```java
/*server_socket - HTTP server implementation which runs as localhost onnport
'8059'.

package server;
import java.io.*;

public class server_socket extends Thread
{
    //declaration of socket, input & output streams
    BufferedReader server_input = null;
    DataOutputStream server_output = null;
    OutputStream output = null;
    Socket client_connect = null;

    //constructing a server response
    static String response_HTML_start = "<html>"+"<title>Java Server - To
Infinity And Beyond</title>"+"<body>";
    static String response_HTML_end = "</body>"+"</html>";

    //constructor called
    public server_socket(Socket client)
    {
        client_connect = client;
    }

    //start the server at the provided port number
    //handles up to 10 requests at the same time.
    public static void main(String[] args) throws Exception
    {
        ServerSocket server1 = new ServerSocket(8059, 10, InetAddress.getByName
("127.0.0.1"));
        System.out.println("SERVER: Waiting for client requests on port number
8059");

        //function to handle the incoming client request
        while (true)
        {
            Socket server1Socket = server1.accept();
            (new server_socket(server1Socket)).start();
        }
    }

    //function to handle the requests coming in
    public void run()
    {
        try
        {
            request_process();
        }

        catch(Exception e)
        {
            System.out.println(e);
        }
    }

    //function to handle the client request
    private void request_process()
    {
        try
        {
            //print: IP address & the port of client connected
            System.out.println("CONNECTED CLIENT: " +
client_connect.getInetAddress()+":"+client_connect.getPort());

            //initializing input & output streams for server communication
            server_input = new BufferedReader(new InputStreamReader
(client_connect.getInputStream()));
            server_output = new DataOutputStream(client_connect.getOutputStream
());

            //initializing variables with request header values
            //reading first line of the request
            String client_request = server_input.readLine();
            String header = client_request;
            System.out.println(header);
            String [] part = header.split(" ");
            String method_name = part[0];
            String file_path = part[1];

            //buffer to read the other header lines for client request
            StringBuffer response_buffer = new StringBuffer();
            response_buffer.append("<b> Server home page...</b><BR>");
            response_buffer.append("CLIENT REQUEST IS:<BR>");
            System.out.println("CLIENT REQUEST IS: ");
            while(server_input.ready())
            {
                response_buffer.append(client_request+"<BR>");
                System.out.println(client_request);
                client_request = server_input.readLine();
            }

            //function handling to call method in client request
            if(method_name.equals("GET"))
            {
                server1_get(file_path, response_buffer);
            }
            else if(method_name.equals("PUT"))
            {
                server1_put(file_path);
            }
            else
            {
                response_sender(991,"SORRY, Not implemented by the server
(method)",false);
            }
        }

        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    //function to handle GET requests from client
    private void server1_get(String path, StringBuffer bffr)
    {
        try
        {
            if(path.equals("/"))
            {
                response_sender(200,bffr.toString(),false);
            }
            else
            {
                String file = path.replaceFirst("/", "");
                file = URLDecoder.decode(file,"UTF-8");

                if(new File(file).isFile())
                {
                    response_sender(200,file,true);
                }
                else
                {
                    response_sender(404,"<b>RESOURCE NOT FOUND ON
SERVER",false);
                }
            }
        }

        catch(Exception e)
        {
            e.printStackTrace();
        }
```

```
}

//function to handle PUT requests coming from client
private void server1_put(String path) throws Exception
{
    String status=null;
    long cntntLength = 0;
    String cntntType = null;

    try
    {
        //extracting file name for the file to be created
        File temp = new File(path);
        String fl_name = temp.getName();

        //status code update
        if(temp.exists())
            status="200 OK";
        else
            status="201 Created";

        //creating a new file
        //writing the data coming over socket
        PrintWriter fout = null;
        File file = new File(fl_name);
        file.createNewFile();

        FileWriter fstream = new FileWriter(fl_name);
        BufferedWriter wrtr = new BufferedWriter(fstream);
        System.out.println("FILE_UPLOAD: "+fl_name);
        fout = new PrintWriter(fl_name);

        while(server_input.ready())
        {
            String crrntLine = server_input.readLine();
            System.out.println(crrntLine);
            wrtr.write(crrntLine+"\n");
        }
        wrtr.close();
        fout.close();
    }

    catch(Exception e)
    {
        e.printStackTrace();
        status = "500 Internal Server Error";
    }
```

```
        finally
        {
            cntntType = "Content-Type: text/html\r\n";
            cntntLength = status.length();
            snd_put_rspns(status,cntntType, cntntLength);
            output.close();
        }
    }

    //sending server response to client
    private void response_sender(int statusCode,String response,boolean isFile)
throws Exception
    {
        String status = null;
        String server1 = "Kiran's Homepage"+"\r\n";
        String cntntLength = null;
        String file = null;
        String cntntType = null;
        FileInputStream fIn = null;

        //status code update
        if(statusCode == 200)
            status = "HTTP/1.0 200 OK"+"\r\n";
        else if (statusCode == 991)
            status = "991 Method not implemented"+"\r\n";
        else
            status = "HTTP/1.0 404 Not Found"+"\r\n";

        //updating content headers
        if(isFile)
        {
            file = response;
            fIn = new FileInputStream(file);
            cntntLength = "Content-Length: "+Integer.toString(fIn.available())
+"\r\n";

            //file format
            if(file.endsWith(".htm") || file.endsWith("html"))
                cntntType = "Content-Type:text/html"+"\r\n";
            else if(file.endsWith(".jpg"))
                cntntType = "Content-Type:image/jpeg"+"\r\n";
            else if(file.endsWith(".txt"))
                cntntType = "Content-Type:text/plain"+"\r\n";
        }

        else
        {
            response = server_socket.response_HTML_start + response +
server_socket.response_HTML_end;
            cntntLength = "Content-Length: " + response.length() + "\r\n";
            cntntType = "Content-Type: "+"\r\n";
        }

        //sending header values of response over socket
        server_output.writeBytes(status);
        server_output.writeBytes(server1);
        server_output.writeBytes(cntntType);
        server_output.writeBytes(cntntLength);
        server_output.writeBytes("Connection: CLOSE\r\n");
        server_output.writeBytes("\r\n");

        //function to call file_send method
        if(isFile)
        {
            file_send(fIn, server_output);
        }
        else
        {
            server_output.writeBytes(response);
        }
        server_output.close();
    }

    //function for sending the client requested file
    private void file_send(FileInputStream file, DataOutputStream out) throws
Exception
    {
        int readBytes;
        byte[] bfr = new byte[2048];

        while((readBytes = file.read(bfr)) != -1)
        {
            out.write(bfr, 0, readBytes);
        }
        file.close();
    }

    //function to handle for sending the PUT request response
    private void snd_put_rspns(String st, String type,long len) throws Exception
    {
        output = client_connect.getOutputStream();
        output.write(("HTTP/1.0"+st+"\r\n").getBytes());
        output.write(("DATE: "+ new Date()+"\r\n").getBytes());
        output.write(("Content-Type:"+type+"\r\n").getBytes());
        output.write(("Content-Length: "+len+"\r\n").getBytes());
        output.write(("\r\n").getBytes());
    }
}
```