003.基于TCP协议的网络编程

利用 TCP 协议进行通信的两个应用程序是有主次之分的，

一个是服务器程序，一个是客户端程序，

两者的功能和编写方法不太一样，

其中 ServerSocket 类表示 Socket 服务器端，Socket 类表示 Socket 客户端。

基于TCP的套接字编程实现流程：

1. 服务器端流程：

（1）创建套接字（ServerSocket）

（2）将套接字绑定到一个本地地址和端口上（bind）

（3）将套接字设定为监听模式，准备接受客户端请求（listen）

（4）阻塞等待客户端请求到来。当请求到来后，接受连接请求，返回一个新的对应于此客户端连接的套接字sockClient（accept）

（5）用返回的套接字sockClient和客户端进行通信（send/recv）；

（6）返回，等待另一个客户端请求（accept）

（7）关闭套接字（close）

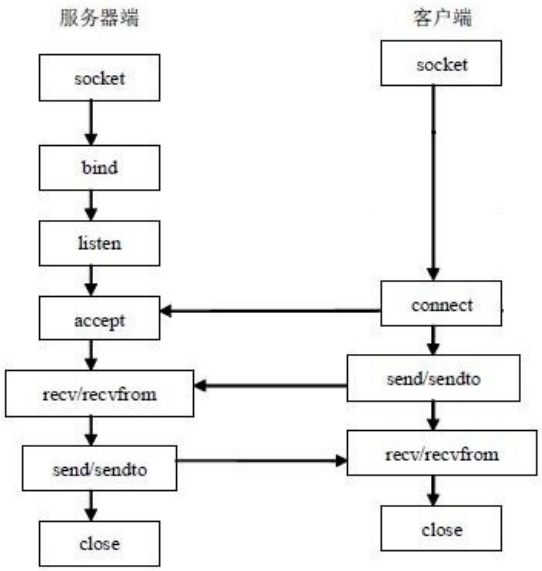
2. 客户端流程：

（1） 创建套接字（socket）

（2） 向服务器发出连接请求（connect）

（3） 和服务器进行通信（send/recv）

（4） 关闭套接字（close）



ServerSocket 类

用于在服务器上开一个端口，被动地等待数据（使用 accept() 方法）并建立连接进行数据交互。

服务器套接字一次可以与一个套接字连接，如果多台客户端同时提出连接请求，

请求连接的客户端会被存入一个队列中，然后从中取出一个套接字与服务器新建的套接字连

接起来。
若请求连接大于最大容纳数，则多出的连接请求被拒绝；默认的队列大小是 50。

下面简单介绍一下 ServerSocket 的构造方法和常用方法。

ServerSocket 的构造方法

ServerSocket 的构造方法如下所示。

- ServerSocket()：无参构造方法。
- ServerSocket(int port)：创建绑定到特定端口的服务器套接字。
- ServerSocket(int port,int backlog)：使用指定的 backlog 创建服务器套接字并将其绑定到指定的本地端口。
- ServerSocket(int port,int backlog,InetAddress bindAddr)：使用指定的端口、监听 backlog 和要绑定到本地的 IP 地址创建服务器。

创建 ServerSocket 时可能会抛出 IOException 异常，所以要进行异常捕捉。

```
try{
    ServerSocket serverSocket=new ServerSocket(8111);
}
catch(IOException e){
    e.printStackTrace();
}
```

ServerSocket 的常用方法

ServerSocket 的常用方法如下所示。

- Server accept()：监听并接收到此套接字的连接。
- void bind(SocketAddress endpoint)：将 ServerSocket 绑定到指定地址（IP 地址和端口号）。
- void close()：关闭此套接字。
- InetAddress getInetAddress()：返回此服务器套接字的本地地址。
- int getLocalPort()：返回此套接字监听的端口。
- SocketAddress getLocalSocketAddress()：返回此套接字绑定的端口的地址，如果尚未绑定则返回 null。
- int getReceiveBufferSize()：获取此 ServerSocket 的 SO_RCVBUF 选项的值，该值是从 ServerSocket 接收的套接字的建议缓冲区大小。

调用 accept() 方法会返回一个和客户端 Socket 对象相连接的 Socket 对象，
服务器端的 Socket使用 getOutputStream() 方法获得的输出流
将指向客户端 Socket 使用 getInputStream() 方法获得那个输入流。

同样，服务器端的 Socket使用的 getInputStream() 方法获得的输入流将指向
客户端 Socket 使用的 getOutputStream() 方法获得的那个输出流。
也就是说，当服务器向输出流写入信息时，客户端通过相应的输入流就能读取，反之同样如此。

Socket 类

Socket 类表示通信双方中的客户端，用于呼叫远端机器上的一个端口，
主动向服务器端发送数据（当连接建立后也能接收数据）。

Socket 的构造方法

Socket 的构造方法如下所示。

- Socket()：无参构造方法。
- Socket(InetAddress address,int port)：创建一个流套接字并将其连接到指定 IP 地址的指定端口。
- Soclcet(InetAddress address,int port,InetAddress localAddr,int localPort)：创建一个套接字并将其连接到指定远程地址上的指定远程端口。
- Socket(String host,int port)：创建一个流套接字并将其连接到指定主机上的指定端口。
- Socket(String host,int port,InetAddress localAddr,int localPort)：创建一个套接字并将其连接到指定远程地址上的指定远程端口。

Socket 的常用方法

Socket 的常用方法如下所示。

- void bind(SocketAddress bindpoint)：将套接字绑定到本地地址。
- void close()：关闭此套接字。
- void connect(SocketAddress endpoint)：将此套接字连接到服务器。
- InetAddress getInetAddress()：返回套接字的连接地址。
- InetAddress getLocalAddress()：获取套接字绑定的本地地址。
- InputStream getInputStream()：返回此套接字的输入流。
- OutputStream getOutputStream()：返回此套接字的输出流。
- SocketAddress getLocalSocketAddress()：返回此套接字绑定的端点地址，如果尚未绑定则返回 null。
- SocketAddress getRemoteSocketAddress()：返回此套接字的连接的端点地址，如果尚未连接则返回 null。
- int getLoacalPort()：返回此套接字绑定的本地端口。
- int getPort()：返回此套接字连接的远程端口。

例子：客户端与服务器端的简单通信

首先来看一下服务器端的代码，如下所示：

```
public class Server {
    public static void main(String[] args) throws Exception {
        //创建一个ServerSocket，用于监听客户端的链接请求
        ServerSocket serverSocket = new ServerSocket();
        serverSocket.bind(new InetSocketAddress("127.0.0.1",30000));
```

```
        //采用循环不断地接受来自客户端的链接
        while (true){
            Socket socket = serverSocket.accept();
            //把socket对应的输出流包装成打印流
            PrintStream ps = new PrintStream(socket.getOutputStream(),true,"GBK");
            //进行普通的IO
            ps.println("这是服务器发出的信息");

            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream(),"GBK"));
            System.out.println(in.readLine());

            ps.close();
            socket.close();
        }
    }
}
```

再来看一个客户端的代码，如下所示：

```
public class Client {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket();
        socket.connect(new InetSocketAddress("127.0.0.1",30000));

        //把socket的输入流包装为BufferedReader
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream(),"GBK"));
        //读取服务器端发送来的数据
        String str = in.readLine();
        System.out.println("来自服务器的信息："  + str);

        PrintStream ps = new PrintStream(socket.getOutputStream(),true,"GBK");
        ps.println("客户端的信息");

        in.close();
        socket.close();
    }
}
```