Programming Assignment #2

Team Members:
Ethan Chan, 84913919, ethanchan@ufl.edu
Al Fahad, 21192006, afahad1@ufl.edu
Bryan Smith 71305927, bzane.smith@ufl.edu

Compilation Instructions:

Before you compile, make sure you have the latest JDK downloaded

Latest Java JDK https://www.oracle.com/java/technologies/downloads/#jdk21-windows

To compile the code, first download the code from git using

git clone https://github.com/BryanZaneee/UDP-TCP

Then,

cd to the repository

Run **javac TCPclient.java** and **TCPjavac server.java** to compile the two TCP files

```
C:\Users\Ethan\Desktop\pa2>javac TCPclient.java
```

```
C:\Users\Ethan\Desktop\pa2>javac TCPserver.java
```

Run **javac UDPclient.java** and **javac UDPserver.java** to compile the two UDP files

```
C:\Users\Ethan\Desktop\pa2>javac UDPclient.java
```

```
C:\Users\Ethan\Desktop\pa2>javac UDPserver.java
```

To run the compiled java program

Open two terminals, one for the client and the other for the server

Run commands **java TCPserver** and **java TCPclient** in each of the terminal

Replace **TCP** with **UDP** to run the UDP version

```
C:\Users\Ethan\Desktop\pa2>java TCPserver 12345
```

```
C:\Users\Ethan\Desktop\pa2>java TCPclient localhost 12345
```

Note: Replace 12345 with your port and localhost with your host ip

Code Structure Description:

TCPclient.Java

1. **Argument verification:** Code starts by verifying if the user has provided necessary arguments
2. **Connection Setup:** It attempts to establish a TCP connection to the server using the provided address and port, also sets up data streams for input and output over the connection
3. **Communication:**
    a. Client enters a loop where it sends a randomly selected number which represents a meme number to the server
    b. It measures the round-trip-time for each communication, which is the time taken from sending the request to receiving the response
    c. Client expects to receive binary data, a meme picture, from the server, which it saves to a file. Print a message based on whether it was received.
4. **Closing the connection:** After completing the loop, the client sends a bye message to the server and flushes the output stream to ensure the message is sent before closing the connection
5. **Statistics Calculation:** client calculates and prints statistics (minimum, maximum, mean, and standard deviation) of the round-trip times measured during the communication with the serve
6. **Error Handling:** The code includes basic error handling for unknown host exceptions and I/O exceptions, printing relevant error messages to the user.
7. **Utility Methods:** The program includes four private static methods (`min`, `max`, `mean`, and `stdDev`) for computing the minimum, maximum, mean, and standard deviation of an array of long integers.

UDPclient.Java

1. **Constants Definition:**
    a. `SERVER_ADDRESS` and `SERVER_PORT` are defined to specify the server's address and port.
2. **Main Method:**
    a. Initializes a `DatagramSocket` for communication.
    b. Converts the server address from a `String` to an `InetAddress`.
    c. Prints a connection message.
    d. Initializes arrays to store measurement data for round trip times and DNS lookup times.
3. **Request Loop:**

    a. Iterates 10 times, sending a request for a random meme number between 1 and 10.
    b. Measures the time taken for DNS lookup by re-resolving the server address each time (not typically necessary in a real application).
    c. Constructs a UDP packet with the request and sends it to the server.
    d. Waits to receive a response from the server and calculates the round trip time.
    e. Parses and prints the response.
4. **Closing Command:**
    a. Sends a "bye" command to the server to signal the end of the session.
    b. Receives the server's acknowledgment and exits if the response is "disconnected".
5. **Statistics Computation:**
    a. After the loop, calculates minimum, maximum, mean, and standard deviation for both round trip times and DNS lookup times.
    b. Prints these statistics.
6. **Helper Methods:**
    a. `min`, `max`, `mean`, and `stdDev` are implemented to calculate statistics.

TCPserver.java

1. **Port Definition:**
    - The PORT constant defines the port number on which the server listens for incoming connections.
2. **Main Method:**
    - Initializes a ServerSocket to listen for incoming connections on the defined port.
    - Uses a while loop to continuously accept new client connections. For each connection, the server prints a log message and starts a new thread (ServerThread) to handle the connection, allowing for multiple concurrent client connections.
3. **ServerThread Class:**
    - An inner class that extends Thread, designed to handle individual client connections.
    - Holds a reference to the client's socket.
    - Overrides the run method to implement the logic for interacting with the client over the socket.
4. **Client Interaction:**
    - Uses DataInputStream and DataOutputStream to read from and write to the client's socket stream, respectively.
    - Continuously reads messages (expected to be integers as text) from the client until a "bye" message is received, indicating the client wishes to disconnect.
    - For each received message, it attempts to parse it as an integer to determine which file (meme image) the client requests.

    - Checks if the requested file exists. If it does, it reads the file into a byte array and sends the file name followed by the file size and content to the client. If the file doesn't exist, it sends a "File not found" message.
5. **File Handling:**
    - Prepares the path for the requested file based on the received message.
    - Uses FileInputStream to read the requested file into a byte array if it exists.
6. **Logging and Resource Management:**
    - Prints log messages to the console for significant events, such as client connections, disconnections, and file transmission details.
    - Ensures resources like socket streams and the client socket itself are properly closed in a finally block to avoid resource leaks.

UDPserver.java

1. **Port Definition:**
    - `PORT` is a constant that defines the port number on which the server listens for incoming UDP packets.
2. **Main Method:**
    - Initializes a `DatagramSocket` on the specified port to listen for incoming packets.
    - Prints a message indicating that the server is listening.
3. **Listening Loop:**
    - The server enters an infinite loop, continuously listening for and processing incoming packets.
    - Each packet is expected to contain a request in the form of a text message, which could be either a number representing a specific meme image or the command "bye" to indicate disconnection.
4. **Packet Reception and Processing:**
    - A `DatagramPacket` for receiving data is created with a pre-defined byte array size.
    - The server waits to receive a packet and then extracts the text from the received packet.
    - The server determines the client's address and port number from the received packet, which are used for sending the response.
5. **Command Handling:**
    - If the received text is "bye", the server sends a "disconnected" response back to the client and continues listening for more requests.
    - For other text, the server attempts to parse it as an integer representing a meme number.
6. **File Handling:**

    - If the text can be parsed as a number, the server constructs a file path based on that number and checks if the corresponding file exists.
    - If the file exists, it reads the file into a byte array and sends it back to the client as a UDP packet.
    - If the file does not exist or if the input is invalid, the server sends an appropriate error message back to the client.
7. **Error Handling:**
    - Catches `IOException` for any input/output errors during operation.
    - Handles `NumberFormatException` when parsing the request to ensure the server can respond to invalid input without crashing.

Experiment Results:

Experiment #1: Done on Windows 11

TCP:

```
C:\Users\Ethan\Desktop\pa2>java TCPclient localhost 12345
Client> Connected to the server.
Client> Received meme: received_meme_4.jpg
Client> Received meme: received_meme_7.jpg
Client> Received meme: received_meme_7.jpg
Client> Received meme: received_meme_5.jpg
Client> Received meme: received_meme_5.jpg
Client> Received meme: received_meme_5.jpg
Client> Received meme: received_meme_8.jpg
Client> Received meme: received_meme_9.jpg
Client> Received meme: received_meme_6.jpg
Client> Received meme: received_meme_7.jpg

Round Trip Time Statistics (in nanoseconds):
Minimum: 353400
Maximum: 4620200
Mean: 889850.0
Standard Deviation: 1262077.2696233776
```

```
C:\Users\Ethan\Desktop\pa2>java TCPserver 12345
Server is listening on port 12345
Server> Got connection request from 127.0.0.1
Server> File sent: memes/meme4.jpg. Access time: 750900 ns.
Server> File sent: memes/meme7.jpg. Access time: 391600 ns.
Server> File sent: memes/meme7.jpg. Access time: 405900 ns.
Server> File sent: memes/meme5.jpg. Access time: 404400 ns.
Server> File sent: memes/meme5.jpg. Access time: 433000 ns.
Server> File sent: memes/meme5.jpg. Access time: 321200 ns.
Server> File sent: memes/meme8.jpg. Access time: 326100 ns.
Server> File sent: memes/meme9.jpg. Access time: 434300 ns.
Server> File sent: memes/meme6.jpg. Access time: 297600 ns.
Server> File sent: memes/meme7.jpg. Access time: 302500 ns.
Server> Client disconnected.
```

UDP:

```
C:\Users\Ethan\Desktop\pa2>java UDPserver 12345
Server is listening on port 12345
Server> Client requested: "Meme 2", returning: "memes/meme2.jpg" file
Server> Client requested: "Meme 8", returning: "memes/meme8.jpg" file
Server> Client requested: "Meme 3", returning: "memes/meme3.jpg" file
Server> Client requested: "Meme 1", returning: "memes/meme1.jpg" file
Server> Client requested: "Meme 1", returning: "memes/meme1.jpg" file
Server> Client requested: "Meme 3", returning: "memes/meme3.jpg" file
Server> Client requested: "Meme 10", returning: "memes/meme10.jpg" file
Server> Client requested: "Meme 4", returning: "memes/meme4.jpg" file
Server> Client requested: "Meme 8", returning: "memes/meme8.jpg" file
Server> Client requested: "Meme 4", returning: "memes/meme4.jpg" file
Server> Client disconnected.
```

```
Client> exit

Round Trip Time Statistics (in nanoseconds):
Minimum: 398100
Maximum: 2960100
Mean: 751220.0
Standard Deviation: 745029.3427778532

DNS Lookup Time Statistics (in nanoseconds):
Minimum: 7100
Maximum: 15100
Mean: 9840.0
Standard Deviation: 2755.4310007692084
```

Experiment #2: Server hosted on Windows 11 and client hosted on MacOS

TCP:

```
C:\Users\Ethan\Desktop\pa2>java TCPserver 12345
Server is listening on port 12345
Server> Got connection request from 10.3.6.110
Server> File sent: memes/meme9.jpg. Access time: 749500 ns.
Server> File sent: memes/meme5.jpg. Access time: 408000 ns.
Server> File sent: memes/meme1.jpg. Access time: 1202800 ns.
Server> File sent: memes/meme6.jpg. Access time: 335800 ns.
Server> File sent: memes/meme2.jpg. Access time: 334400 ns.
Server> File sent: memes/meme9.jpg. Access time: 300200 ns.
Server> File sent: memes/meme7.jpg. Access time: 446200 ns.
Server> File sent: memes/meme8.jpg. Access time: 438000 ns.
Server> File sent: memes/meme10.jpg. Access time: 348600 ns.
Server> File sent: memes/meme7.jpg. Access time: 337000 ns.
Server> Client disconnected.
```

```
ethans-MacBook-Pro:pa2 ethanchan$ java TCPclient 10.20.81.163 12345
Client> Connected to the server.
Client> Received meme: received_meme_9.jpg
Client> Received meme: received_meme_5.jpg
Client> Received meme: received_meme_1.jpg
Client> Received meme: received_meme_6.jpg
Client> Received meme: received_meme_2.jpg
Client> Received meme: received_meme_9.jpg
Client> Received meme: received_meme_7.jpg
Client> Received meme: received_meme_8.jpg
Client> Received meme: received_meme_10.jpg
Client> Received meme: received_meme_7.jpg

Round Trip Time Statistics (in nanoseconds):
Minimum: 1644105
Maximum: 220819940
Mean: 2.62137517E7
Standard Deviation: 6.49052333265678 66E7
ethans-MacBook-Pro:pa2 ethanchan$
```

UDP:

```
C:\Users\Ethan\Desktop\pa2>java UDPserver 12345
Server is listening on port 12345
Server> Client requested: "Meme 4", returning: "memes/meme4.jpg" file
Server> Client requested: "Meme 7", returning: "memes/meme7.jpg" file
Server> Client requested: "Meme 2", returning: "memes/meme2.jpg" file
Server> Client requested: "Meme 1", returning: "memes/meme1.jpg" file
Server> Client requested: "Meme 9", returning: "memes/meme9.jpg" file
Server> Client requested: "Meme 6", returning: "memes/meme6.jpg" file
Server> Client requested: "Meme 3", returning: "memes/meme3.jpg" file
Server> Client requested: "Meme 1", returning: "memes/meme1.jpg" file
Server> Client requested: "Meme 3", returning: "memes/meme3.jpg" file
Server> Client requested: "Meme 8", returning: "memes/meme8.jpg" file
Server> Client disconnected.
```

```
Client> exit

Round Trip Time Statistics (in nanoseconds):
Minimum: 4309576
Maximum: 7864202
Mean: 5396865.9
Standard Deviation: 1046266.5365220709

DNS Lookup Time Statistics (in nanoseconds):
Minimum: 8644
Maximum: 23496
Mean: 13807.7
Standard Deviation: 4357.555072514861
ethans-MacBook-Pro:pa2 ethanchan$
```

Experiment #3: Server hosted on Windows 11 and client hosted on MacOS

TCP:

```
ethans-MacBook-Pro:pa2 ethanchan$ java TCPclient 10.20.81.163 12345
Client> Connected to the server.
Client> Received meme: received_meme_6.jpg
Client> Received meme: received_meme_4.jpg
Client> Received meme: received_meme_10.jpg
Client> Received meme: received_meme_2.jpg
Client> Received meme: received_meme_6.jpg
Client> Received meme: received_meme_7.jpg
Client> Received meme: received_meme_9.jpg
Client> Received meme: received_meme_5.jpg
Client> Received meme: received_meme_2.jpg
Client> Received meme: received_meme_8.jpg

Round Trip Time Statistics (in nanoseconds):
Minimum: 2616849
Maximum: 9435137
Mean: 4144542.1
Standard Deviation: 1880945.3902782747
ethans-MacBook-Pro:pa2 ethanchan$
```

```
ethans-MacBook-Pro:pa2 ethanchan$ java TCPclient 10.20.81.163 12345
Client> Connected to the server.
Client> Received meme: received_meme_9.jpg
Client> Received meme: received_meme_5.jpg
Client> Received meme: received_meme_1.jpg
Client> Received meme: received_meme_6.jpg
Client> Received meme: received_meme_2.jpg
Client> Received meme: received_meme_9.jpg
Client> Received meme: received_meme_7.jpg
Client> Received meme: received_meme_8.jpg
Client> Received meme: received_meme_10.jpg
Client> Received meme: received_meme_7.jpg

Round Trip Time Statistics (in nanoseconds):
Minimum: 1644105
Maximum: 220819940
Mean: 2.62137517E7
Standard Deviation: 6.49052333265678 66E7
ethans-MacBook-Pro:pa2 ethanchan$
```

UDP:

```
C:\Users\Ethan\Desktop\pa2>java UDPserver 12345
Server is listening on port 12345
Server> Client requested: "Meme 9", returning: "memes/meme9.jpg" file
Server> Client requested: "Meme 4", returning: "memes/meme4.jpg" file
Server> Client requested: "Meme 7", returning: "memes/meme7.jpg" file
Server> Client requested: "Meme 2", returning: "memes/meme2.jpg" file
Server> Client requested: "Meme 3", returning: "memes/meme3.jpg" file
Server> Client requested: "Meme 8", returning: "memes/meme8.jpg" file
Server> Client requested: "Meme 2", returning: "memes/meme2.jpg" file
Server> Client requested: "Meme 10", returning: "memes/meme10.jpg" file
Server> Client requested: "Meme 2", returning: "memes/meme2.jpg" file
Server> Client requested: "Meme 9", returning: "memes/meme9.jpg" file
Server> Client disconnected.
```

```
Client> exit

Round Trip Time Statistics (in nanoseconds):
Minimum: 3957819
Maximum: 7880139
Mean: 5140360.8
Standard Deviation: 1178043.5130806332

DNS Lookup Time Statistics (in nanoseconds):
Minimum: 11990
Maximum: 24699
Mean: 17657.0
Standard Deviation: 4077.799774388144
ethans-MacBook-Pro:pa2 ethanchan$
```

Lesson Learned:

Compared to experiment #1, experiment #3 has a much greater mean for both TCP and UDP. But compared to experiment #2 the difference in means is not that noticeable for both TCP and UDP.

The lesson learned while completing this assignment is the knowledge of transferring a TCP-based communication to UDP. We learned about the lifecycle of a UDP connection, how it starts, the data transfer, and the closure. As well we learned about the error handle, how it acts when a connection has already been started, and what happens if a picture does not exist.