# Day 1: Introduction to Java Socket Programming
# Socket and ServerSocket Basics

Irfan Ferdous Siam
BSc in CSE, Green University of Bangladesh

## Contents

# 1 What Is Socket Programming?

## 1.1 Definition

Socket programming is a way for two computers or programs to communicate with each other over a network using code. In Java, socket programming enables two applications to exchange data through TCP/IP protocols by using Socket APIs. It allows a client (user) to send a request to a server, and the server to respond with a reply.

## 1.2 Real-Life Analogy

Imagine two people talking over walkie-talkies:

- One speaks first (Client)
- The other listens and replies (Server)

Another, Imagine you and your friend are using walkie-talkies:

- One initiates the conversation (Client)
- The other listens and responds (Server)

This is exactly how socket communication works:

- The client connects to a specific IP + port
- The server listens at that port and accepts incoming requests

# 2 Java Classes for Socket Programming

| **Java Class** | **Definition** |
|---|---|
| ServerSocket | A Java class that waits for client connections on a specific port. |
| Socket | A Java class that connects to a server using an IP address and port. |
| InputStream | Used to read data (messages) from the connection. |
| OutputStream | Used to send data (messages) over the connection. |

Today well focus only on Socket and ServerSocket.

# 3 Basic Networking Terms

Every CSE student should know these terms:

| Term | Simple Meaning |
|---|---|
| IP Address | Internet address of a computer (like a house number). |
| Port | A door number used to identify specific processes running on a device. |
| localhost | Refers to the current computer (IP: 127.0.0.1). |
| TCP | A reliable, connection-based protocol (used in most socket programs). |
| UDP | A fast but unreliable protocol (not used in Java socket basics). |

# 4 ServerSocket: The Listener

## 4.1 Definition

A ServerSocket is used by the server to open a specific port and wait for client connections. **Think of it as a shopkeeper waiting for a customer to walk in.**

```
1 ServerSocket serverSocket = new ServerSocket(1234);
```

Listing 1: ServerSocket Initialization

- This line opens port 1234
- The server will now wait for a client to connect
- Waits for client requests using `.accept()`

This `.accept()` method is blocking  it will pause the program until a client connects.

# 5 Socket: The Connector

## 5.1 Definition

Socket is used on the client side (or by the server after a connection is accepted) to send and receive data. A Socket represents a connection between two devices. It can be created on:

- The client side (to connect to a server)
- The server side (after accepting a connection)

**Think of it as a customer walking into a shop and talking to the shopkeeper.**

```
1 Socket socket = new Socket("localhost", 1234);
```

Listing 2: Socket Initialization

- This line connects to a server running on the same machine (localhost) and port 1234
- Connects to a server running on localhost (this computer) at port 1234

**Note:**
localhost means this same computer  its the IP address 127.0.0.1. So both client and server run on your own machine.

# 6 Why throws IOException?

Java socket operations can fail due to:

- Port already in use
- Server not running
- Network issue or timeout

These issues cause an IOException. Thats why we write `throws IOException` for now well learn try-catch error handling on Day 2.

# 7 Code Example: Simple Connection

## 7.1 Server Code

```
1  import java.io.*;
2  import java.net.*;
3
4  public class Server {
5      public static void main(String[] args) throws IOException {
6          ServerSocket serverSocket = new ServerSocket(1234); //
               Step 1
7          System.out.println("Server started. Waiting for a client
               ...");
8
9          Socket socket = serverSocket.accept(); // Step 2
10         System.out.println("Client connected from " + socket.
               getInetAddress());
11
12         socket.close();         // Step 3
13         serverSocket.close();   // Step 4
14     }
15 }
```

Listing 3: Server.java

### 7.1.1 Line-by-Line Explanation

1. `ServerSocket serverSocket = new ServerSocket(1234);`
   Opens port 1234 and starts listening for clients.

2. `Socket socket = serverSocket.accept();`
   Waits for a client to connect. This is a blocking call (the program stops here until a client connects).

3. `System.out.println("Client connected from " + socket.getInetAddress());`
   Prints the clients IP address  a useful method of the Socket class.

4. `socket.close();`
   Closes the client connection.

5. `serverSocket.close();`
   Closes the server.

## 7.2 Client Code

```java
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("localhost", 1234); // Step 1
        System.out.println("Connected to the server.");

        socket.close(); // Step 2
    }
}
```

Listing 4: Client.java

### 7.2.1 Line-by-Line Explanation

1. `Socket socket = new Socket("localhost", 1234);`
   Tries to connect to the server on the same machine (localhost) at port 1234.

2. `socket.close();`
   Closes the connection.

# 8 Common Errors

| Issue | What Happens |
|---|---|
| Running client first | Client cant find server  connection error |
| Using busy port | Java will say port already in use |
| Not closing socket | May cause memory or network issues |
| Server not started | Client will throw ConnectionRefusedException |

# 9 Best Practices

- Always run the server first, then the client

- Use try-catch blocks to handle errors (well do this in later days)
- Always close your sockets to free system resources
- Keep port numbers above 1024 (below that are often reserved)
- Close your Socket and ServerSocket to avoid issues

# 10 Practice Task

Create two Java files: `Server.java` and `Client.java` using the code above. Then:

1. Compile and run `Server.java`
2. Then run `Client.java`
3. Try changing the port from 1234 to 5678
4. Try running the client before the server  note the error message.
5. **Bonus:** Modify the server to print the clients IP address using `socket.getInetAddress()` (already added above).