```java
package SingleThreadServer;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {

    public static void main(String[] args) {

        try (ServerSocket welcomingSocket = new ServerSocket(7657)) {
            System.out.print("Server started.\nWaiting for a client ... ");
            try (Socket connectionSocket = welcomingSocket.accept()) {
                System.out.println("client accepted!");
                OutputStream out = connectionSocket.getOutputStream();
                InputStream in = connectionSocket.getInputStream();
                byte[] buffer = new byte[2048];
                String[] messages = {"salam", "khubam!", "salamati!"};
                for (String msg: messages) {
                    int read = in.read(buffer);
                    System.out.println("RECV: " + new String(buffer, 0, read));
                    out.write(msg.getBytes());
                    System.out.println("SENT: " + msg);
                }
                System.out.print("All messages sent.\nClosing client ... ");
            } catch (IOException ex) {
                System.err.println(ex);
            }
            System.out.print("done.\nClosing server ... ");
        } catch (IOException ex) {
            System.err.println(ex);
        }
        System.out.println("done.");
    }
}

package SingleThreadServer;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;

public class Client {

    public static void main(String[] args) {
        try (Socket client = new Socket("127.0.0.1", 7657)) {
            System.out.println("Connected to server.");
            OutputStream out = client.getOutputStream();
            InputStream in = client.getInputStream();
            byte[] buffer = new byte[2048];
            String[] messages = {"salam", "chetori?", "che-khabar?"};
            for (String msg: messages) {
                out.write(msg.getBytes());
                System.out.println("SENT: " + msg);
                int read = in.read(buffer);
                System.out.println("RECV: " + new String(buffer, 0, read));
            }
            System.out.print("All messages sent.\nClosing ... ");
        } catch (IOException ex) {
            System.err.println(ex);
        }
        System.out.println("done.");
    }
}

// ================================================================================
```

```java
// Note: Client is the same as single thread example.

package MultiThreadServer;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Server {

    public static void main(String[] args) {
        ExecutorService pool = Executors.newCachedThreadPool();
        int count = 0;
        try (ServerSocket welcomingSocket = new ServerSocket(7660)) {
            System.out.print("Server started.\nWaiting for a client ... ");
            while (count < 3) {
                Socket connectionSocket = welcomingSocket.accept();
                count++;
                System.out.println("client accepted!");
                pool.execute(new ClientHandler(connectionSocket, count));
            }
            pool.shutdown();
            System.out.print("done.\nClosing server ... ");
        } catch (IOException ex) {
            System.err.println(ex);
        }
        System.out.println("done.");
    }

}

class ClientHandler implements Runnable {

    private Socket connectionSocket;
    private int clientNum;

    public ClientHandler(Socket connectionSocket, int clientNum) {
        this.connectionSocket = connectionSocket;
        this.clientNum=clientNum;
    }

    @Override
    public void run() {
        try {
            OutputStream out = connectionSocket.getOutputStream();
            InputStream in = connectionSocket.getInputStream();
            byte[] buffer = new byte[2048];
            String[] messages = {"salam", "khubam!", "salamati!"};
            for (String msg: messages) {
                int read = in.read(buffer);
                System.out.println("RECV from "+clientNum+": " + new String(buffer, 0,
                read));
                out.write(msg.getBytes());
                System.out.println("SENT to "+clientNum+": " + msg);
                Thread.sleep(2000);
            }
            System.out.print("All messages sent.\nClosing client ... ");
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        } finally {
            try {
                connectionSocket.close();
            } catch (IOException ex) {
```

```
                System.err.println(ex);
            }
        }
    }
}

// ============================================================================

import javax.net.ssl.HttpsURLConnection;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class HttpDownloader implements Runnable{
    private URL url;
    private String directory;
    private String targetFileName;

    public HttpDownloader(String url, String targetFileName) throws
    MalformedURLException {
        this.url = new URL(url);
        this.targetFileName=targetFileName;
        directory = System.getProperty("user.home") +
                    File.separator + "Downloads" + File.separator;
    }

    private String getFileName() {
        return this.targetFileName;
    }

    @Override
    public void run() {
        System.out.printf("Starting Download:\n\t%s\n\t%s\n", url.getPath(), directory);
        HttpURLConnection connection;
        try {
            if ("http".equals(url.getProtocol())) {
                connection = (HttpURLConnection) url.openConnection();
            } else if ("https".equals(url.getProtocol())) {
                connection = (HttpsURLConnection) url.openConnection();
            } else {
                System.err.println("UNSUPPORTED PROTOCOL!");
                return;
            }
            connection.connect();
            // Make sure response code is in the 200 range.

            if (connection.getResponseCode() / 100 != 2)
                throw new IOException(connection.getResponseCode() + connection.
                getResponseMessage());
        } catch (IOException ex) {
            System.err.println("FAILED TO OPEN CONNECTION!" + ex);
            return;
        }

        File file = new File(directory + getFileName());
        long contentLength = connection.getContentLengthLong();
        System.out.println("Content Length = " + contentLength+" bytes.");

        try(InputStream in = connection.getInputStream();
            FileOutputStream out = new FileOutputStream(file)) {
            int totalRead = 0;
            byte[] buffer = new byte[1000000];
            while (totalRead < contentLength) {
                int read = in.read(buffer);
```

```
                if (read == -1)
                    break;
                out.write(buffer, 0, read);
                totalRead += read;
                System.out.println("Downloading>>total read is "+totalRead+" bytes.");
            }
            System.out.println("Download finished!\nTotal Read = " + totalRead);
        } catch (IOException ex) {
            System.err.println("");
        }
    }

    public static void main(String[] args) {
        ExecutorService executor= Executors.newCachedThreadPool();
        Scanner in =new Scanner(System.in);
        do {
            try {
                System.out.println("Enter the link for download file:");
                String linkURL=in.nextLine();
                System.out.println("Enter the target file name:");
                String targetFileName=in.nextLine();
                executor.execute(new HttpDownloader(linkURL,targetFileName));
                executor.awaitTermination(5, TimeUnit.MINUTES);
            } catch (MalformedURLException | InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            System.out.println("Do you have new link for download? (yes or no)");
        }while (!in.nextLine().equals("no"));
        executor.shutdown();
    }
}
```