IO in Java

Contents

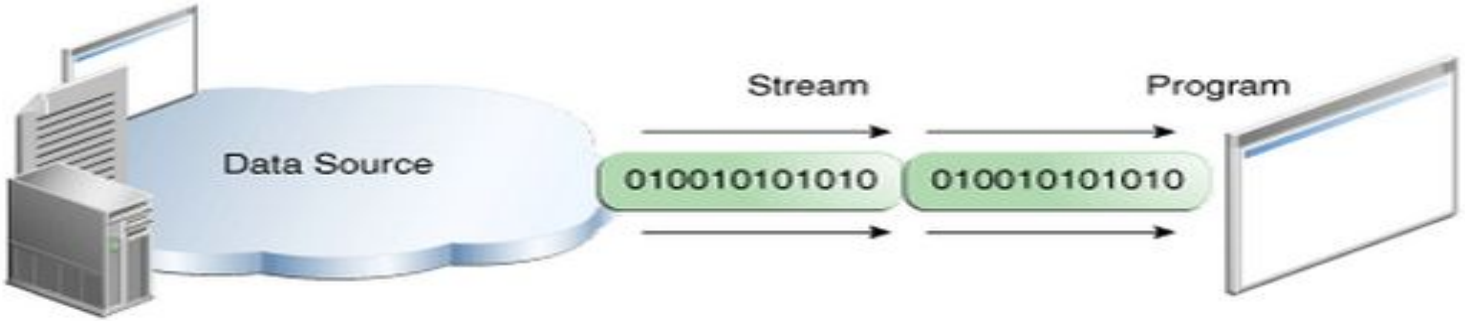
1. What are I/O streams
2. Different kinds of I/O Streams & its application
3. File & Folder
4. Java Socket I/O

What are I/O Streams?

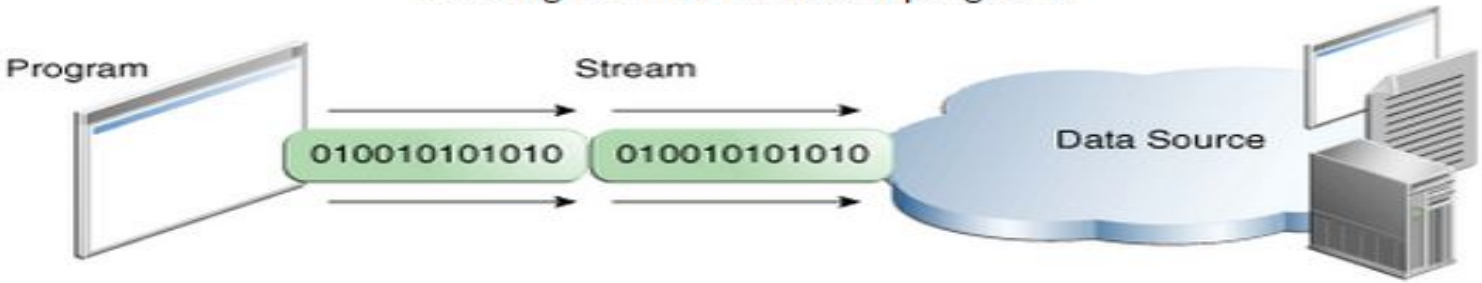
- **A stream is a sequence of data**. A program uses an ***input stream*** **to read data from a source**, one item at a time
- A program uses an ***output stream*** **to write data to a destination**, one item at time
- The data source and data destination can be anything that **holds, generates, or consumes data**. Obviously this includes **disk files**, but a source or destination **can also be another program, a peripheral device, a network socket, or an array**

Source: https://docs.oracle.com/javase/tutorial/essential/io/streams.html

What are I/O Streams



Reading information into a program.



Writing information from a program.

Source: https://docs.oracle.com/javase/tutorial/essential/io/streams.html

Different kinds of IO Streams

1. **Byte Streams**: I/O dữ liệu nhị phân
2. **Character Streams**: I/O dữ liệu ký tự
3. **Buffered Streams**: tối ưu I/O, giảm số lần gọi đến các Native API.
4. **Data Streams**: I/O String values & các kiểu dữ liệu cơ sở.
5. **Object Streams**: I/O binary của các Objects.

Byte Streams

- **Perform input and output of 8-bit bytes**. All byte stream classes are descended from **InputStream** and **OutputStream**.
- There are many byte stream classes. To demonstrate how byte streams work, we'll focus on the file I/O byte streams, **FileInputStream** and **FileOutputStream**. Other kinds of byte streams are used in much the same way;

Source: https://docs.oracle.com/javase/tutorial/essential/io/streams.html

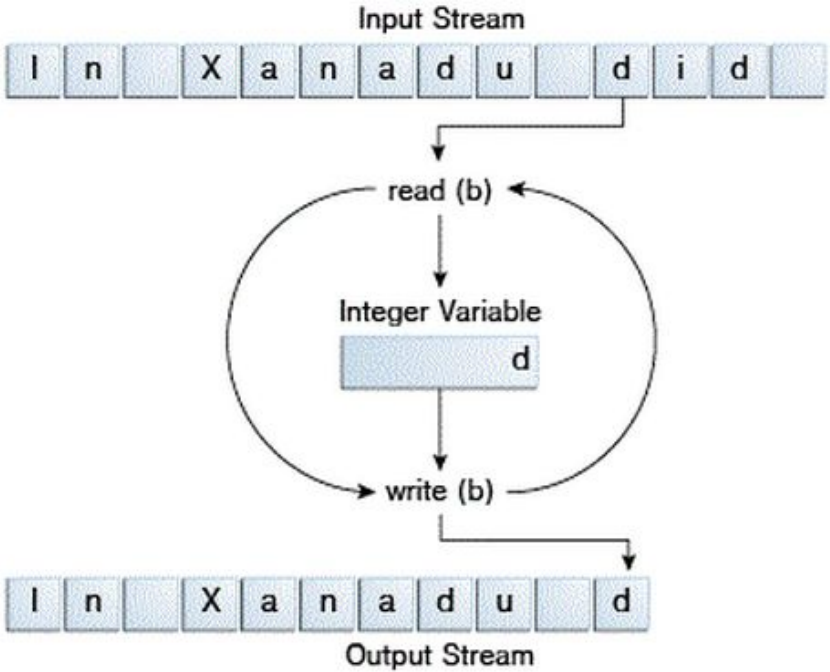
Byte Streams - Example

Example 1: I/O file using Byte Streams

```
FileInputStream in = null;
FileOutputStream out = null;
try {
    in = new FileInputStream("/home/tinhuynh/NetBeansProjects/JavaCoreDemo/in.txt");
    out = new FileOutputStream("/home/tinhuynh/NetBeansProjects/JavaCoreDemo/out.txt");
    int c;
    while ((c = in.read()) != -1) {
        out.write(c); // Writes the specified byte to this file output stream.
    }
    // the next byte of data, or -1 if the end of the file is reached.
}
finally {
    if (in != null)
        in.close();
    if (out != null)
        out.close();
}
```

Byte Streams

- Open stream.
- I/O.
- Close stream



Simple byte stream input and output.

Source: https://docs.oracle.com/javase/tutorial/essential/io/streams.html

Byte Streams

- Khi nào không dùng Byte Streams?
  - I/O character tốt nhất dùng character streams.
  - Byte Streams dùng cho các I/O nền tảng nhất. Byte Streams là nền tảng để xây dựng nên các Streams khác.

Character Streams

- Character stream I/O **automatically translates this internal format to and from the local character set**.
- All character stream classes are descended from **Reader** and **Writer**.
- File I/O: FileReader and FileWriter.

Character Streams - example

Example 2: I/O file using FileReader & FileWriter

```
try {
    in1 = new FileReader("/home/tinhuynh/NetBeansProjects/JavaCoreDemo/in.txt");
    out1 = new FileWriter("/home/tinhuynh/NetBeansProjects/JavaCoreDemo/out.txt");
    int c;
    while ((c = in1.read()) != -1) {
        out1.write(c);
    }
    // the next byte of data, or -1 if the end of the file is reached.
} finally {
    if (in1 != null) {
        in1.close();
    }
    if (out1 != null) {
        out1.close();
    }
}
```

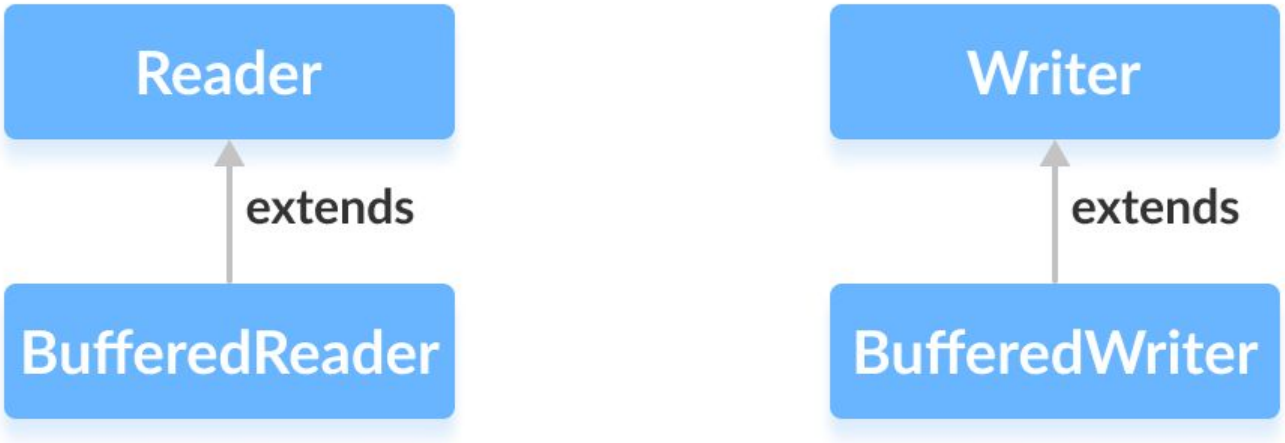
- *Ví dụ 1, Ví dụ 2 dùng biến int để đọc và ghi. Biến int trong ví dụ 2 lưu giá trị Character 16 bits. Biến int trong ví dụ 1 lưu giá trị Byte 8 bits.*

Character Streams - example

- *Example 3: kết hợp FileReader, FileWriter với BufferedReader, BufferedWriter/PrintWriter*

```
BufferedReader inputStream = null;
PrintWriter outputStream = null;
try {
    inputStream = new BufferedReader(new FileReader("input.txt"));
    outputStream = new PrintWriter(new FileWriter("output.txt"));
    String l;
    while ((l = inputStream.readLine()) != null) {
        outputStream.println(l);
    }
} finally {
    if (inputStream != null) {
        inputStream.close();
    }
    if (outputStream != null) {
        outputStream.close();
    }
}
```

BufferedReader, BufferedWriter



Why BufferedReader, BufferedWriter?

- Maintains an **internal buffer** of 8192 characters.
- During the read/write operation, **a chunk of characters is read/written from/to the disk and stored in the internal buffer**. And from the internal buffer characters are read/written individually.
- Hence, **the number of communication to the disk is reduced**. This is why using BufferedReader/BufferedWriter is faster.

Why BufferedReader, BufferedWriter?

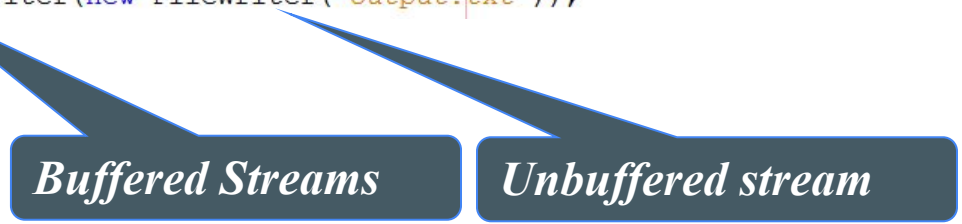
```
for(int i = 0; i < 100; i++) {
    writer.write("foorbar");
    writer.write(NEW_LINE);
}
writer.close();
```

- *Ví dụ này sẽ gọi Native API (System calls) 200 lần để ghi dữ liệu. Trong khi dùng luồng đệm thì gọi Native API 1 lần.*

Buffered Streams

- Chuyển luồng không đệm luồng đệm dùng kiểu bao

```
BufferedReader inputStream = new BufferedReader(new FileReader("input.txt"));
BufferedWriter outputStream = new BufferedWriter(new FileWriter("output.txt"));
```



- **BufferedInputStream & BufferedOutputStream**: *đệm byte*
- **BufferedReader & BufferedWriter**: *đệm character*

Buffered Streams - Example

```
public static void copyFile(String fileIn, String fileOut) throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(fileIn));
    BufferedWriter bw = new BufferedWriter(new FileWriter(fileOut));
    String line = br.readLine();
    while (line != null) {
        bw.write(line);
        bw.newLine();
        line = br.readLine();
    }
    bw.flush(); // Flush a stream manually
    br.close();
    bw.close();
}
```

Buffered Streams - Example

```
public static void copyFile(String fileIn, String fileOut) throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(fileIn));
    // Some buffered output classes support autoflush,
    // specified by an optional constructor argument
    PrintWriter bw = new PrintWriter(new FileWriter(fileOut), true);
    String line = br.readLine();
    while (line != null) {
        bw.println(line);
        line = br.readLine();
    }
    br.close();
    bw.close();
}
```

- *Đối tượng PrintWriter tự động flush vùng đệm mỗi lần gọi phương thức println(...).*

Java Scanner

- Scanner class in Java is found in the **java.util package**.
- That is **used to get input from standard I/O or files with primitives types such as int, double, strings**, …
- A simpler text scanner which can parse primitive types and strings **using regular expression**. A Scanner **breaks its input into tokens using delimiter pattern**, which by default matches whitespace (includes tabs, blanks, and line terminators).

Java Scanner - Example

```
// creates an object of Scanner
Scanner input = new Scanner(System.in);
System.out.print("Enter your full name: ");
// takes input from the keyboard
String fullname = input.nextLine();
// prints the name
System.out.println("Your full name is " + fullname);
// closes the scanner
input.close();
```

Java Scanner - Example

```
// creates an object of Scanner
Scanner input = new Scanner(System.in);
System.out.print("Enter an integer: ");
// takes input from the keyboard
int number = input.nextInt();
// prints the name
System.out.println("Input int:" + number);
// closes the scanner
input.close();
```

Java Scanner - Example

```
try (Scanner s = new Scanner(new BufferedReader(
        new FileReader("C:\\Input.txt")))) {
    while (s.hasNext()) {
        System.out.println(s.next());
    }
}
```

Java Scanner - Example

```
Scanner scan = new Scanner("NguyenVanA;NguyenVan2;NguyenVanC");
// declare the delimiter to be used by Scanner object
scan.useDelimiter(";");
Pattern pattern = Pattern.compile("[A-Za-z]*");
while (scan.hasNext()) {
    // check if the token consists of declared pattern
    if (scan.hasNext(pattern)) {
        System.out.println(scan.next());
    } else {
        scan.next();
    }
}
scan.close(); // closing the scanner stream
```

Java Scanner - disadvantage

- Scanner has a little buffer (1KB char buffer).
- Scanner is **slower than BufferReader because Scanner does parsing of input data**, and BufferReader simply reads sequence of characters.
- A Scanner is **not safe for multithreaded** use without external synchronization.

Formatting

- Formatting API: đưa dữ liệu về kiểu phù hợp.
- Các luồng hiện thực việc định dạng dữ liệu: bổ sung thêm tập các phương thức cho phép chuyển đổi kiểu dữ liệu nội tại bên trong thành dữ liệu định dạng tương ứng đầu ra.
  - **PrintWriter**: luồng ký tự
  - **PrintStream**: luồng byte

Formatting

- Phương thức **format()**: định dạng hầu hết các giá trị số dựa trên một chuỗi với nhiều options định dạng trước.

```
int i = 3;
double r = Math.sqrt(i);
System.out.format("The square root of %d is %f.%n", i, r);
```

  - *d: formats an integer value as a decimal value.*
  - *f formats a floating point value as a decimal value.*
  - *n outputs a platform-specific line terminator.*
  - *x formats an integer as a hexadecimal value.*
  - *s formats any value as a string.*
  - *tB formats an integer as a locale-specific month name.*

Data Streams

- Hỗ trợ I/O String values & các kiểu dữ liệu cơ sở.
- *Ví dụ: ghi và đọc lên các mẫu tin từ file.*

```
static final String dataFile = "C:\\invoicedata";
static final int[] units = {1, 2, 3, 4, 5};
static final double[] prices = {100.00, 105.99, 15.99, 30.99, 4.99};
static final String[] descs = {
    "MotherBoard",
    "CPU",
    "Ram",
    "HDD",
    "Mouse"
};
```

Data Streams

```
try (DataOutputStream out = new DataOutputStream(new BufferedOutputStream(
        new FileOutputStream(dataFile)))) {
    // DataStreams writes out the records and closes the output stream.
    for (int i = 0; i < prices.length; i++) {
        // writeUTF method writes out String values in a modified form of UTF-8.
        // This is a variable-width character encoding
        // that only needs a single byte for common Western characters.
        out.writeUTF(descs[i]);
        out.writeInt(units[i]);
        out.writeDouble(prices[i]);
    }
}
```

Data Streams

```
double price;
int unit;
String desc;
double total = 0.0;
// DataStreams can read each record in the stream,
// reporting on the data it encounters.
try (DataInputStream in = new DataInputStream(
        new BufferedInputStream(new FileInputStream(dataFile)))) {
    while (true) {
        desc = in.readUTF();
        unit = in.readInt();
        price = in.readDouble();
        System.out.format("You ordered %d" + " units of %s at $%.2f%n",
                unit, desc, price);
        total += unit * price;
    }

} catch (EOFException e) {
}
```

Object Streams

- Object Streams hỗ trợ I/O binary của các Objects, Complex Objects.
- Luồng I/O Object: **ObjectInputStream** và **ObjectOutputStream**.

Class ObjectInputStream

java.lang.Object
    java.io.InputStream
        java.io.ObjectInputStream

All Implemented Interfaces:

`Closeable, DataInput, ObjectInput, ObjectStreamConstants, AutoCloseable`

Class ObjectOutputStream

java.lang.Object
    java.io.OutputStream
        java.io.ObjectOutputStream

All Implemented Interfaces:

`Closeable, DataOutput, Flushable, ObjectOutput, ObjectStreamConstants, AutoCloseable`

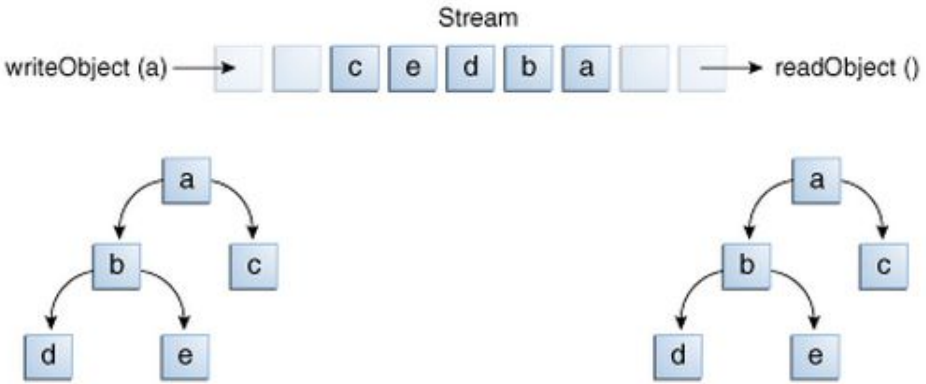
Tất cả phương thức I/O kiểu cơ sở trong Data Streams cũng được hiện thực trong Object Streams, do implement DataInput/DataOutput

```
String s1 = "Trường Đại học CNTT";
String s2 = "Nhập môn Lập trình Java";
int i = 897648764;
try {
    // create a new file with an ObjectOutputStream
    ObjectOutputStream out = new ObjectOutputStream(
            new FileOutputStream("C:\\OutputStream.txt"));
    // write something in the file
    out.writeObject(s1);
    out.writeObject(s2);
    out.writeObject(i);
    // close the stream
    out.close();

    // create an ObjectInputStream for the file we created before
    ObjectInputStream ois
            = new ObjectInputStream(new FileInputStream("C:\\OutputStream.txt"));
    // read and print what we wrote before
    System.out.println("" + (String) ois.readObject());
    System.out.println("" + (String) ois.readObject());
    System.out.println("" + ois.readObject());
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Object Streams

- I/O các complex Objects: gọi ***writeObject(a***) không chỉ ghi ***a***, mà tất cả các objects tạo nên/liên quan ***a (là b,c,d,e)*** cũng được ghi. Khi ***a*** được đọc bởi ***readObject***, các object liên quan ***(b,c,d,e)*** cũng được đọc. Tất cả các references được giữ nguyên. (hình vẽ)



I/O of multiple referred-to objects

Object Streams

- Khi 2 Objects (**A & B**) refer đến cùng 1 object nào đó (**C**)? Khi 2 Objects A & B được đọc lên trong luồng thì sao?
  - *Luồng chỉ chứa 1 copy của object **C**, nhưng có thể có nhiều references đến **C**.*
  - *Ghi 1 Object xuống stream 2 lần □ chỉ ghi 2 lần reference.*

```
Object ob = new Object();
out.writeObject(ob);
out.writeObject(ob);
Object ob1 = in.readObject();
Object ob2 = in.readObject();
```

- *Hai biến ob1, ob2 refer đến cùng 1 single object.*
- *Nhưng nếu 1 Object được ghi xuống 2 luồng khác nhau □ thì duplicate □ đọc từ 2 luồng sẽ thấy 2 object khác nhau.*

Làm việc thư mục – Lớp File

- Java.lang.Object

  + **java.io.File**

- **Lớp File**: không phục vụ cho việc nhập/xuất dữ liệu trên luồng. Lớp File thường dùng để biết được các thông tin chi tiết về tập tin cũng như thư mục (tên, ngày giờ tạo, kích thước, …)

Làm việc thư mục – Lớp File

- **Các Constructor:**
  - Tạo đối tượng File từ đường dẫn tuyệt đối

    public File(String pathname)

    ví dụ: File f = new File("C:\\Java\\vd1.java");
  - Tạo đối tượng File từ tên đường dẫn và tên tập tin tách biệt

    public File(String parent, String child)

    ví dụ: File f = new File("C:\\Java", "vd1.java");
  - Tạo đối tượng File từ một đối tượng File khác

    public File(File parent, String child)

    ví dụ: *File dir = new File ("C:\\Java");*

    File f = new File(dir, "vd1.java");

Làm việc thư mục – Lớp File

- **Một số phương thức thường dùng:**

public String getName()	Lấy tên của đối tượng File
public String getPath()	Lấy đường dẫn của tập tin
public boolean isDirectory()	Kiểm tra xem tập tin có phải là thư mục không?
public boolean isFile()	Kiểm tra xem tập tn có phải là một file không?
…	
public String[] list()	Lấy danh sách tên các tập tin và thư mục con của đối tượng File đang xét và trả về trong một mảng.

File, Directory - Example 1

```
public static void main(String[] args) {
      File dir = new File("/home/tinhuynh");
      String[] children = dir.list();

      if (children == null) {
         System.out.println( "Either dir does not exist or is not a directory");
      } else {
         for (int i = 0; i< children.length; i++) {
            String filename = children[i];
            System.out.println(filename);
         }
      }
   }
}
```

Source: https://www.tutorialspoint.com/javaexamples/dir_sub.htm

File, Directory - Example 2

```
public class ReadFiles {
   public static File folder = new File("/home/tinhuynh");
   static String temp = "";
   public static void main(String[] args) {
        System.out.println("Reading files under the folder "+
        folder.getAbsolutePath());
        listFilesForFolder(folder);
   }
```

Source: https://www.tutorialspoint.com/javaexamples/dir_sub.htm

```
public static void listFilesForFolder(final File folder) {
        for (final File fileEntry : folder.listFiles()) {
            if (fileEntry.isDirectory()) {
                listFilesForFolder(fileEntry);
            } else {
                if (fileEntry.isFile()) {
                    temp = fileEntry.getName();
                    if ((temp.substring(temp.lastIndexOf('.')
                            + 1, temp.length()).toLowerCase()).equals("txt")) {
                        System.out.println(
                                "File = " + folder.getAbsolutePath() + "\\" + fileEntry.getName());
                    }
                }
            }
        }
    }
}
```

Source: https://www.tutorialspoint.com/javaexamples/dir_sub.htm

Java Socket I/O

http://www.oracle.com/technetwork/java/socket-140484.html

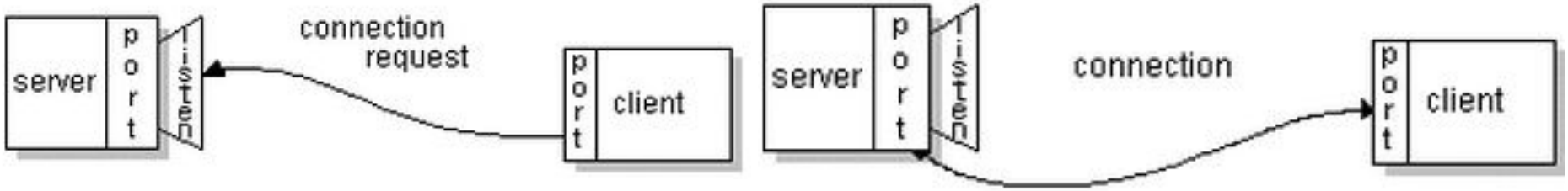
What is a socket?

- *A socket is one **end-point of a two-way communication link** between two programs running on the network. **Socket classes are used to represent the connection between a client program and a server program**. The java.net package provides two classes--Socket and ServerSocket--that implement the client side of the connection and the server side of the connection, respectively.*
- ***An end-point is a combination of an IP address and a port number**. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.*

http://www.oracle.com/technetwork/java/socket-140484.html

What is a Socket?

- Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.
- If everything goes well, the server accepts the connection.
- The client and server can now communicate by writing to or reading from their sockets



Java Socket - Example - Server

```
try ( var server = new ServerSocket(50000)) {
        System.out.println("The date server is running...");
        while (true) {
            try ( var socket = server.accept()) {
                var out = new PrintWriter(socket.getOutputStream(), true);
                out.println("Hello Client. This is SocketServer");
                out.println("InetAddress" + socket.getInetAddress().toString());
                long millis = System.currentTimeMillis();
                java.util.Date date = new java.util.Date(millis);
                out.println("Today is " + date);
            }
        }
}
```

Java Socket - Example - Client

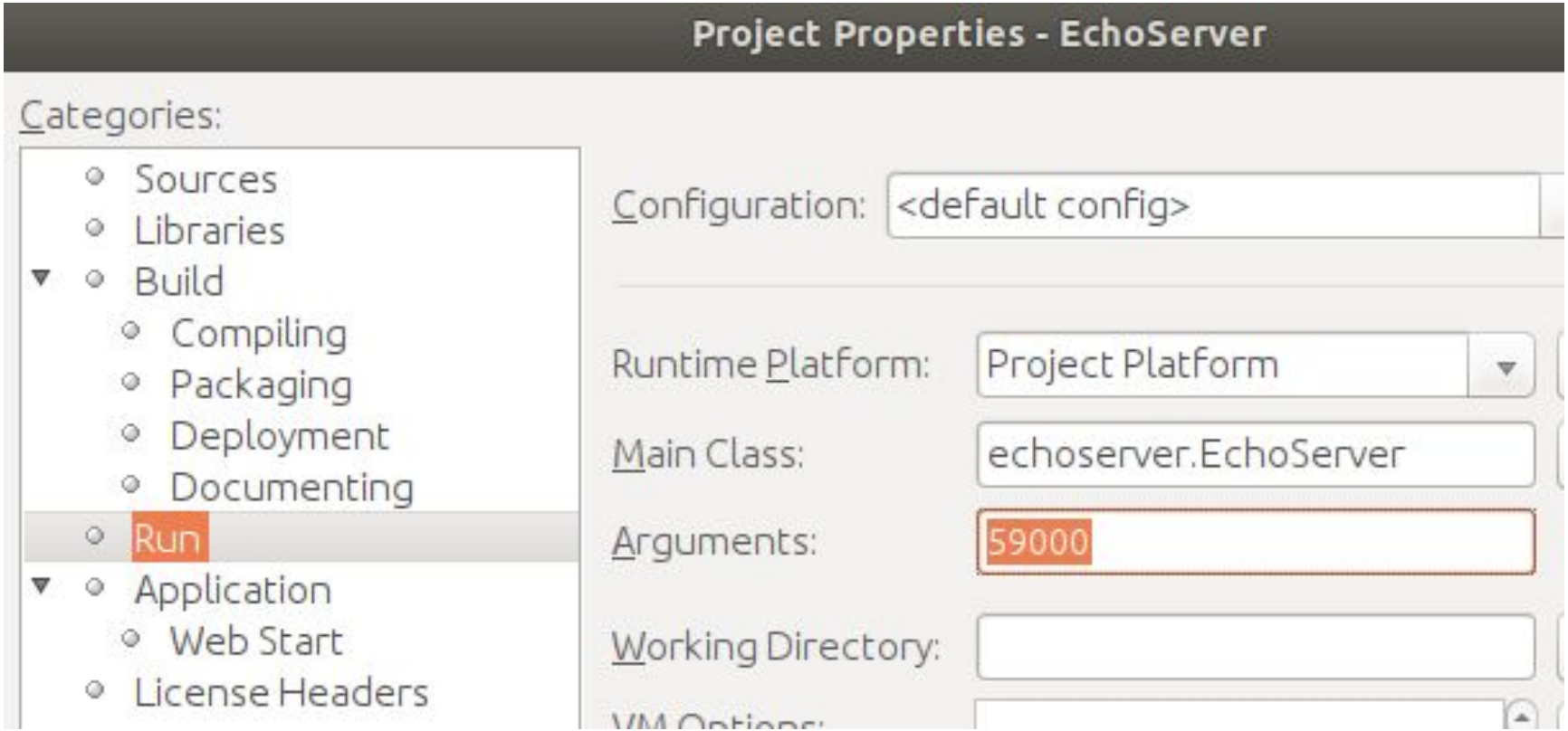
```
var socket = new Socket("localhost", 50000);
var in = new Scanner(socket.getInputStream());
System.out.println("Server response: ");
while(in != null && in.hasNext()) {
        System.out.println(in.nextLine());
}
```

Java Socket - Echo Example

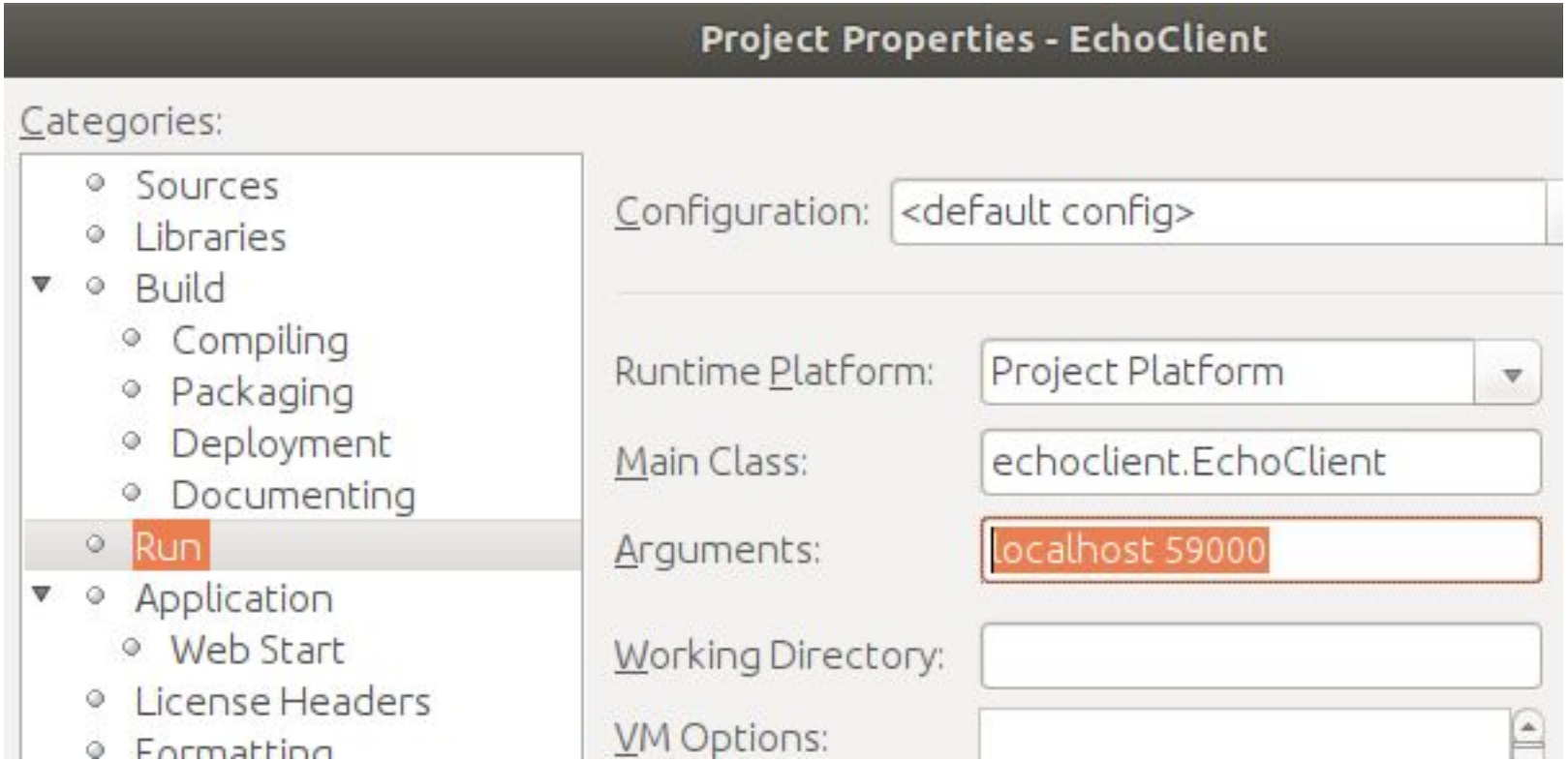
The example program implements a client, EchoClient, that connects to an echo server. The echo server receives data from its client and echoes it back.

Source:
https://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html

```java
public class EchoServer {
    public static void main(String[] args) throws IOException {
        if (args.length != 1) {
            System.err.println("Usage: java EchoServer <port number>");
            System.exit(1);
        }
        int portNumber = Integer.parseInt(args[0]);
        try (
                ServerSocket serverSocket = new ServerSocket(Integer.parseInt(args[0]));
                Socket clientSocket = serverSocket.accept();
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(
                        new InputStreamReader(clientSocket.getInputStream()));) {
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                out.println(inputLine);
            }
        } catch (IOException e) {
            System.out.println("Exception caught when trying to listen on port "
                    + portNumber + " or listening for a connection");
            System.out.println(e.getMessage());
        }
    }
}
```

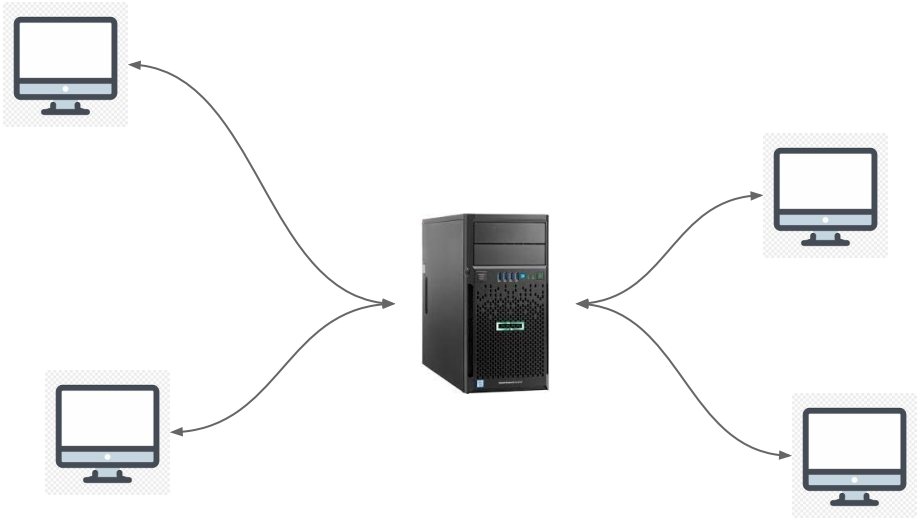

Project Properties - EchoServer
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Configuration:
<default config>
Runtime Platform:
Project Platform
Main Class:
echoserver.EchoServer
Arguments:
59000
Working Directory:

```java
public class EchoClient {
    public static void main(String[] args) throws IOException {
        if (args.length != 2) {
            System.err.println("Usage: java EchoClient <host name> <port number>");
            System.exit(1);
        }
        String hostName = args[0];
        int portNumber = Integer.parseInt(args[1]);
        try (
                Socket echoSocket = new Socket(hostName, portNumber);
                PrintWriter out = new PrintWriter(echoSocket.getOutputStream(), true);
                BufferedReader in = new BufferedReader(
                        new InputStreamReader(echoSocket.getInputStream()));
                BufferedReader stdIn = new BufferedReader(
                        new InputStreamReader(System.in))) {
            String userInput;
            while ((userInput = stdIn.readLine()) != null) {
                out.println(userInput);
                System.out.println("echo: " + in.readLine());
            }
        } catch (UnknownHostException e) {
            System.exit(1);
        } catch (IOException e) {
            System.exit(1);
        }
    }
}
```


Project Properties - EchoClient
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Configuration:
<default config>
Runtime Platform:
Project Platform
Main Class:
echoclient.EchoClient
Arguments:
localhost 59000
Working Directory:
VM Options:

Java Socket - Lan Chat Example

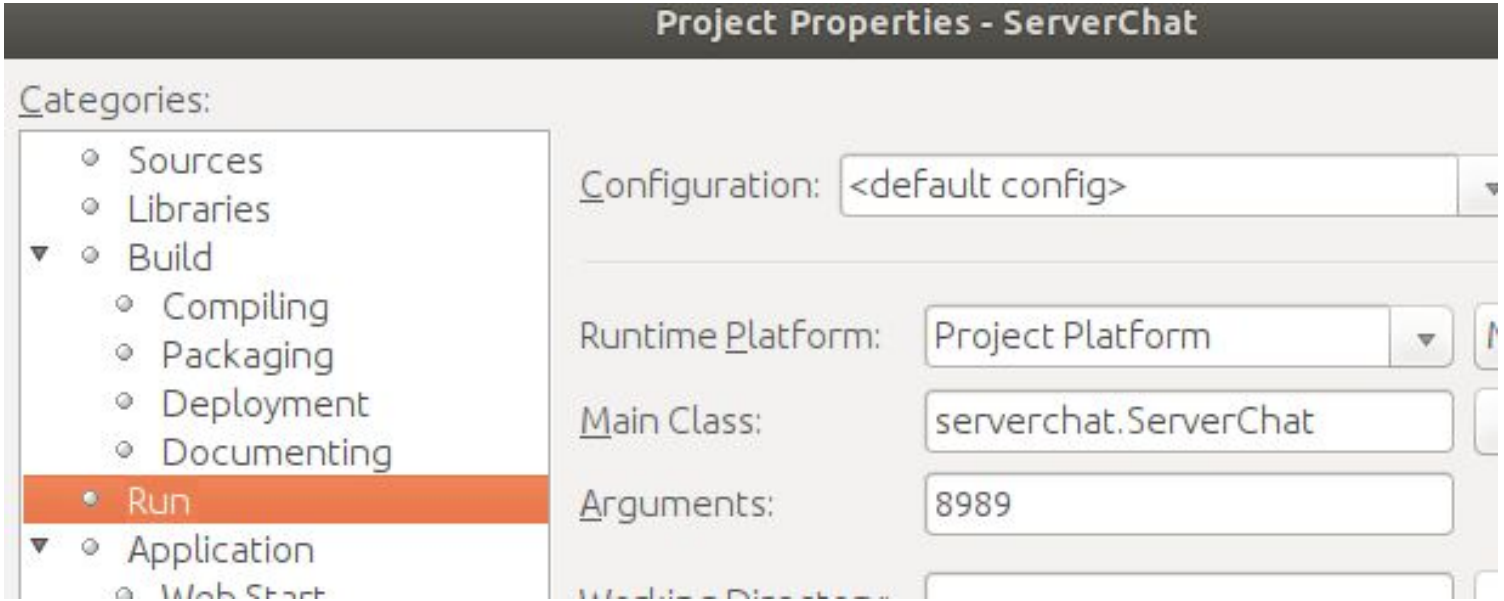
The chat application includes two different parts, **server** and **client** running independently on separate computers.



Java Socket - Lan Chat Example Server

- Open a **server socket** listening on a specified port
- Establish **a thread for each client** connected to.
- **Maintain a list of connected users/clients** as well as various threads for different clients, respectively.
- Delivers a message from one user to others (**broadcasting**)

Java Socket - Lan Chat Example Server



```
public class ServerChat {
    private int port;
    private Set<String> userNames = new HashSet<>();
    private Set<UserThread> userThreads = new HashSet<>();
    public ServerChat(int port) {
        this.port = port;
    }

    public static void main(String[] args) {
        // TODO code application logic here
        if (args.length < 1) {
            System.out.println("Syntax: java ChatServer <port-number>");
            System.exit(0);
        }
        int port = Integer.parseInt(args[0]);
        ServerChat server = new ServerChat(port);
        server.execute();
    }
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
public void execute() {
    try ( ServerSocket serverSocket = new ServerSocket(port)) {
        System.out.println("Chat Server is listening on port " + port);
        while (true) {
            Socket socket = serverSocket.accept();
            System.out.println("New user connected");
            UserThread newUser = new UserThread(socket, this);
            userThreads.add(newUser);
            newUser.start();
        }
    } catch (IOException ex) {
        System.out.println("Error in the server: " + ex.getMessage());
        ex.printStackTrace();
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
void broadcast(String message, UserThread excludeUser) {
    for (UserThread aUser : userThreads) {
        if (aUser != excludeUser) {
            aUser.sendMessage(message);
        }
    }
}

void addUserName(String userName) {
    userNames.add(userName);
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
    void removeUser(String userName, UserThread aUser) {
        boolean removed = userNames.remove(userName);
        if (removed) {
            userThreads.remove(aUser);
            System.out.println("The user " + userName + " quitted");
        }
    }
    Set<String> getUserNames() {
        return this.userNames;
    }
    boolean hasUsers() {
        return !this.userNames.isEmpty();
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
public class UserThread extends Thread {
    private Socket socket;
    private ServerChat server;
    private PrintWriter writer;
    public UserThread(Socket socket, ServerChat server) {
        this.socket = socket;
        this.server = server;
    }
    void printUsers() {
        if (server.hasUsers()) {
            writer.println("Connected users: " + server.getUserNames());
        } else {
            writer.println("No other users connected");
        }
    }
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
public void run() {
    try {
        InputStream input = socket.getInputStream();
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        OutputStream output = socket.getOutputStream();
        writer = new PrintWriter(output, true);
        printUsers();
        String userName = reader.readLine();
        server.addUserName(userName);
        String serverMessage = "New user connected: " + userName;
        server.broadcast(serverMessage, this);
        String clientMessage;
        do {
            clientMessage = reader.readLine();
            serverMessage = "[" + userName + "]: " + clientMessage;
            server.broadcast(serverMessage, this);
        } while (!clientMessage.equals("bye"));
        server.removeUser(userName, this);
        socket.close();
        serverMessage = userName + " has quitted.";
        server.broadcast(serverMessage, this);
    } catch (IOException ex) {
        System.out.println("Error in UserThread: " + ex.getMessage());
        ex.printStackTrace();
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

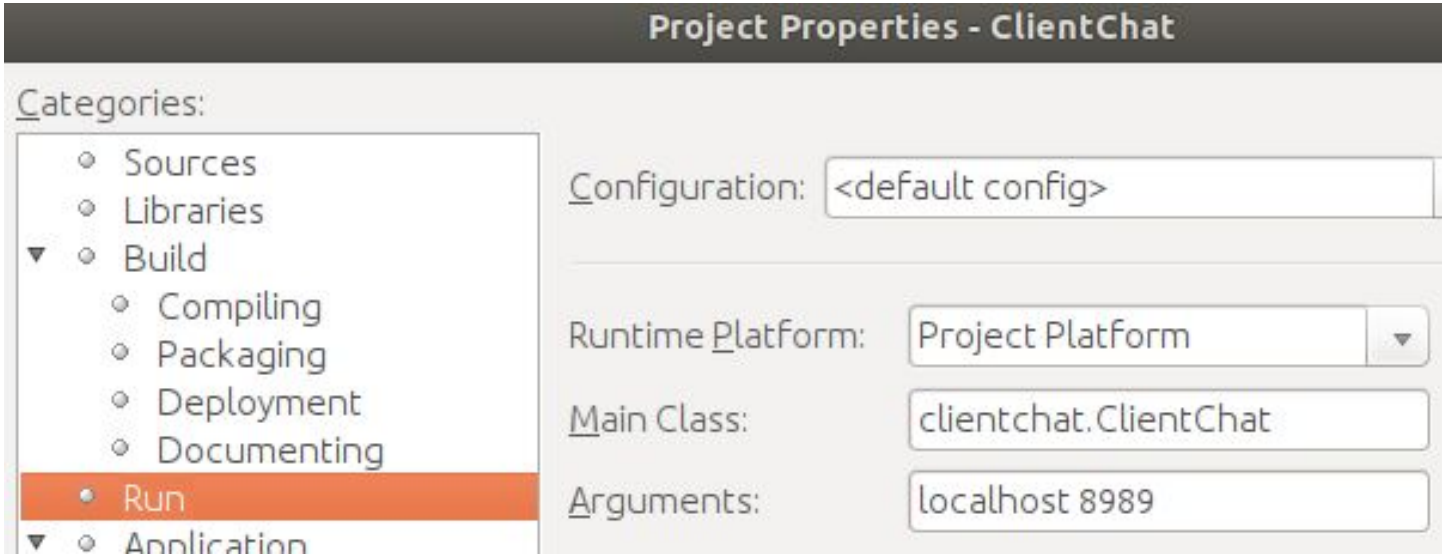
```
    void sendMessage(String message) {
        writer.println(message);
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

Java Socket - Lan Chat Example Client

- Connect to a specified server by establishing a connection to a the listening serversocket through IP, Port.
- Create 2 seperate threads, one for reading and another one for writing data from or to the connected socket, respectively.

Java Socket - Lan Chat Example Client



```
public class ClientChat {
    private String hostname;
    private int port;
    private String userName;
    public ClientChat(String hostname, int port) {
        this.hostname = hostname;
        this.port = port;
    }

    public static void main(String[] args) {
        if (args.length < 2) return;
        String hostname = args[0];
        int port = Integer.parseInt(args[1]);
        ClientChat client = new ClientChat(hostname, port);
        client.execute();
    }
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
public void execute() {
    try {
        Socket socket = new Socket(hostname, port);
        System.out.println("Connected to the chat server");
        new WriteThread(socket, this).start();
        new ReadThread(socket, this).start();
    } catch (UnknownHostException ex) {
        System.out.println("Server not found: " + ex.getMessage());
    } catch (IOException ex) {
        System.out.println("I/O Error: " + ex.getMessage());
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
    void setUserName(String userName) {
        this.userName = userName;
    }

    String getUserName() {
        return this.userName;
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
public class ReadThread extends Thread {
    private BufferedReader reader;
    private Socket socket;
    private ClientChat client;
    public ReadThread(Socket socket, ClientChat client) {
        this.socket = socket;
        this.client = client;
         try {
            InputStream input = socket.getInputStream();
            reader = new BufferedReader(new InputStreamReader(input));
        } catch (IOException ex) {
            System.out.println("Error getting input stream: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
    public void run() {
        while (true) {
            try {
                String response = reader.readLine();
                System.out.println("\n" + response);
                // prints the username after displaying the server's message
                if (client.getUserName() != null) {
                    System.out.print("[" + client.getUserName() + "]: ");
                }
            } catch (IOException ex) {
                System.out.println("Error reading from server: " + ex.getMessage());
                ex.printStackTrace();
                break;
            }
        }
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
public class WriteThread extends Thread {
    private PrintWriter writer;
    private Socket socket;
    private ClientChat client;
    public WriteThread(Socket socket, ClientChat client) {
        this.socket = socket;
        this.client = client;
        try {
            OutputStream output = socket.getOutputStream();
            writer = new PrintWriter(output, true);
        } catch (IOException ex) {
            System.out.println("Error getting output stream: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

```
    public void run() {
        System.out.println("Enter your username: ");
        Scanner scanner =  new Scanner(System.in);
        String userName = scanner.nextLine();
        client.setUserName(userName);
        writer.println(userName);
        String text;
        do {
            text = scanner.nextLine();
            writer.println(text);
            System.out.print("[" + userName + "]: ");
        } while (!text.equals("bye"));
        try {
            socket.close();
        } catch (IOException ex) {
            System.out.println("Error writing to server: " + ex.getMessage());
        }
    }
}
```

Source: https://www.codejava.net/java-se/networking/how-to-create-a-chat-console-application-in-java-using-socket

Homework

Bài 1: Viết chương trình đọc 1 file text, tách các câu trong file (mỗi câu cách nhau bởi các dấu "." ";", "!", "?") và ghi ra các câu đọc được ra một file text mới theo định dạng là mỗi dòng 1 câu.

Bài 2: Viết chương trình kiểm tra xem 1 từ có xuất hiện trong file text hay không và đếm số lần xuất hiện của từ đó.

Bài 3: Viết chương trình đếm số lần xuất hiện của các từ khác nhau trong một file text.

Input: một hay nhiều files dạng văn bản

Output: từ điển gồm các từ khác nhau và tần suất xuất hiện của chúng, dạng HashMap [key, value]

Bài 4. Viết chương trình loại bỏ các stop words ra khỏi một câu văn bản tiếng Việt.

- Input: một câu + 1 file chứa các stop words
- Output: Các từ khác nhau trong câu và không có tính stop word. Mỗi từ quy ước cách nhau bằng khoảng trắng

Bài 5: Viết chương trình mã hóa một văn bản thành một vector n chiều

- Input: kho dữ liệu là một thư mục chứa nhiều file dạng văn bản
- Output: Mỗi file được biểu diễn bằng 1 vector n chiều.

(n: là số từ khác nhau trong kho dữ liệu; giá trị mỗi chiều là 1 nếu từ đó tồn tại trong file, ngược lại là 0)

Homework

Bài 6: Viết chương trình mã hóa một văn bản thành một vector n chiều, với trọng số của mỗi chiều là tf*idf (tf: term frequency; idf; inverse document frequency)

- Input: kho dữ liệu là một thư mục chứa nhiều file dạng văn bản
- Output: Mỗi file được biểu diễn bằng 1 vector n chiều.

(n: là số từ khác nhau trong kho dữ liệu; giá trị mỗi chiều là tf*idf)

Bài 7. Viết chương trình chat qua LAN dùng Java Socket

Bài 8. Viết chương trình chat qua internet

Bài 9. Viết chương trình đọc và hiển thị cấu trúc file, thư mục trên máy.

References

1. https://docs.oracle.com/javase/tutorial/essential/io/streams.html
2. https://docs.oracle.com/javase/tutorial/essential/io/index.html
3. https://docs.oracle.com/javase/tutorial/essential/io/QandE/questions.html
4. http://www.oracle.com/technetwork/java/socket-140484.html
5. https://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html
6. https://www.tutorialspoint.com/java/java_files_io.htm
7. https://www.javatpoint.com/java-io