**5COSC022W.2 Client Server Architectures**

**Tutorial Week 03: Socket Programming**

# Exercise 1: Building Basic Client-Server Communication in Java

**Objective:** The goal of this exercise is to understand the fundamentals of network programming in Java by creating simple client and server classes. You will learn how to establish a connection, send messages, and receive responses using sockets.

**Overview:** In this exercise, you will be creating two separate classes - Client and Server. The Client class will connect to the Server class using a specific port. Once the connection is established, the Client will send a message to the Server, which will then respond back to the Client.

**Key Concepts:**

* **Sockets:** A socket is one endpoint of a two-way communication link between two programs running on the network. You will use the Socket class for the client and the ServerSocket class for the server.
* **Input/Output Streams:** These are used for communication between the client and server. You will use the InputStreamReader and BufferedReader classes for reading data, and the PrintWriter class for writing data.
* **Exception Handling:** Network operations can fail for various reasons, so it’s important to handle exceptions properly in your code. You will use a try-catch block to catch any IOException that may occur.

**Expected Outcome:** By the end of this exercise, you should be able to create a simple client-server application in Java. You will gain a basic understanding of network programming, and be prepared to explore more complex topics like multi-threading and non-blocking I/O.

## Steps:

1. **Download the Project Zip File**:
   * Log in to your Blackboard account.
   * Click on the ‘Week 3’ section.
   * Find the file named Tutorial\_week03\_1\_Questions.zip.
   * Click on the file name to download it.
2. **Extract the Zip File**:
   * Locate the downloaded zip file on your computer. It’s usually in the ‘Downloads’ folder unless you chose a different location. You may also you the ZIP file to import it directly to the NetBeans without requiring extracting the ZIP file.
   * Right-click on the zip file and select ‘Extract All…’.
   * Choose a suitable location on your computer to extract the files.
3. **Open NetBeans IDE**:
   * Launch the NetBeans IDE 18 or above on your computer.
4. **Import the Project**:
   * In NetBeans, click on ‘File’ in the menu bar, then select ‘Open Project’.
   * In the file chooser window, navigate to the location where you extracted the project files.
   * You should see a folder with the same name as the zip file. Click on it to select it.
   * Click on the ‘Open Project’ button.
5. **Explore the Project**:
   * The project should now be visible in the ‘Projects’ tab on the left side of the IDE.
   * Click on the project name to expand it and explore its structure.
6. **Implement Your Code**:
   * Now, you can start implementing your code under each comment in the provided classes.

# Exercise 2: Building a Simple Chat Application in Java

**Objective:** The goal of this exercise is to understand the fundamentals of network programming in Java by creating simple chat client and server classes. You will learn how to establish a connection, send messages, and receive responses using sockets.

**Overview:** In this exercise, you will be creating two separate classes - SimpleChatClient and SimpleChatServer. The SimpleChatClient class will connect to the SimpleChatServer class using a specific port. Once the connection is established, the SimpleChatClient will send a message to the SimpleChatServer, which will then respond back to the SimpleChatClient.

**Key Concepts:**

* **Sockets:** A socket is one endpoint of a two-way communication link between two programs running on the network. You will use the Socket class for the client and the ServerSocket class for the server.
* **Input/Output Streams:** These are used for communication between the client and server. You will use the InputStream and OutputStream classes for reading data and writing data.
* **Exception Handling:** Network operations can fail for various reasons, so it’s important to handle exceptions properly in your code. You will use a try-catch block to catch any IOException that may occur.

**Expected Outcome:** By the end of this exercise, you should be able to create a simple chat application in Java. You will gain a basic understanding of network programming, and be prepared to explore more complex topics like multi-threading and non-blocking I/O.

## Steps:

1. **Download the Project Zip File**:
   1. Log in to your Blackboard account.
   2. Click on the ‘Week 3’ section.
   3. Find the file named Tutorial\_week03\_2\_Questions.zip.
   4. Click on the file name to download it.
2. **Extract the Zip File**:
   1. Locate the downloaded zip file on your computer. It’s usually in the ‘Downloads’ folder unless you chose a different location. You may also you the ZIP file to import it directly to the NetBeans without requiring extracting the ZIP file.
   2. Right-click on the zip file and select ‘Extract All…’.
   3. Choose a suitable location on your computer to extract the files.
3. **Open NetBeans IDE**:
   1. Launch the NetBeans IDE on your computer.
4. **Import the Project**:
   1. In NetBeans, click on ‘File’ in the menu bar, then select ‘Open Project’.
   2. In the file chooser window, navigate to the location where you extracted the project files.
   3. You should see a folder with the same name as the zip file. Click on it to select it.
   4. Click on the ‘Open Project’ button.
5. **Explore the Project**:
   1. The project should now be visible in the ‘Projects’ tab on the left side of the IDE.
   2. Click on the project name to expand it and explore its structure.
6. **Implement Your Code**:
   1. Now, you can start implementing your code under each comment in the provided classes.