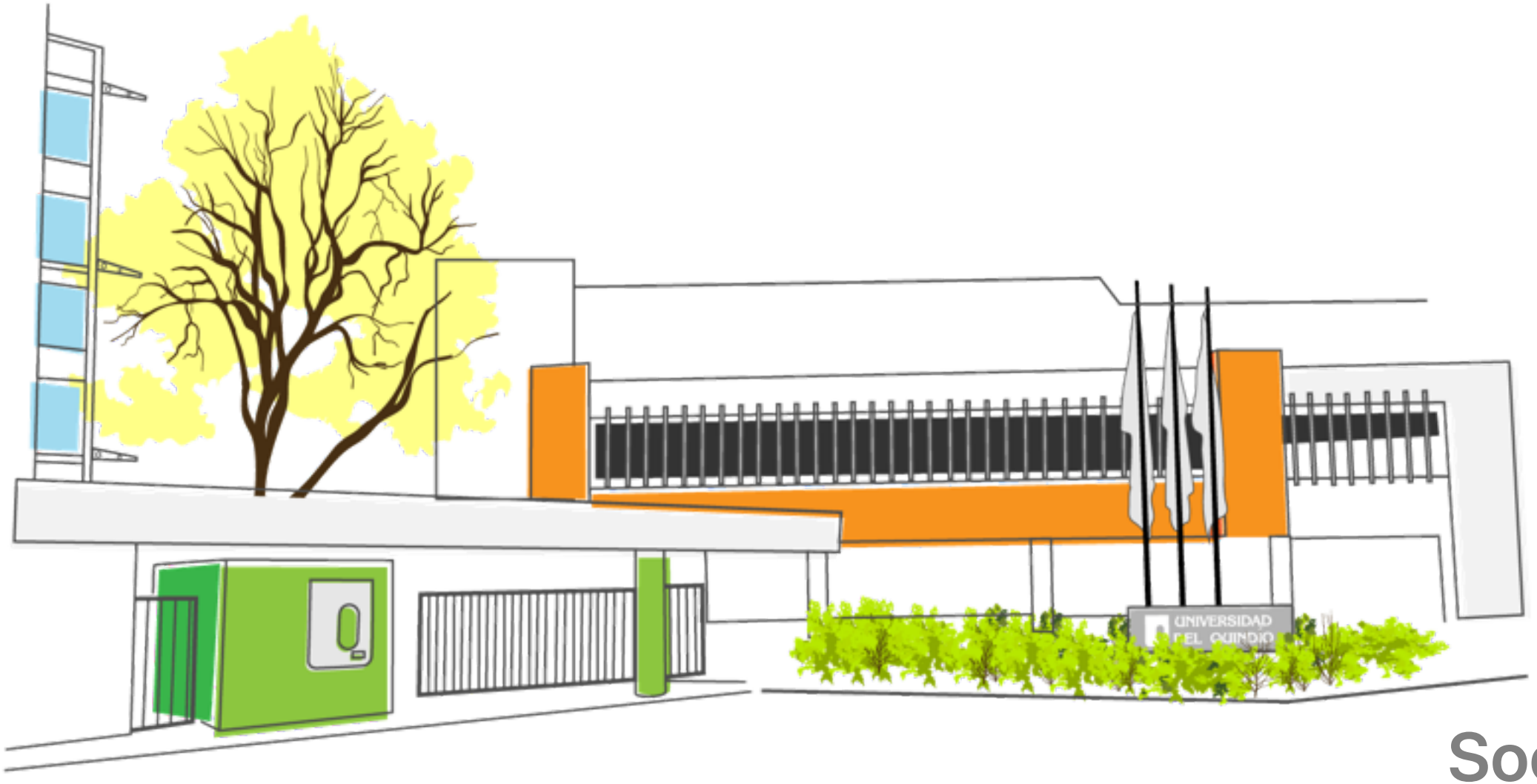
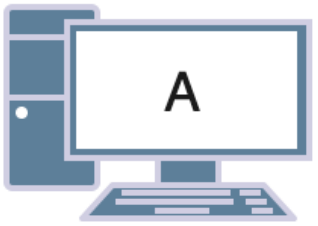
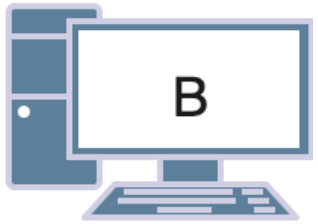


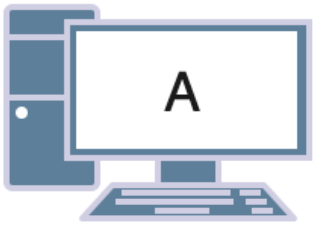
Sockets

- Introducción
- Mensaje
- Canal de comunicación (Flujos)
- Jerarquía de los flujos
- Hola Mundo

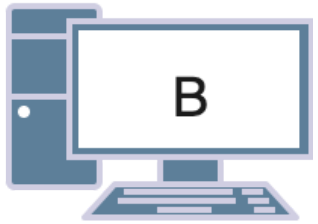


¿Cómo envío un mensaje de A a B ?

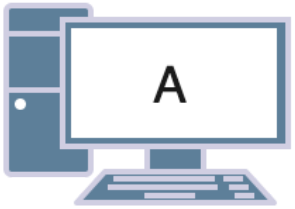




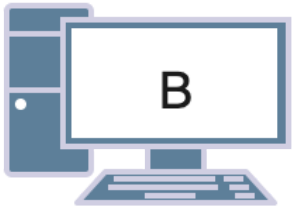
¿Cómo envío un mensaje de A a B ?
¿Dónde está B ?

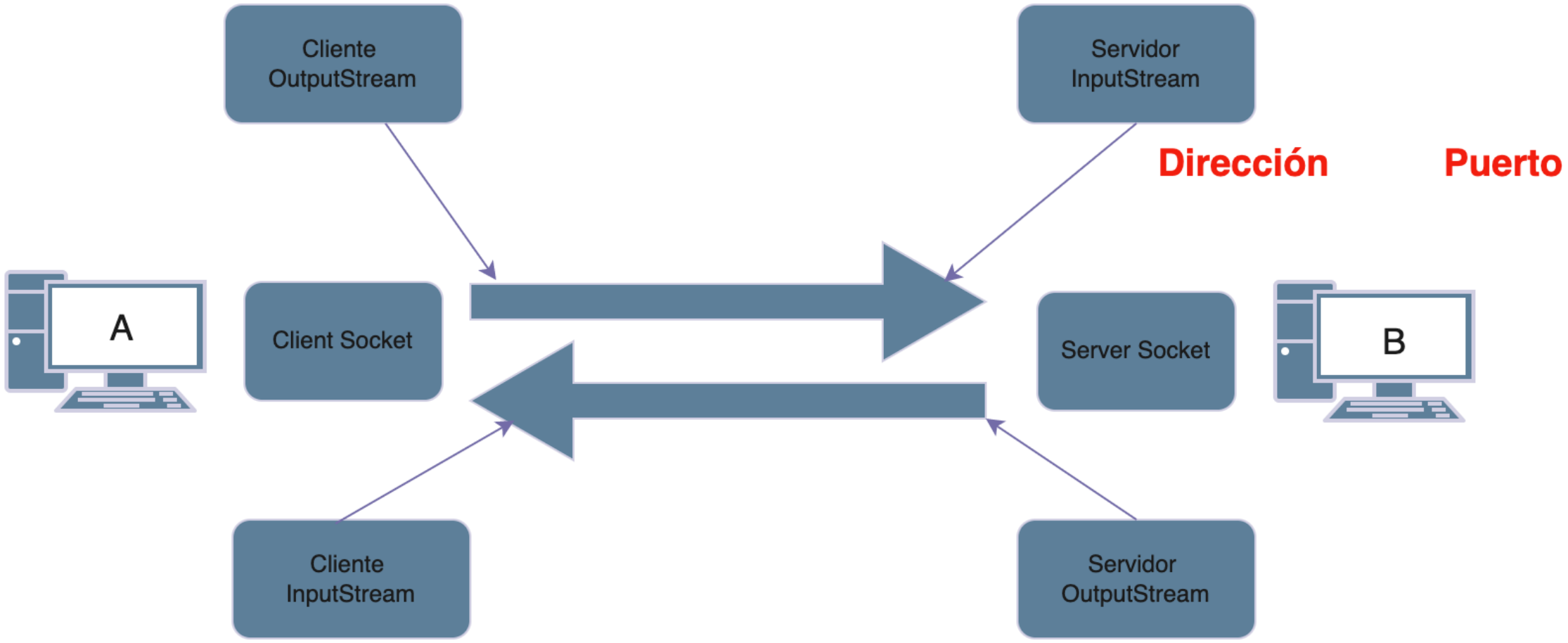


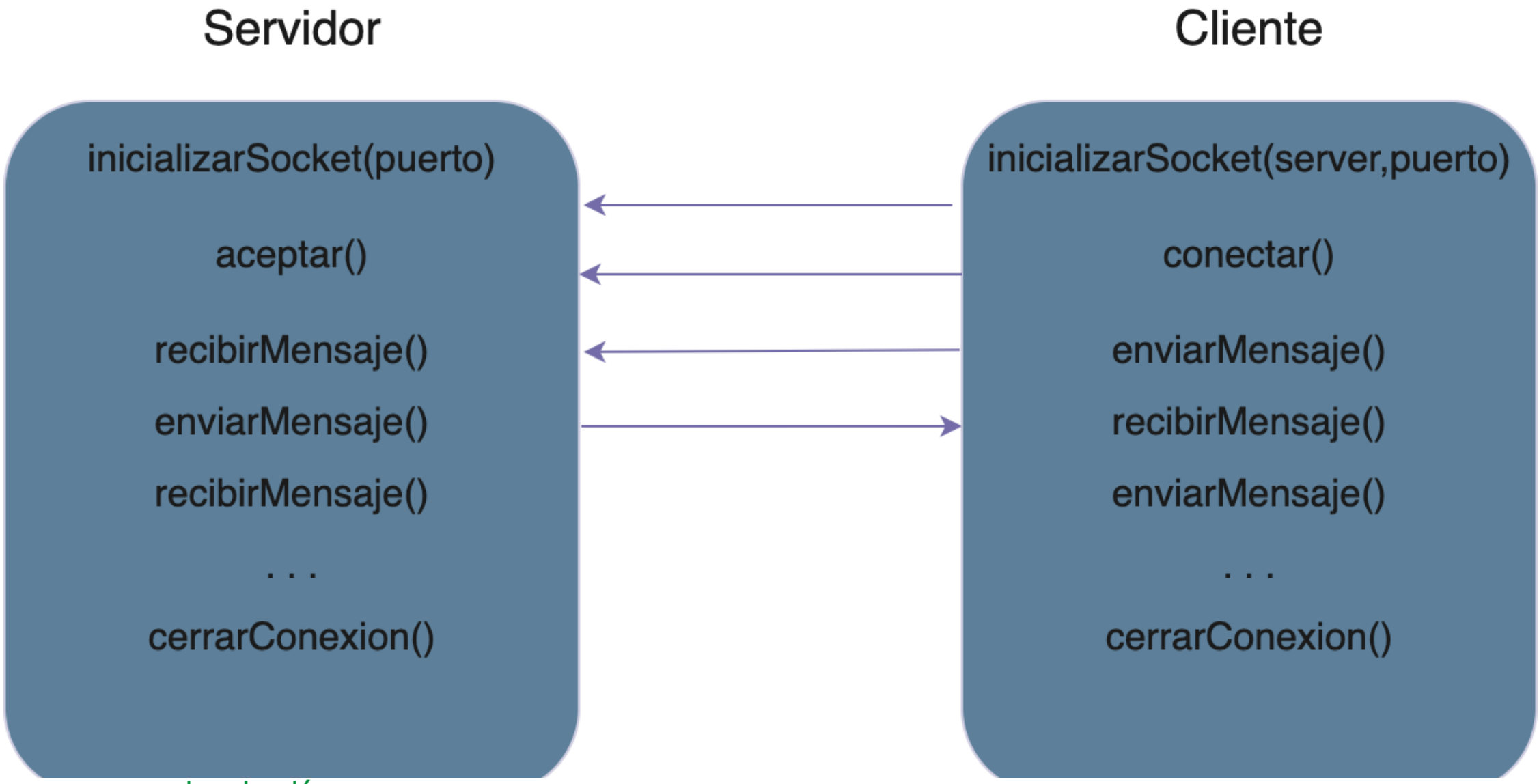
Dirección **Puerto**

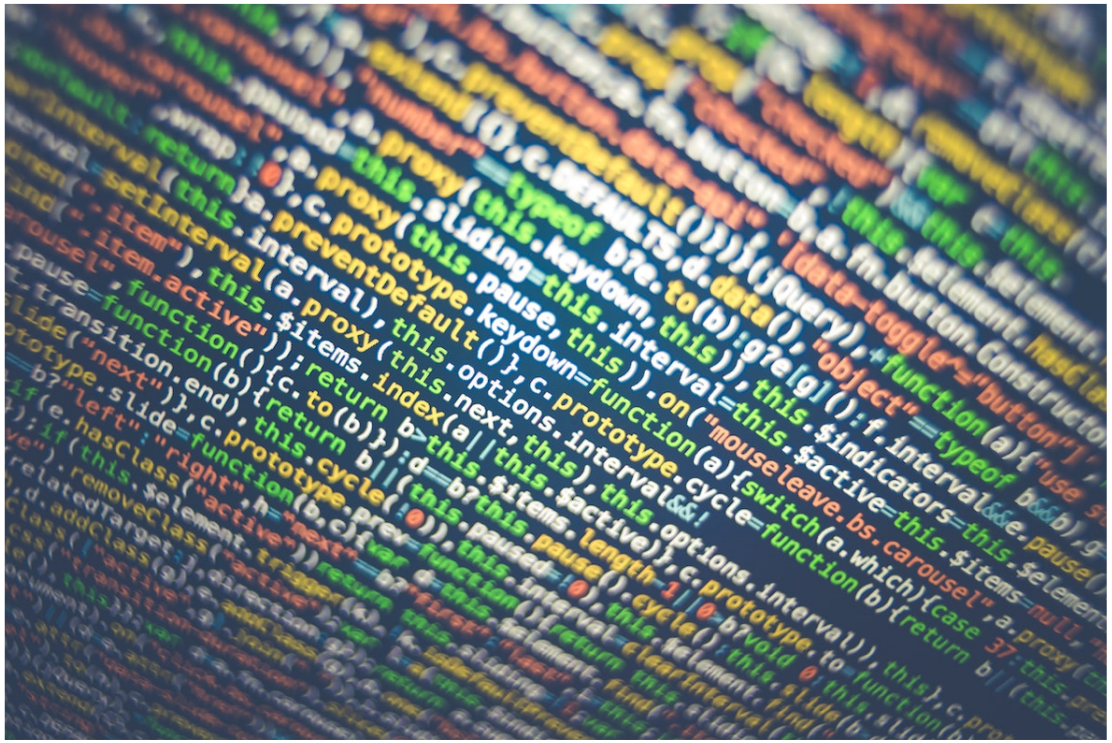


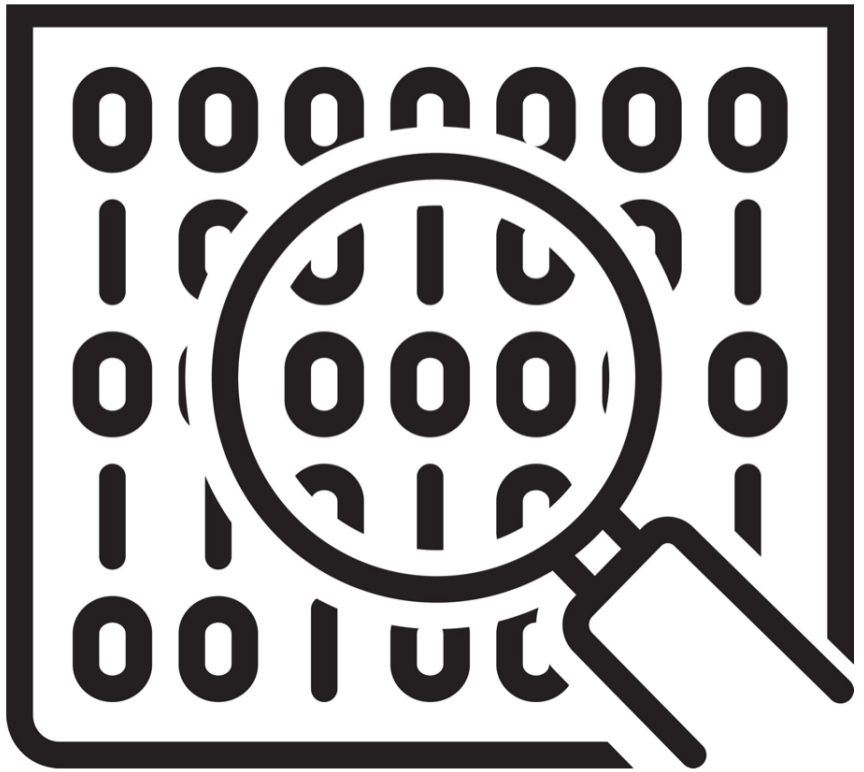
¿Cómo envío un mensaje de A a B ?
¿Dónde está B ?




Cliente
OutputStream
Servidor
InputStream
Dirección
Puerto
A
Client Socket
Server Socket
B
Cliente
InputStream
Servidor
OutputStream

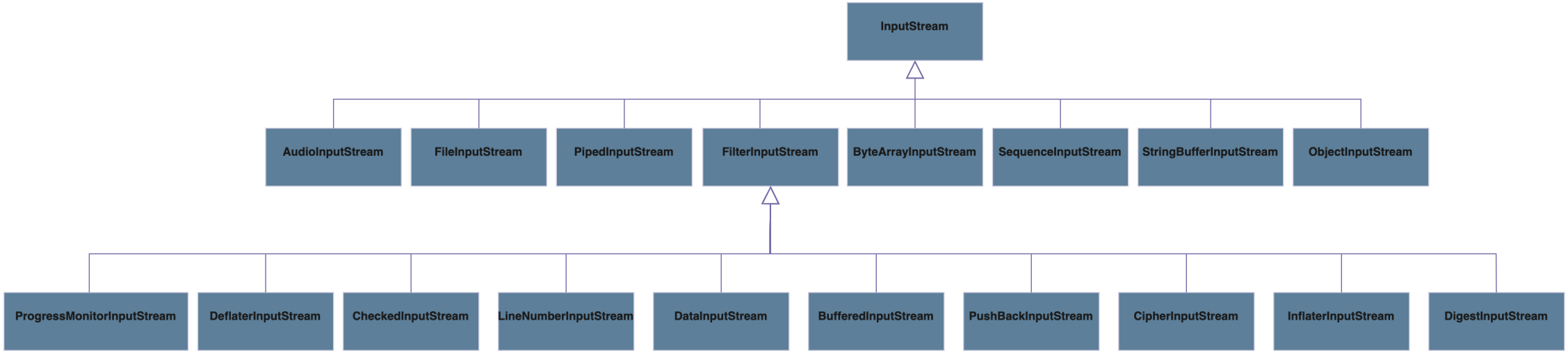

Servidor
Cliente
inicializarSocket(puerto)
aceptar()
recibirMensaje()
enviarMensaje()
recibirMensaje()
. . .
cerrarConexion()
inicializarSocket(server,puerto)
conectar()
enviarMensaje()
recibirMensaje()
enviarMensaje()
. . .
cerrarConexion()

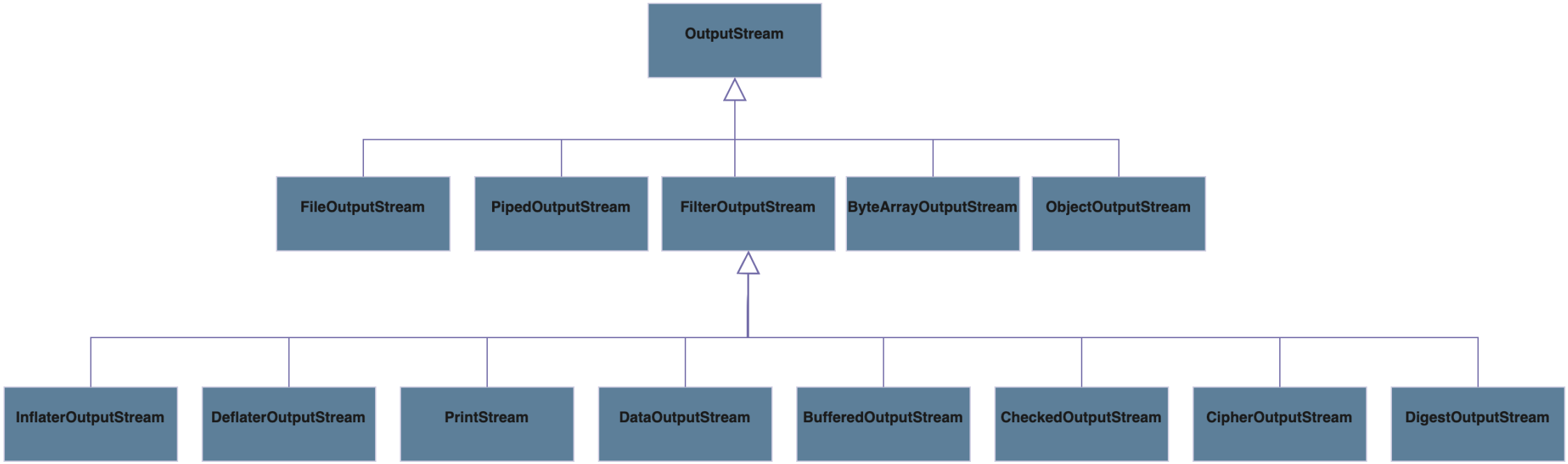


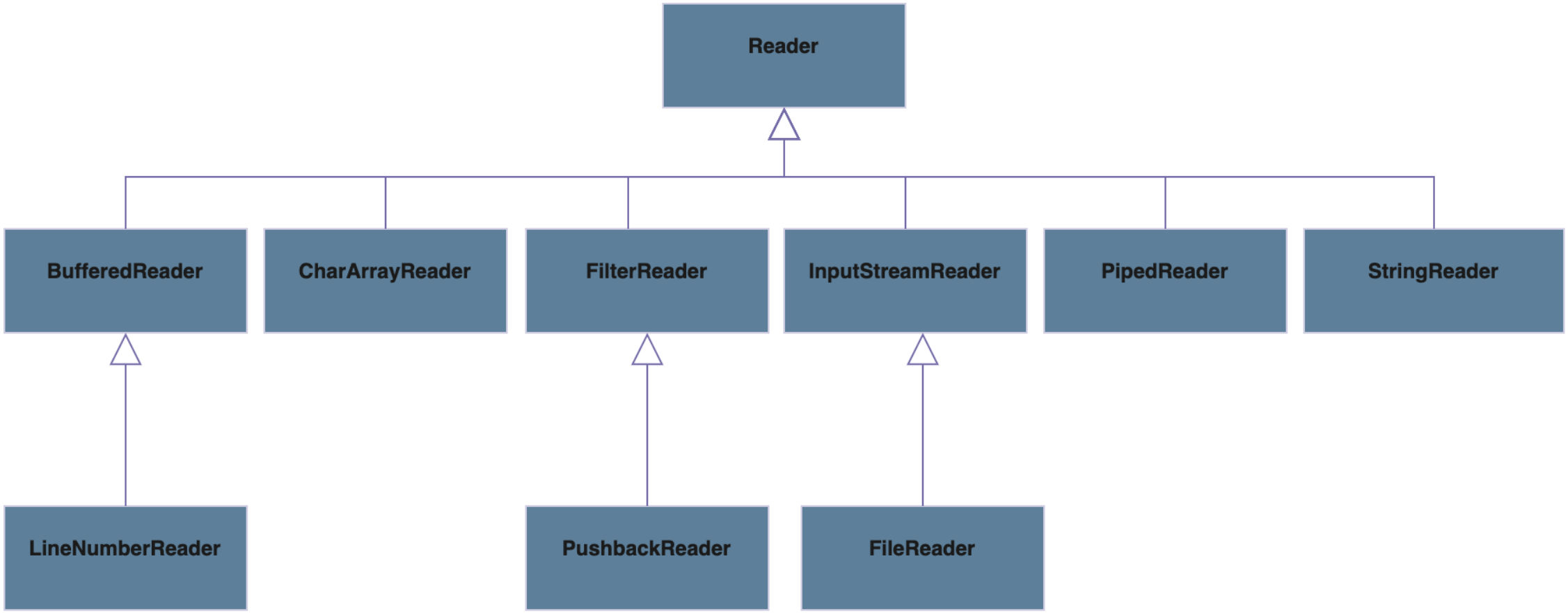


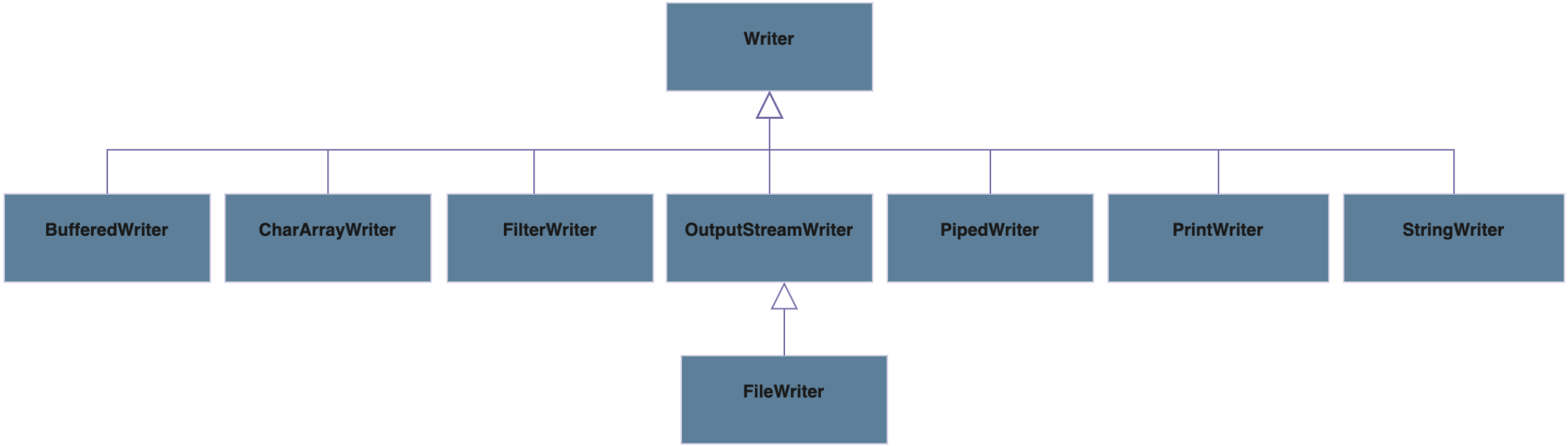
- Formato del mensaje
  - Texto
  - Binario
- Estructura del mensaje
  - sumar|23|35 (11 bytes)
  - 000000010001011100100011 (3 bytes)
- Dialogo
  - Orden de intervención en la comunicación
  - Mensajes a intercambiar

- Cliente
  - InputStream
  - OutputStream
- Servidor
  - InputStream
  - OutputStream


InputStream
AudioInputStream
FileInputStream
PipedInputStream
FilterInputStream
ByteArrayInputStream
SequenceInputStream
StringBufferInputStream
ObjectInputStream
ProgressMonitorInputStream
DeflaterInputStream
CheckedInputStream
LineNumberInputStream
DataInputStream
BufferedInputStream
PushBackInputStream
CipherInputStream
InflaterInputStream
DigestInputStream


OutputStream
FileOutputStream
PipedOutputStream
FilterOutputStream
ByteArrayOutputStream
ObjectOutputStream
InflaterOutputStream
DeflaterOutputStream
PrintStream
DataOutputStream
BufferedOutputStream
CheckedOutputStream
CipherOutputStream
DigestOutputStream


Reader
BufferedReader
CharArrayReader
FilterReader
InputStreamReader
PipedReader
StringReader
LineNumberReader
PushbackReader
FileReader


Writer
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
PipedWriter
PrintWriter
StringWriter
FileWriter

Hola Mundo

```java
public class App {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

```
public class App {
    public static void main(String[] args) throws IOException {
        final PrintStream out = System.out;
        final BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        out.println("Ingrese su nombre:");
        final var nombre = in.readLine();
        out.println("Hello "+nombre);
    }
}
```

```java
public class App {
    public static void main(String[] args) throws IOException {
        final PrintStream out = System.out;
        final BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        saludar(out, in);
    }

    public static void saludar(PrintStream out, BufferedReader in) throws IOException {
        out.println("Ingrese su nombre:");
        final var nombre = in.readLine();
        out.println("Hello "+nombre);
    }
}
```

Creación del servidor

```
final int puerto = 5555;
ServerSocket serverSocket = new ServerSocket(puerto);
final var clientSocket = serverSocket.accept();
```

Obtención de flujos de entrada y salida de un Socket

```
clientSocket.getOutputStream();
clientSocket.getInputStream();
```

Obtención de flujos de entrada y salida de un Socket

```
final PrintStream out = new PrintStream(clientSocket.getOutputStream());
final BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

Servidor

```java
public class ServerApp {

    public static void main(String[] args) throws IOException {
        final int puerto = 5555;
        ServerSocket serverSocket = new ServerSocket(puerto);
        final var clientSocket = serverSocket.accept();

        final PrintStream out = new PrintStream(clientSocket.getOutputStream());
        final BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

        saludar(out,in);

        clientSocket.close();
        serverSocket.close();
    }

    public static void saludar(PrintStream out, BufferedReader in) throws IOException {
        out.println("Ingrese su nombre:");
        final var nombre = in.readLine();
        out.println("Hello "+nombre);
    }

}
```

Cliente

```
public class ClientApp {
    public static void main(String[] args) throws IOException {
        final String server = "localhost";
        final int port = 5555;

        try(final var socket = new Socket(server,port)) {
            final PrintStream outScreen = System.out;
            final PrintStream outServer = new PrintStream(socket.getOutputStream());
            final BufferedReader inServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            final BufferedReader inKeyBoard = new BufferedReader(new InputStreamReader(System.in));

            final String mensaje = inServer.readLine();
            outScreen.println(mensaje);

            final var nombre = inKeyBoard.readLine();
            outServer.println(nombre);

            final String respuesta = inServer.readLine();
            outScreen.println(respuesta);
        }
    }
}
```

- Cree un servidor que al aceptar un cliente:
  - Lea un texto con el formato: hola
  - Responda con el texto Bienvenido
  - Lea una texto que puede ser:
    - FECHA
    - HORA
    - BYE
  - En caso de haber recibido el texto FECHA debe responder con la fecha en formato YYYY-MM-DD y repite la lectura de texto
  - En caso de haber recibido el texto HORA debe responder con la fecha en formato HH:MM y repite la lectura de texto
  - En caso de haber recibido el texto BYE finaliza la conexión

- Cree un cliente que permita interactuar con el servidor anterior.
- Usando su cliente conectese al servidor construido por uno de sus compañeros.
- Use su servidor para atender los clientes construidos por otros compañeros.

- Para enviar un objeto a través de un flujo es necesario serializarlo.
- Implementar la interfaz Serializable.
- Todos sus atributos deben ser serializables.
- Los que no se desean serializar se marcan con el modificador trasient

- https://github.com/christiancandela/sockets-202301.git

UNIVERSIDAD DEL QUINDÍO

ACREDITADA DE ALTA CALIDAD

PERTINENTE
CREATIVA
INTEGRADORA

MUCHAS GRACIAS

@uniquindio universidaddelquindio universidad_del_quindio