socket -Mayank */
24
```

```
server.java
1  import java.net.*;
2
3  import java.io.*;
4
5  public class server{
6
7  public static void main(String[] args) throws IOException
8  {
9  ServerSocket ss = new ServerSocket(4999);
10
11 Socket s= ss.accept();
12
13 System.out.println("client connected");
14
15 InputStreamReader in = new InputStreamReader (s.getInputStream());
16 BufferedReader bf = new BufferedReader(in);
17
18 String str = bf.readLine();
19
20 System.out.println("client : "+ str);
21
22 PrintWriter pr = new PrintWriter (s.getOutputStream());
23
24 pr.println("yes");
25 pr. flush();
26 }
27 }
```

The status bar at the bottom indicates the current cursor positions: Line 24, Column 1 for client.java and Line 22, Column 34 for server.java.

The image shows two Sublime Text editor windows and a command prompt window. The left window, titled 'client.java', contains the following code:

```
1 import java.net.*;
2 import java.io.*;
3
4 public class client{
5
6     public static void main(String[] args) throws IOException
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

The right window, titled 'server.java (numbers)', contains the following code:

```
1 import java.net.*;
2
3 import java.io.*;
4
5 public class server{
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```

A command prompt window is open in the foreground, showing the following commands and output:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>cd ..
C:\Users>cd ..
C:\>cd client-server
C:\client-server>javac client.java
C:\client-server>java client
Exception in thread "main" java.net.ConnectException: Connection refused: connect
    at java.base/sun.nio.ch.Net.connect0(Native Method)
    at java.base/sun.nio.ch.Net.connect(Net.java:576)
    at java.base/sun.nio.ch.Net.connect(Net.java:565)
    at java.base/sun.nio.ch.NioSocketImpl.connect(NioSocketImpl.java:588)
    at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:333)
    at java.base/java.net.Socket.connect(Socket.java:645)
    at java.base/java.net.Socket.connect(Socket.java:595)
    at java.base/java.net.Socket.<init>(Socket.java:519)
    at java.base/java.net.Socket.<init>(Socket.java:293)
    at client.main(client.java:8)
C:\client-server>
```

The status bar at the bottom of the editors shows 'Line 21, Column 2' for the client.java editor and 'Line 15, Column 66' for the server.java editor.

2. Write a program to demonstrate status of key on Applet window such as KeyPressed, Key Released, KeyUp, KeyDown.

In one code all key commands in single a code

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class KeyEventDemo extends Applet
implements KeyListener
{
    String msg = "";

    @Override
    public void init()
    {
        addKeyListener(this);
    }

    @Override
    public void keyReleased(KeyEvent k)
    {
```

```
        showStatus("Key Released");  
        repaint();  
    }
```

```
/**  
 *  
 * @param k  
 */  
@Override  
public void keyTyped(KeyEvent k)  
{  
    showStatus("Key Typed");  
    repaint();  
}  
public void keyUp(KeyEvent k)  
{  
    showStatus("Key UP");  
    repaint();  
}  
public void keyDown(KeyEvent k)  
{  
    showStatus("Key Down");  
}
```

```

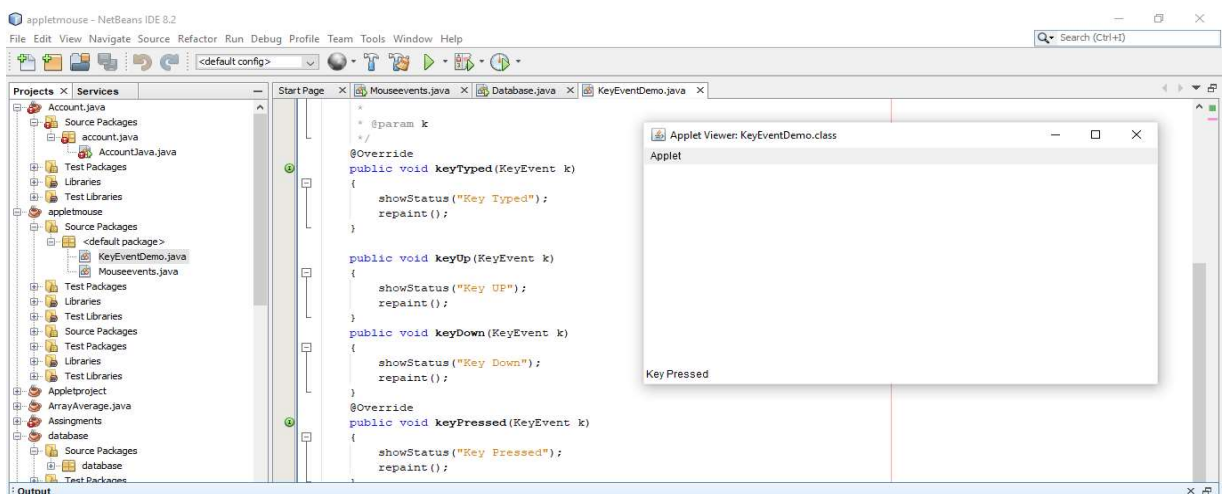
        repaint();
    }

    @Override
    public void keyPressed(KeyEvent k)
    {
        showStatus("Key Pressed");
        repaint();
    }

    @Override
    public void paint(Graphics g)
    {
        g.drawString(msg, 10, 10);
    }
}

```

OUTPUT:-



RUNNING KeyUp, KeyDown & KeyPressed, Key Released

SEPARATELY TO SEE THEIR WORKING :

For keypressed & key released using Applet:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class AppletKeyboard extends Applet
implements KeyListener
{
    TextField t,tt,tp,tr;
    @Override
    public void init()
    {
        t=new TextField(20);
        t.addKeyListener(this);
        tt=new TextField(70);
        tp=new TextField(70);
        tr=new TextField(70);
        add(t);
        add(tt);
        add(tr);
        add(tp);
    }
}
```

```

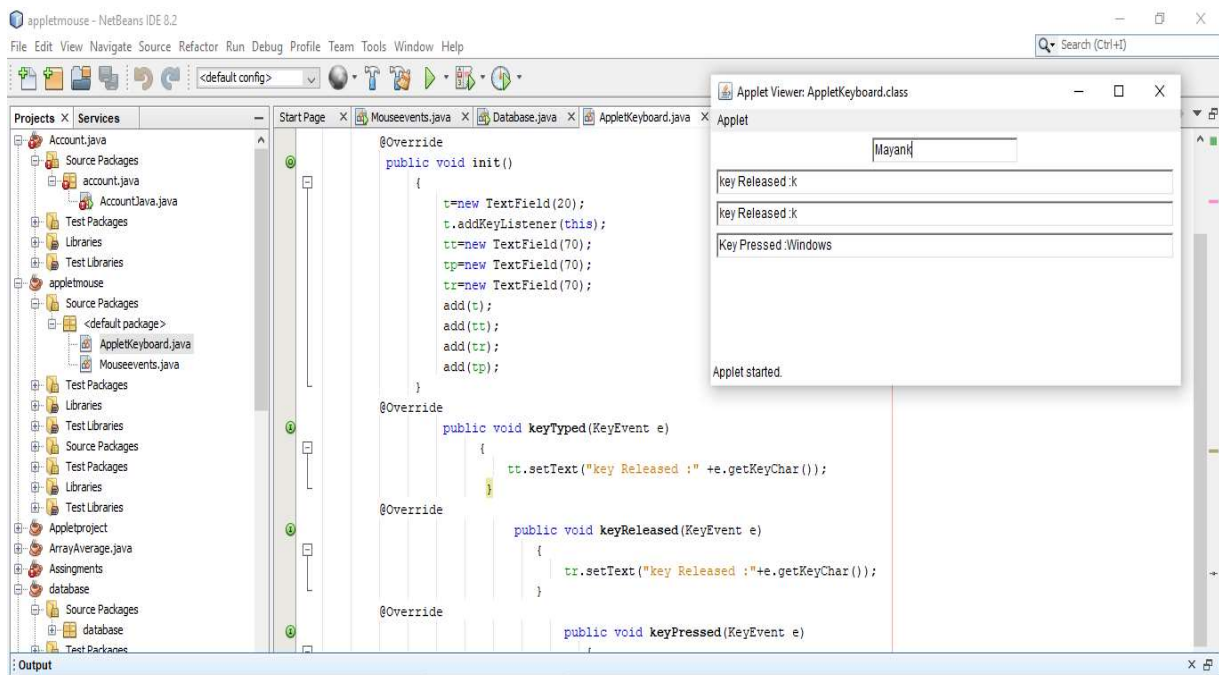
    }
    @Override
    public void keyTyped(KeyEvent e)
    {
        tt.setText("key Released :"+e.getKeyChar());
    }
    @Override
    public void keyReleased(KeyEvent e)
    {
        tr.setText("key Released :"+e.getKeyChar());
    }
    @Override
    public void keyPressed(KeyEvent e)
    {
        int kc;
        String s;

        kc=e.getKeyCode();
        s=KeyEvent.getKeyText(kc);
        tp.setText("Key Pressed :"+s);
    }
}

```

```
}  
  
}
```

OUTPUT :



For keyUp & keyDown in Applet :

```
import java.applet.Applet;
```

```
import java.awt.*;

public class key extends Applet{

    private Font f;

    private String letter;

    private boolean first;


    @Override

    public void init()

    {

        f = new Font( "Courier", Font.BOLD, 72 );

        first = true;

    }


    @Override

    public void paint( Graphics g )

    {

        g.setFont( f );
```

```
if ( !first )  
    g.drawString( letter, 75, 70 );  
}
```

```
@Override  
public boolean keyDown( Event e, int key )  
{  
    showStatus( "keyDown: the " + ( char ) key +  
        " was pressed." );  
  
    letter = String.valueOf( ( char ) key );  
    first = false;  
    repaint();  
  
    return true; // event has been handled  
}
```

@Override

```
public boolean keyUp( Event e, int key )
```

```
{
```

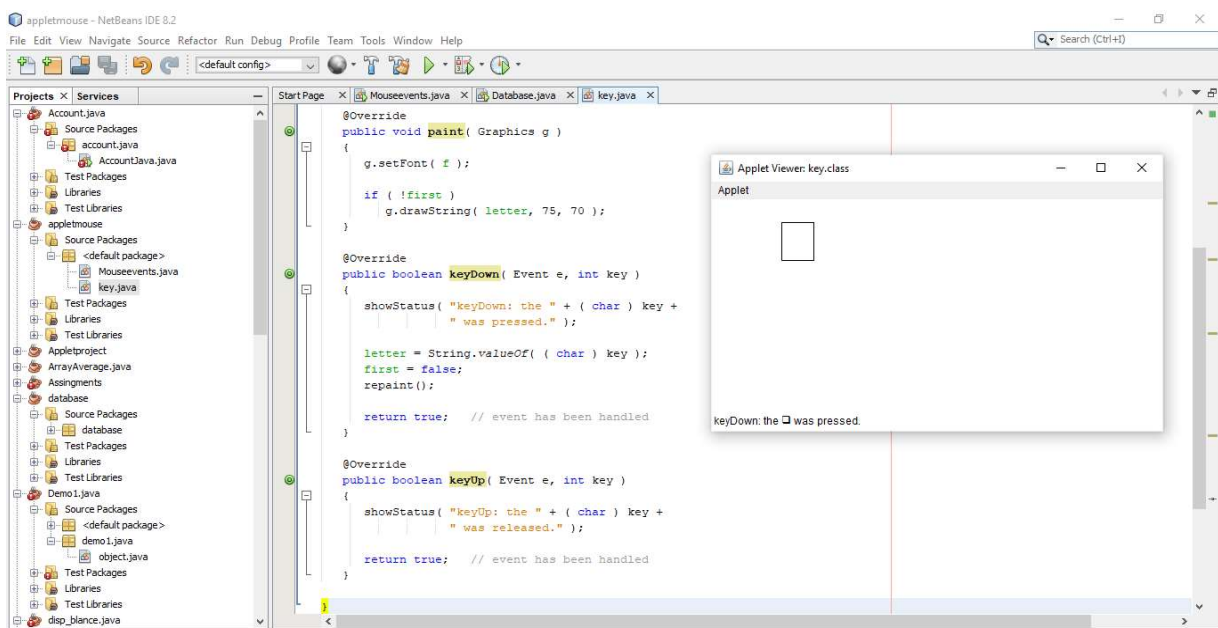
```
    showStatus( "keyUp: the " + ( char ) key +  
                " was released." );
```

```
    return true; // event has been handled
```

```
}
```

```
}
```

OUTPUT :



3. Write a java program to create a file with your name, save it in the desktop, write some data on the file and then read and print that data into the console.

```
import java.util.*;
import java.io.*;
class Rfile
{
    public static void main(String args[])throws
IOException
    {
        int j=1;
        char ch;
        Scanner scr=new Scanner(System.in);
        System.out.print("\nEnter File name: ");
        String str=scr.next();
        FileInputStream f=new FileInputStream(str);
```

```
System.out.println("\nContents of the file are");
```

```
    int n=f.available();
```

```
    System.out.print(j+": ");
```

```
    for(int i=0;i<n;i++)
```

```
    {
```

```
        ch=(char)f.read();
```

```
        System.out.print(ch);
```

```
        if(ch=='\n')
```

```
        {
```

```
            System.out.print(++j+": ");
```

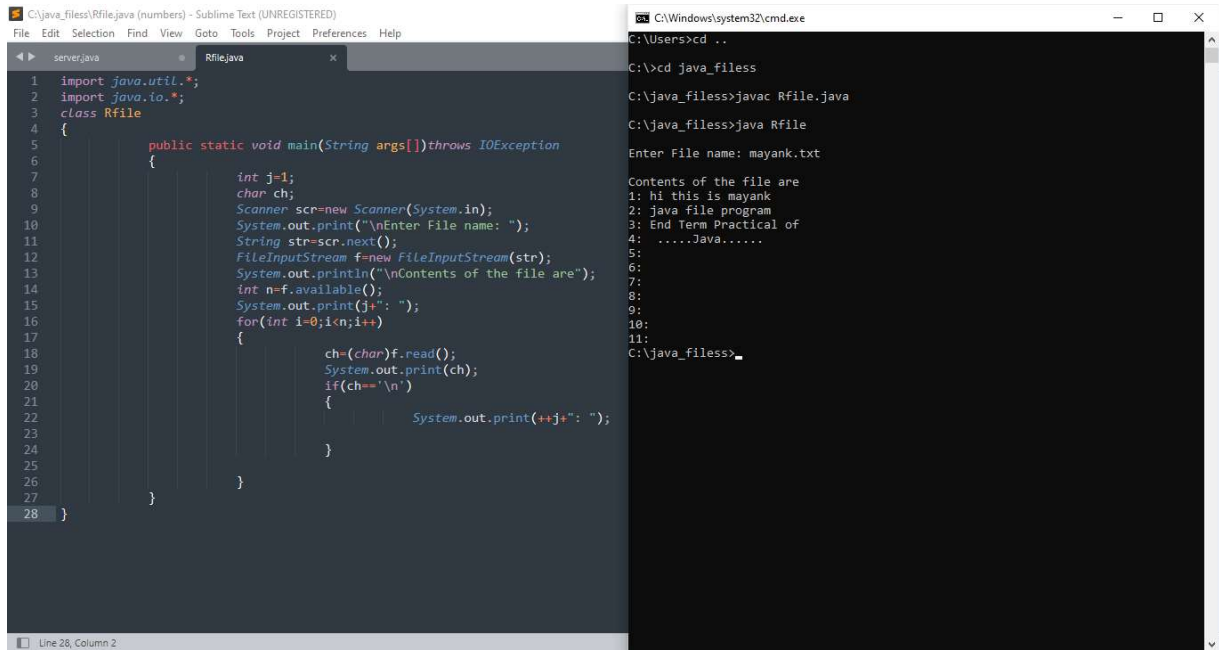
```
        }
```

```
    }
```

```
}
```

```
}
```


OUTPUT :



The image shows a side-by-side comparison of a Java source code file and its execution output. On the left, a Sublime Text editor window displays the code for a class named `Rfile`. The code imports `java.util.*` and `java.io.*`, then defines a `main` method that prompts the user for a file name, reads the file's contents line by line, and prints them. On the right, a Windows command prompt window shows the steps to compile and run the program. The user navigates to the directory containing the files, compiles `Rfile.java` using `javac`, and runs it using `java Rfile`. The program prompts for a file name, and the user enters `mayank.txt`. The output then displays the contents of the file, which are three lines of text.

```
1 import java.util.*;
2 import java.io.*;
3 class Rfile
4 {
5     public static void main(String args[])throws IOException
6     {
7         int j=1;
8         char ch;
9         Scanner scr=new Scanner(System.in);
10        System.out.print("\nEnter File name: ");
11        String str=scr.next();
12        FileInputStream f=new FileInputStream(str);
13        System.out.println("\nContents of the file are");
14        int n=f.available();
15        System.out.print(j+": ");
16        for(int i=0;i<n;i++)
17        {
18            ch=(char)f.read();
19            System.out.print(ch);
20            if(ch=='\n')
21            {
22                System.out.print(++j+": ");
23            }
24        }
25    }
26 }
27
28 }
```

```
C:\Users>cd ..
C:\>cd java_files
C:\java_files>javac Rfile.java
C:\java_files>java Rfile
Enter File name: mayank.txt
Contents of the file are
1: hi this is mayank
2: java file program
3: End Term Practical of
4: .....Java.....
5:
6:
7:
8:
9:
10:
11:
C:\java_files>
```