Exp: CREATE A SOCKET FOR HTTP FOR WEBPAGE UPLOAD AND DOWNLOAD

Objective:

To write a java program for creating socket for HTTP web page upload and download.

Learning Outcomes:

After the completion of this experiment, student will be able to

 Implement a program that can be used to understand the implementation of HTTP.

 Send HTTP request to a server and obtain the HTTP response message.

 Display the contents of the resolved file using HTTP server.

Problem Statement:

 A client program to get the file name from the user.

 A HTTP server program that resolves the given file and displays the contents of the file.

Concept: HTTP

1. The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the WWW.
2. HTTP functions as a combination of FTP and SMTP.
3. SMTP messages are stored and forwarded, but HTTP messages are delivered immediately.
4. The commands from the client to the server are embedded in a request message. The contents of the
requested file or other information are embedded in a response message.

System and Software tools required:

Package Required : Java Compiler

Operating System : UBUNTU

Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Algorithm:

Step1: Set a server port as 80.

Step2: Using HTTP services create a Socket for server by specifying the server port

Step3: Use HTTP socket for connecting the client to the URL.

Step4: Use BufferedReader to output stream to place the response from the server by the client.

Step5: Close the Connection as soon the request is been serviced. Use Malformed URL exception
If any errors in grabbing the server

Execution of program:

Compiling the program: javac file name.java

Executing the program : java file name

Sample Coding:

Client.java

```
/* …create file object…*/
importjava.io.File;
importjava.io.IOException;
/*…used to perform read and write operation…*/
importjavax.imageio.ImageIO;
public class Client{
public static void main(String args[]) throws Exception{
Socket soc;
BufferedImageimg = null;
soc=new Socket("localhost",4000);
System.out.println("Client is running. ");
try {
System.out.println("Reading image from disk. ");
/*…read image file…*/
img = ImageIO.read(new File("kalpanasonika.jpg"));
ByteArrayOutputStreambaos = new ByteArrayOutputStream();
/*…write image file…*/
ImageIO.write(img, "jpg", baos);
baos.flush();
/*…we use toByteArray() method of ByteArrayOutputStream class…*/
byte[] bytes = baos.toByteArray();
baos.close();
System.out.println("Sending image to server. ");
OutputStream out = soc.getOutputStream();
DataOutputStream dos = new DataOutputStream(out);
dos.writeInt(bytes.length);
dos.write(bytes, 0, bytes.length);
System.out.println("Image sent to server. ");
```

Server.java

```
//…Create Server Socket…//

ServerSocket server=null;
Socket socket;
```

//…Register Service port to 4000…//

server=new ServerSocket(4000);

System.out.println("Server Waiting for image");

socket=server.accept(); System.out.println("Client connected.");

InputStream in =socket.getInputStream();

DataInputStream dis = new DataInputStream(in);

intlen = dis.readInt();

System.out.println("Image Size: " + len/1024 + "KB");

byte[] data = new byte[len];

dis.readFully(data);

//…method is used to request for closing or terminating an object…//

dis.close();

in.close();

InputStreamian = new ByteArrayInputStream(data);

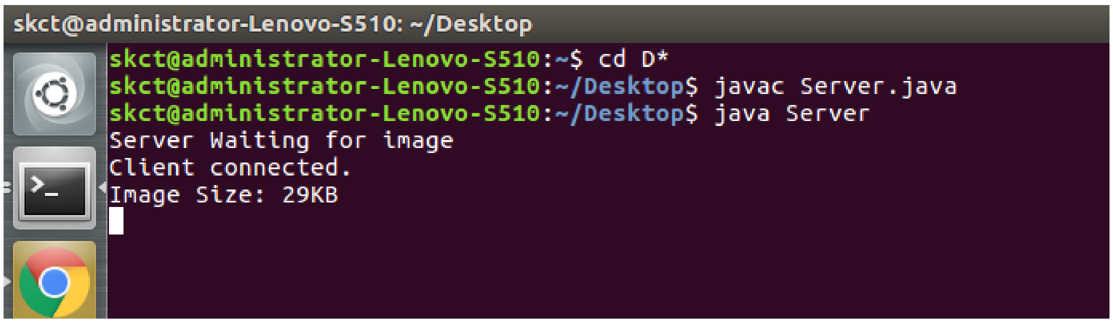
BufferedImagebImage = ImageIO.read(ian);

//…create a frame window entitled "server"…//

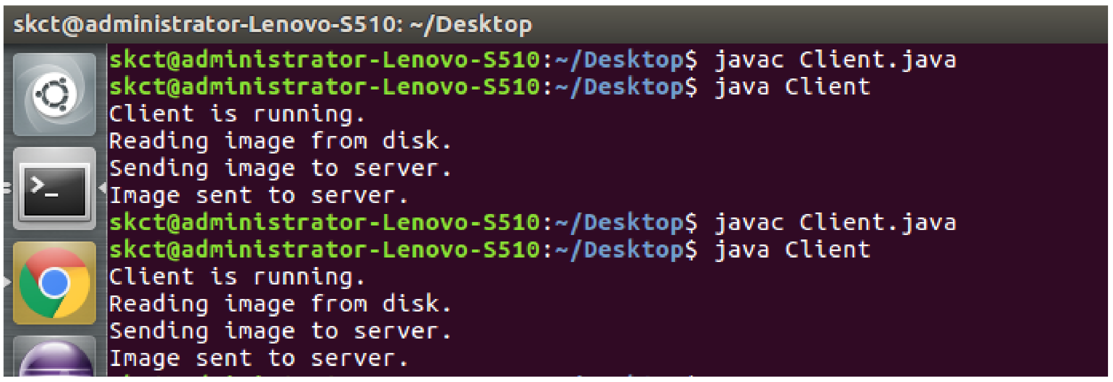
JFrame f = new JFrame("Server");

ImageIcon icon = new ImageIcon(bImage);

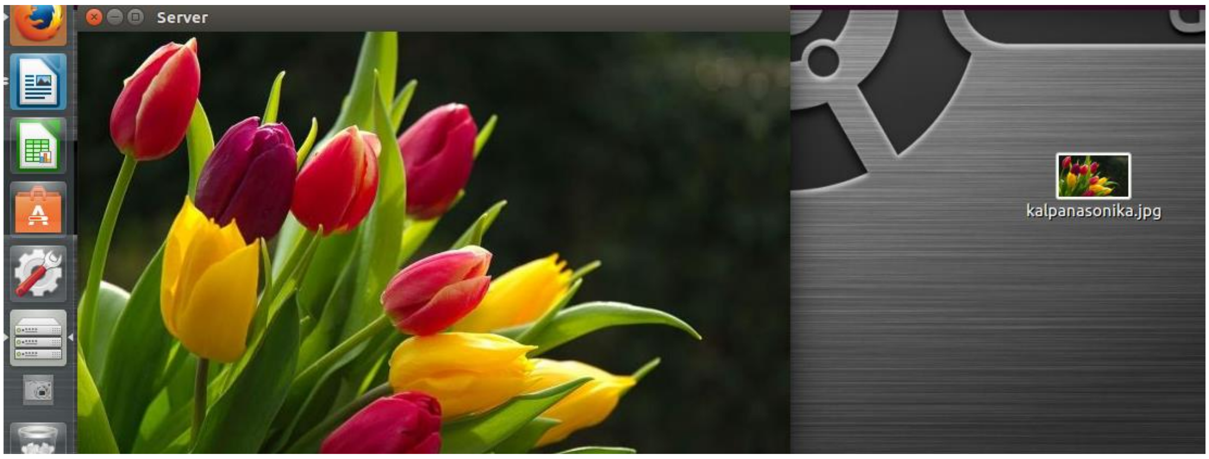
OUTPUT:



```
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~$ cd D*
skct@administrator-Lenovo-S510:~/Desktop$ javac Server.java
skct@administrator-Lenovo-S510:~/Desktop$ java Server
Server Waiting for image
Client connected.
Image Size: 29KB
```



```
skct@administrator-Lenovo-S510: ~/Desktop
skct@administrator-Lenovo-S510:~/Desktop$ javac Client.java
skct@administrator-Lenovo-S510:~/Desktop$ java Client
Client is running.
Reading image from disk.
Sending image to server.
Image sent to server.
skct@administrator-Lenovo-S510:~/Desktop$ javac Client.java
skct@administrator-Lenovo-S510:~/Desktop$ java Client
Client is running.
Reading image from disk.
Sending image to server.
Image sent to server.
```


Server
kalpanasonika.jpg