How **ServerSocket** Works

- Each **ServerSocket** listens on a particular port on the server machine.
- When a client **Socket** on a remote machine attempts to connect to that port, the server wakes up, negotiates a connection, and opens a regular **Socket** between the two machines.
- Once a **ServerSocket** sets up a connection, the server uses a regular **Socket** to send data to the client. Data always travels over a regular **Socket**.

Life Cycle of a Server

1. A new **ServerSocket** is created on a particular port using a **ServerSocket()** constructor.
2. The **ServerSocket** listens for incoming connection attempts on that port using its **accept()** method. **accept()** blocks until a client attempts to make a connection. Then **accept()** returns a **Socket** connecting the server to the client.
3. Either or both of the **Socket**'s **getInputStream()** and **getOutputStream()** methods are called to get input and output streams that communicate with the client.

Life Cycle of a Server

4. The server and the client interact according to an agreed-upon protocol until it is time to close the connection.
5. The server, the client, or both close the connection.
6. The server returns to Step 2 and waits for another connection.

One More Time...

- A **ServerSocket** generally operates in a loop that repeatedly accepts connections.
- Each pass through the loop invokes the **accept()** method which returns a **Socket**.
- Interaction with the client takes place through the **Socket**.
- When the transaction is finished, the server should invoke the **Socket**'s **close()** method and get ready for the next connection.
- If the server needs to shut down and stop processing requests, the server should invoke the **ServerSocket**'s **close()** method.

Handling Exceptions

- Exceptions thrown by **`ServerSocket`** should probably shut down the server and log an error message.
- Exceptions thrown by **`Socket`** should just close the active connection.
- Exceptions thrown by the **`accept()`** method are an intermediate case that can go either way.
- To do all this, you need to nest **`try`** blocks which leads to convoluted code. ☹