Problem 1: Creating a Simple Server

Write a Java program to create a simple server that listens on port 8080 and sends a welcome message to any client that connects. **Instructions:**

- Use the ServerSocket class to create the server.
- Accept client connections and send the message "Welcome to the Server!".

Sample Output:

```
Server is listening on port 8080...
Client connected!
Message sent: Welcome to the Server!
```

Problem 2: Creating a Simple Client

Write a Java program to create a client that connects to a server on port 8080 and prints the message received from the server. **Instructions:**

- Use the Socket class to connect to the server.
- Read the message sent by the server and display it.

Sample Output:

```
Connected to the server.
Message received: Welcome to the Server!
```

Problem 3: Chat Application

Write a Java program to create a simple chat application where a client and a server can exchange messages. Implement basic functionality for sending and receiving messages. **Instructions:**

- Use InputStream and OutputStream for communication.
- Implement a loop to allow multiple messages to be exchanged.

Sample Output:

```
Server: Hello!
Client: Hi!
```

Server: How are you?
Client: I am good, thanks!

Problem 4: Multithreaded Server

Write a Java program to create a multithreaded server that can handle multiple clients simultaneously. **Instructions:**

- Use the Thread class to handle client connections.
- Each thread should handle communication with a single client.

Sample Output:

Server is listening on port 8080...
Client 1 connected!
Client 2 connected!
Message from Client 1: Hello Server! Message from
Client 2: Hi Server!

Problem 5: Broadcasting Messages to Clients

Write a Java program where a server broadcasts a message to all connected clients.

Instructions:

- Use a list to store all connected clients.
- Send the same message to all clients in the list.

Sample Output:

Server: Broadcasting message: "Hello Clients!" Client 1 received:
Hello Clients!
Client 2 received: Hello Clients!

Problem 6: File Transfer Application

Write a Java program where a client sends a file to a server. **Instructions:**

- The client reads a file and sends its content to the server.
- The server saves the file with a specified name.

Sample Output:

```
Client: Sending file...
Server: File received and saved as "received.txt".
```

Problem 7: Timeout Handling

Write a Java program to set a timeout for server connections. If a client does not connect within 10 seconds, the server should terminate the connection.

Instructions:

- Use setSoTimeout() to set a timeout on the server socket.
- Handle the SocketTimeoutException.

Sample Output:

```
Server is listening on port 8080...
Connection timed out after 10 seconds.
```