Lecture 2 - Java Socket Programming and I/O

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

CSC-1004: Computational Laboratory Using Java
Course Page: [Click]

Introduction

Java Socket programming is used for communication between the applications running on different machines.

Introduction

In Java Socket programming, the most important elements are:

- Socket and ServerSocket are used for connection-oriented socket programming.
- The client must know the IP Address of the server and port number.

Introduction

One-way client and server communication : the client sends a message to the server, and the server reads the message and prints it.

The Socket class communicates client and server.

The ServerSocket class is used on the server side. Its accept() method blocks the console until the client is connected.

Example of Java Socket Programming

Creating Server: 1) We use the 6666 port number for the communication between the client and server. 2) The accept() method waits for the client. If clients connect with the given port number, it returns an instance of Socket.

Example of Java Socket Programming

Creating Client: We pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on the same system.

Example of Java Socket Programming

MyServer.java

MyClient.java

Java I/O (Input and Output)

Basic Knowledge of Java Input and Output .

Stream: A stream is a sequence of data. In Java, a stream is composed of bytes.

OutputStream: Java application uses an output stream to write data to a destination; it may be a file, an array or sockets.

InputStream: Java application uses an input stream to read data from a source; it may be a file, an array, or sockets.

Java I/O (Input and Output)

Java I/O (Input and Output)

Stream Wrappers

What if we want to do more than read and write a mess of bytes or characters?

DataInputStream and DataOutputStream are ltered streams that let you read or write strings and primitive data types that comprise more than a single byte.

The readUTF() and writeUTF() methods of DataInputStream and DataOutputStream read and write a Java String of Unicode characters using the UTF-8 (encoding of Unicode characters commonly used for the transmission and storage of Unicode text).

Use the ush() method to data out the contents.

Java I/O (Input and Output)

Java I/O (Input and Output)

Java I/O (Input and Output)

Character Streams

InputStreamReader and OutputStreamWriter , bridge the gap between the world of character streams and the world of byte streams. These are character streams that are wrapped around an underlying byte stream.

When we wrap an InputStreamReader around System.in, we object converts the incoming bytes of System.in to characters using the default encoding scheme.

Java I/O (Input and Output)

Bu ered streams

The Bu eredReader , and Bu eredWriter classes add a data bu er of a speci ed size to the stream path.

A bu er can increase e ciency by reducing the number of physical read or write operations that correspond to read() or write() method calls.

You can wrap another stream around a bu ered stream. Bu eredReader gives us the readLine() method, which we can use to retrieve a full line of text into a String.

Java I/O (Input and Output)

Read data from the console.

Enter your
Nakul Jain
Welcome Nakul Jain

Java I/O (Input and Output)

Read data from console until the user writes stop

Enter data: Nakul

data is: Nakul

Enter data: 12

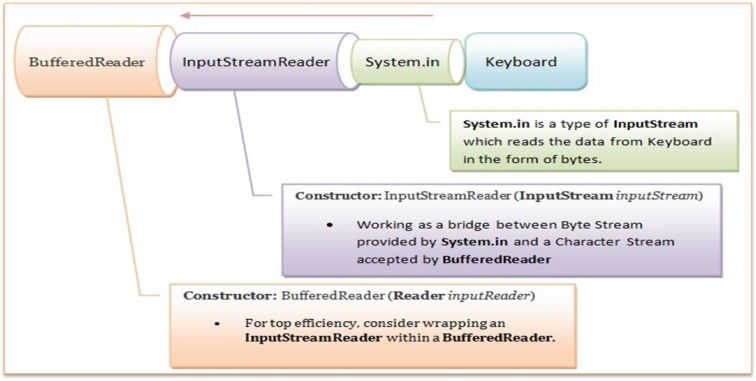
data is: 12

Enter data: stop

data is: stop

Java I/O (Input and Output)

Java InputStreamReader – Read from Keyboard



Example of Java Socket Programming and Data Streaming

MyServer.java

```
import java.net.*;
import java.io.*;
class MyServer{
public static void main(String args[])throws Exception{
ServerSocket ss=new ServerSocket(3333);
Socket s=ss.accept();
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";
while(!str.equals("stop")){
str=din.readUTF();
System.out.println("client says: "+str);
str2=br.readLine();
dout.writeUTF(str2);
dout.flush();
}
din.close();
s.close();
ss.close();
}}
```

MyClient.java

```
import java.net.*;
import java.io.*;
class MyClient{
public static void main(String args[])throws Exception{
Socket s=new Socket("localhost",3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";
while(!str.equals("stop")){
str=br.readLine();
dout.writeUTF(str);
dout.flush();
str2=din.readUTF();
System.out.println("Server says: "+str2);
}

dout.close();
s.close();
}}
```

Question and Answering (Q&A)