

```

1  package SingleThreadServer;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.io.OutputStream;
6  import java.net.ServerSocket;
7  import java.net.Socket;
8
9  public class Server {
10
11      public static void main(String[] args) {
12
13          try (ServerSocket welcomingSocket = new ServerSocket(7657)) {
14              System.out.print("Server started.\nWaiting for a client ... ");
15              try (Socket connectionSocket = welcomingSocket.accept()) {
16                  System.out.println("client accepted!");
17                  OutputStream out = connectionSocket.getOutputStream();
18                  InputStream in = connectionSocket.getInputStream();
19                  byte[] buffer = new byte[2048];
20                  String[] messages = {"salam", "khubam!", "salamati!"};
21                  for (String msg: messages) {
22                      int read = in.read(buffer);
23                      System.out.println("RECV: " + new String(buffer, 0, read));
24                      out.write(msg.getBytes());
25                      System.out.println("SENT: " + msg);
26                  }
27                  System.out.print("All messages sent.\nClosing client ... ");
28              } catch (IOException ex) {
29                  System.err.println(ex);
30              }
31              System.out.print("done.\nClosing server ... ");
32          } catch (IOException ex) {
33              System.err.println(ex);
34          }
35          System.out.println("done.");
36      }
37  }
38
39  package SingleThreadServer;
40
41  import java.io.IOException;
42  import java.io.InputStream;
43  import java.io.OutputStream;
44  import java.net.Socket;
45
46  public class Client {
47
48      public static void main(String[] args) {
49          try (Socket client = new Socket("127.0.0.1", 7657)) {
50              System.out.println("Connected to server.");
51              OutputStream out = client.getOutputStream();
52              InputStream in = client.getInputStream();
53              byte[] buffer = new byte[2048];
54              String[] messages = {"salam", "chetori?", "che-khabar?"};
55              for (String msg: messages) {
56                  out.write(msg.getBytes());
57                  System.out.println("SENT: " + msg);
58                  int read = in.read(buffer);
59                  System.out.println("RECV: " + new String(buffer, 0, read));
60              }
61              System.out.print("All messages sent.\nClosing ... ");
62          } catch (IOException ex) {
63              System.err.println(ex);
64          }
65          System.out.println("done.");
66      }
67  }
68
69  // =====

```

```

70 // Note: Client is the same as single thread example.
71
72 package MultiThreadServer;
73
74 import java.io.IOException;
75 import java.io.InputStream;
76 import java.io.OutputStream;
77 import java.net.ServerSocket;
78 import java.net.Socket;
79 import java.util.concurrent.ExecutorService;
80 import java.util.concurrent.Executors;
81
82 public class Server {
83
84     public static void main(String[] args) {
85         ExecutorService pool = Executors.newCachedThreadPool();
86         int count = 0;
87         try (ServerSocket welcomingSocket = new ServerSocket(7660)) {
88             System.out.print("Server started.\nWaiting for a client ... ");
89             while (count < 3) {
90                 Socket connectionSocket = welcomingSocket.accept();
91                 count++;
92                 System.out.println("client accepted!");
93                 pool.execute(new ClientHandler(connectionSocket, count));
94             }
95             pool.shutdown();
96             System.out.print("done.\nClosing server ... ");
97         } catch (IOException ex) {
98             System.err.println(ex);
99         }
100         System.out.println("done.");
101     }
102
103 }
104
105 class ClientHandler implements Runnable {
106
107     private Socket connectionSocket;
108     private int clientNum;
109
110     public ClientHandler(Socket connectionSocket, int clientNum) {
111         this.connectionSocket = connectionSocket;
112         this.clientNum = clientNum;
113     }
114
115     @Override
116     public void run() {
117         try {
118             OutputStream out = connectionSocket.getOutputStream();
119             InputStream in = connectionSocket.getInputStream();
120             byte[] buffer = new byte[2048];
121             String[] messages = {"salam", "khubam!", "salamati!"};
122             for (String msg: messages) {
123                 int read = in.read(buffer);
124                 System.out.println("RECV from "+clientNum+": " + new String(buffer, 0,
125                                     read));
126                 out.write(msg.getBytes());
127                 System.out.println("SENT to "+clientNum+": " + msg);
128                 Thread.sleep(2000);
129             }
130             System.out.print("All messages sent.\nClosing client ... ");
131         } catch (IOException e) {
132             e.printStackTrace();
133         } catch (InterruptedException e) {
134             e.printStackTrace();
135         } finally {
136             try {
137                 connectionSocket.close();
138             } catch (IOException ex) {

```

```

138         System.err.println(ex);
139     }
140 }
141 }
142 }
143
144 // =====
145
146 import javax.net.ssl.HttpURLConnection;
147 import java.io.*;
148 import java.net.HttpURLConnection;
149 import java.net.MalformedURLException;
150 import java.net.URL;
151 import java.util.Scanner;
152 import java.util.concurrent.ExecutorService;
153 import java.util.concurrent.Executors;
154 import java.util.concurrent.TimeUnit;
155
156 public class HttpDownloader implements Runnable{
157     private URL url;
158     private String directory;
159     private String targetFileName;
160
161     public HttpDownloader(String url, String targetFileName) throws
162     MalformedURLException {
163         this.url = new URL(url);
164         this.targetFileName=targetFileName;
165         directory = System.getProperty("user.home") +
166             File.separator + "Downloads" + File.separator;
167     }
168
169     private String getFileName() {
170         return this.targetFileName;
171     }
172
173     @Override
174     public void run() {
175         System.out.printf("Starting Download:\n\t%s\n\t%s\n", url.getPath(), directory);
176         HttpURLConnection connection;
177         try {
178             if ("http".equals(url.getProtocol())) {
179                 connection = (HttpURLConnection) url.openConnection();
180             } else if ("https".equals(url.getProtocol())) {
181                 connection = (HttpsURLConnection) url.openConnection();
182             } else {
183                 System.err.println("UNSUPPORTED PROTOCOL!");
184                 return;
185             }
186             connection.connect();
187             // Make sure response code is in the 200 range.
188             if (connection.getResponseCode() / 100 != 2)
189                 throw new IOException(connection.getResponseCode() + connection.
190                     getResponseMessage());
191         } catch (IOException ex) {
192             System.err.println("FAILED TO OPEN CONNECTION!" + ex);
193             return;
194         }
195
196         File file = new File(directory + getFileName());
197         long contentLength = connection.getContentLengthLong();
198         System.out.println("Content Length = " + contentLength+" bytes.");
199
200         try(InputStream in = connection.getInputStream();
201             FileOutputStream out = new FileOutputStream(file)) {
202             int totalRead = 0;
203             byte[] buffer = new byte[1000000];
204             while (totalRead < contentLength) {
205                 int read = in.read(buffer);

```

```

205         if (read == -1)
206             break;
207         out.write(buffer, 0, read);
208         totalRead += read;
209         System.out.println("Downloading>>total read is "+totalRead+" bytes.");
210     }
211     System.out.println("Download finished!\nTotal Read = " + totalRead);
212 } catch (IOException ex) {
213     System.err.println("");
214 }
215 }
216
217 public static void main(String[] args) {
218     ExecutorService executor= Executors.newCachedThreadPool();
219     Scanner in =new Scanner(System.in);
220     do {
221         try {
222             System.out.println("Enter the link for download file:");
223             String linkURL=in.nextLine();
224             System.out.println("Enter the target file name:");
225             String targetFileName=in.nextLine();
226             executor.execute(new HttpDownloader(linkURL,targetFileName));
227             executor.awaitTermination(5, TimeUnit.MINUTES);
228         } catch (MalformedURLException | InterruptedException e) {
229             // TODO Auto-generated catch block
230             e.printStackTrace();
231         }
232         System.out.println("Do you have new link for download? (yes or no)");
233     }while (!in.nextLine().equals("no"));
234     executor.shutdown();
235 }
236 }
237

```