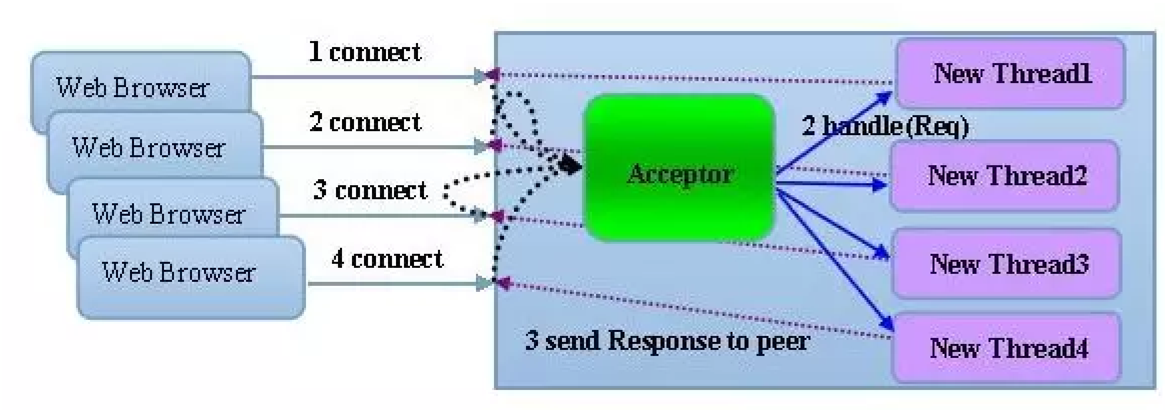
Java BIO 认识

Original 2018-04-15 JamesCSH JamesCSH

本文简单介绍 Java 中的 BIO 知识。

1 BIO 通信模型

BIO 通信模型如下图所示：

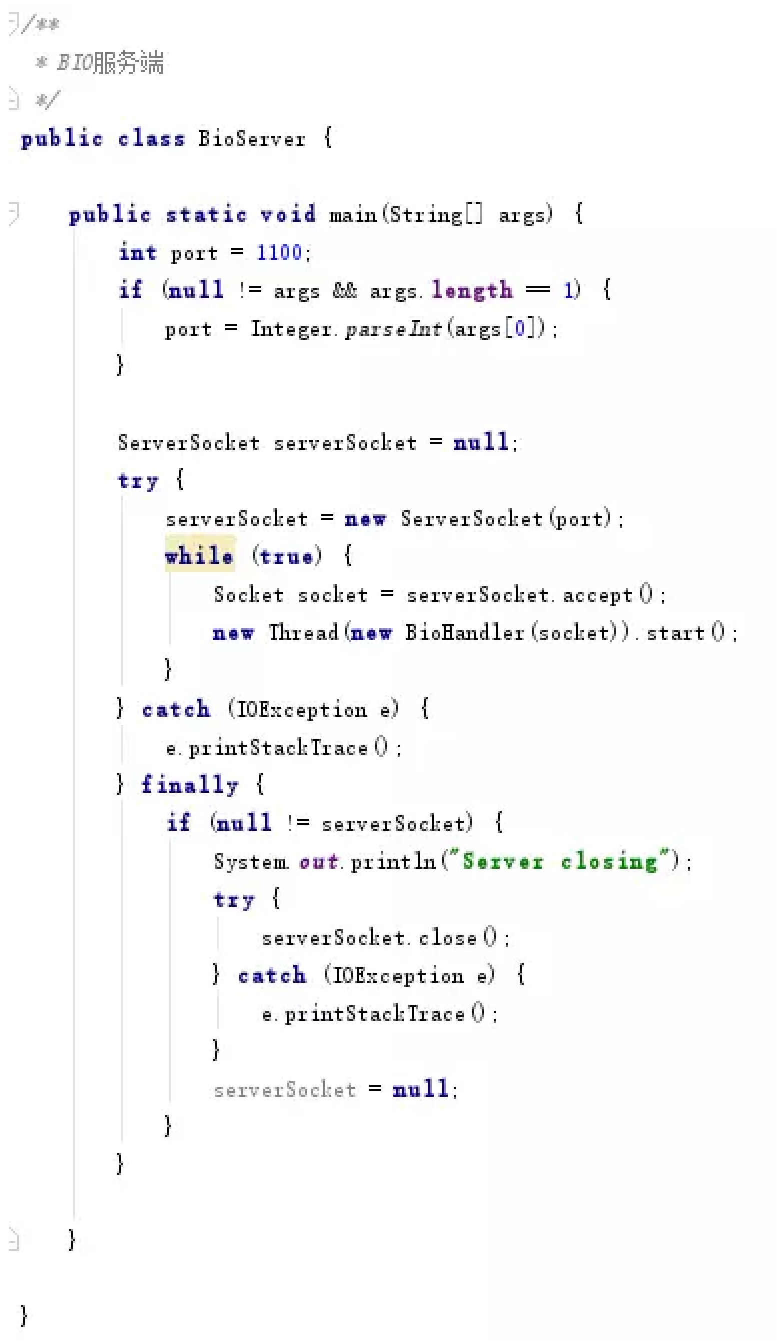


BIO 是一个一请求一应答的通讯模型，怎么理解一请求一应答呢？

服务端由一个独立的 Acceptor 线程负责监听所有客户端的连接，接收到客户端连接请求之后为每个客户端创建一个新的线程进行处理，处理完成之后，通过输出流返回应答给客户端，然后线程销毁。也即是一个客户端请求会对应一个服务端线程来处理。

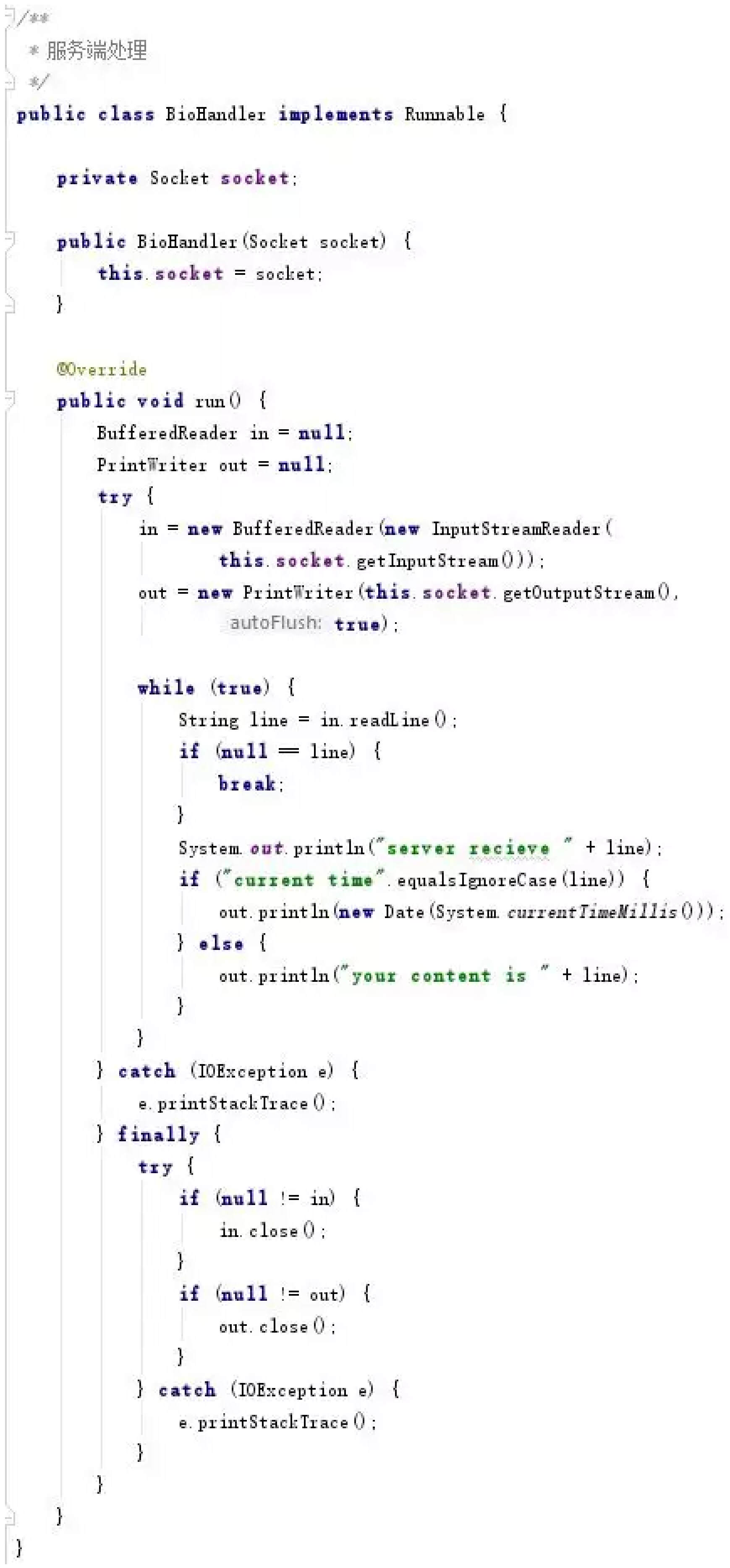
2 BIO 的一个简单例子

- BioServer 作为服务端。创建一个 ServerSocket 绑定 IP，调用 accept() 来接收客户端的请求，每接收到一个客户端的请求，就新创建一个线程来处理。



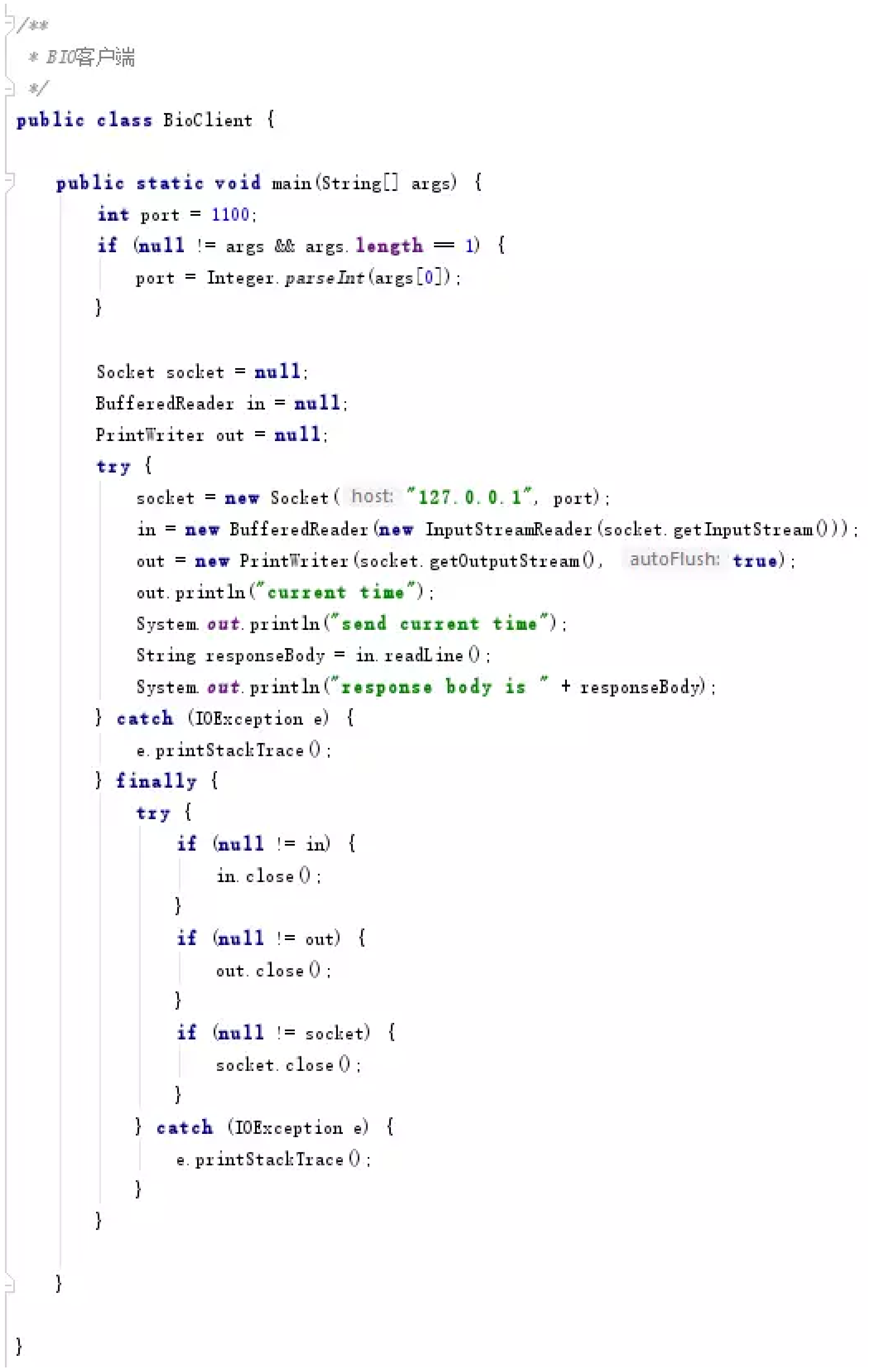
```
/**
 * BIO服务端
 */
public class BioServer {

    public static void main(String[] args) {
        int port = 1100;
        if (null != args && args.length == 1) {
            port = Integer.parseInt(args[0]);
        }

        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(port);
            while (true) {
                Socket socket = serverSocket.accept();
                new Thread(new BioHandler(socket)).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (null != serverSocket) {
                System.out.println("Server closing");
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                serverSocket = null;
            }
        }

    }

}
```

- BioHandler 用来处理客户端的请求。通过 BufferedReader 接收客户端请求的输入内容，用 PrintWriter 返回输出内容到客户端。下面代码实现如果客户端发送的内容是 current time，则返回当前的时间。



```java
/**
 * 服务端处理
 */
public class BioHandler implements Runnable {

    private Socket socket;

    public BioHandler(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        BufferedReader in = null;
        PrintWriter out = null;
        try {
            in = new BufferedReader(new InputStreamReader(
                    this.socket.getInputStream()));
            out = new PrintWriter(this.socket.getOutputStream(),
                     autoFlush: true);

            while (true) {
                String line = in.readLine();
                if (null == line) {
                    break;
                }
                System.out.println("server recieve " + line);
                if ("current time".equalsIgnoreCase(line)) {
                    out.println(new Date(System.currentTimeMillis()));
                } else {
                    out.println("your content is " + line);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (null != in) {
                    in.close();
                }
                if (null != out) {
                    out.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

- BioClient 作为客户端，连接服务端，然后通过 PrintWriter 向服务端发送数据，使用 BufferedReader 接收服务端返回的数据。



```java
/**
 * BIO客户端
 */
public class BioClient {

    public static void main(String[] args) {
        int port = 1100;
        if (null != args && args.length == 1) {
            port = Integer.parseInt(args[0]);
        }

        Socket socket = null;
        BufferedReader in = null;
        PrintWriter out = null;
        try {
            socket = new Socket( host: "127.0.0.1", port);
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream(),  autoFlush: true);
            out.println("current time");
            System.out.println("send current time");
            String responseBody = in.readLine();
            System.out.println("response body is " + responseBody);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (null != in) {
                    in.close();
                }
                if (null != out) {
                    out.close();
                }
                if (null != socket) {
                    socket.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

    }

}
```

3 BIO 弊端

每一个新的客户端请求接入时，服务端必须创建一个新的线程处理新接入的客户端链路，一个服务端线程只能处理一个客户端连接。在高并发的情况下，容易占用大量资源，无法高效的处理。