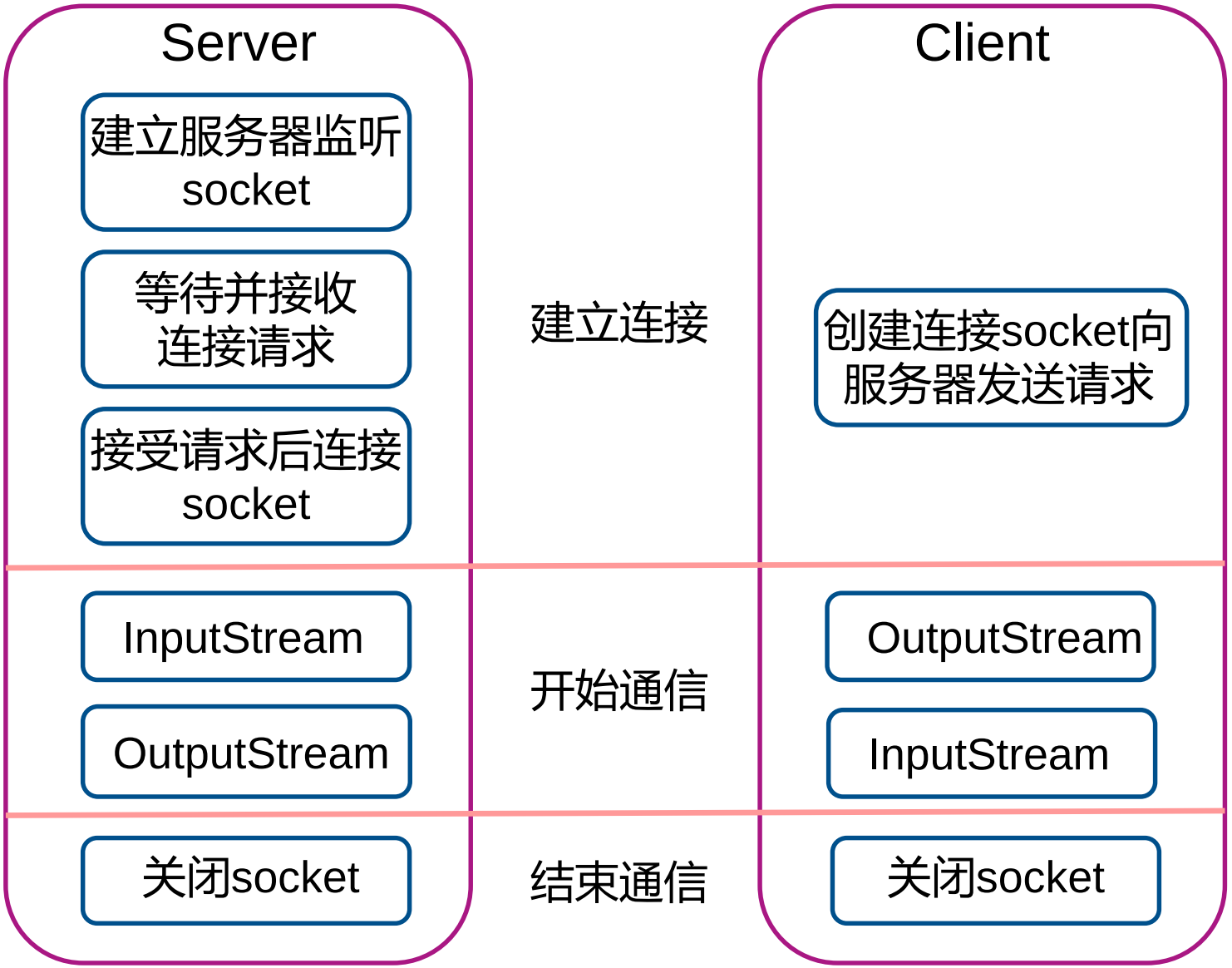
面向对象的软件构造实践

实验三

2022春

哈尔滨工业大学（深圳）

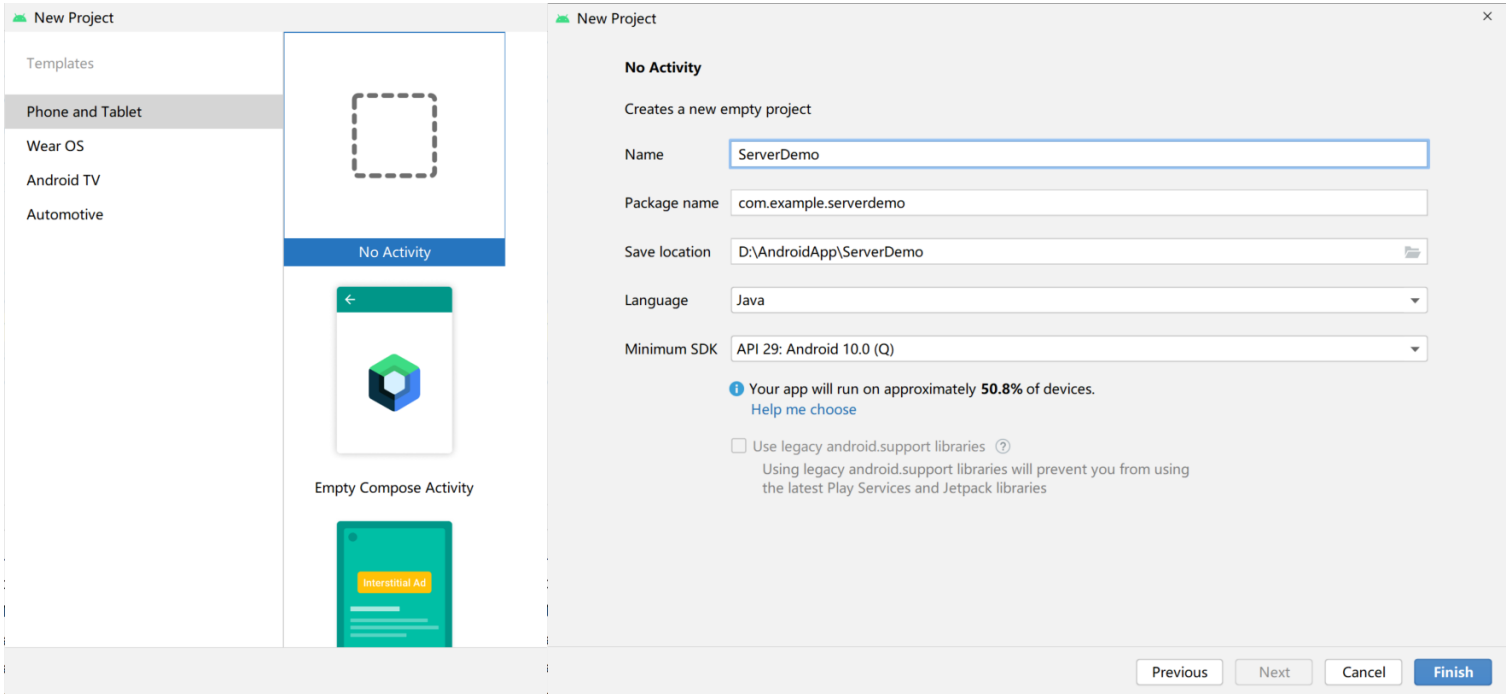
Socket通信过程


Server
建立服务器监听socket
等待并接收连接请求
接受请求后连接socket
InputStream
OutputStream
关闭socket
建立连接
开始通信
结束通信
Client
创建连接socket向服务器发送请求
OutputStream
InputStream
关闭socket

实验任务

1 创建服务器程序

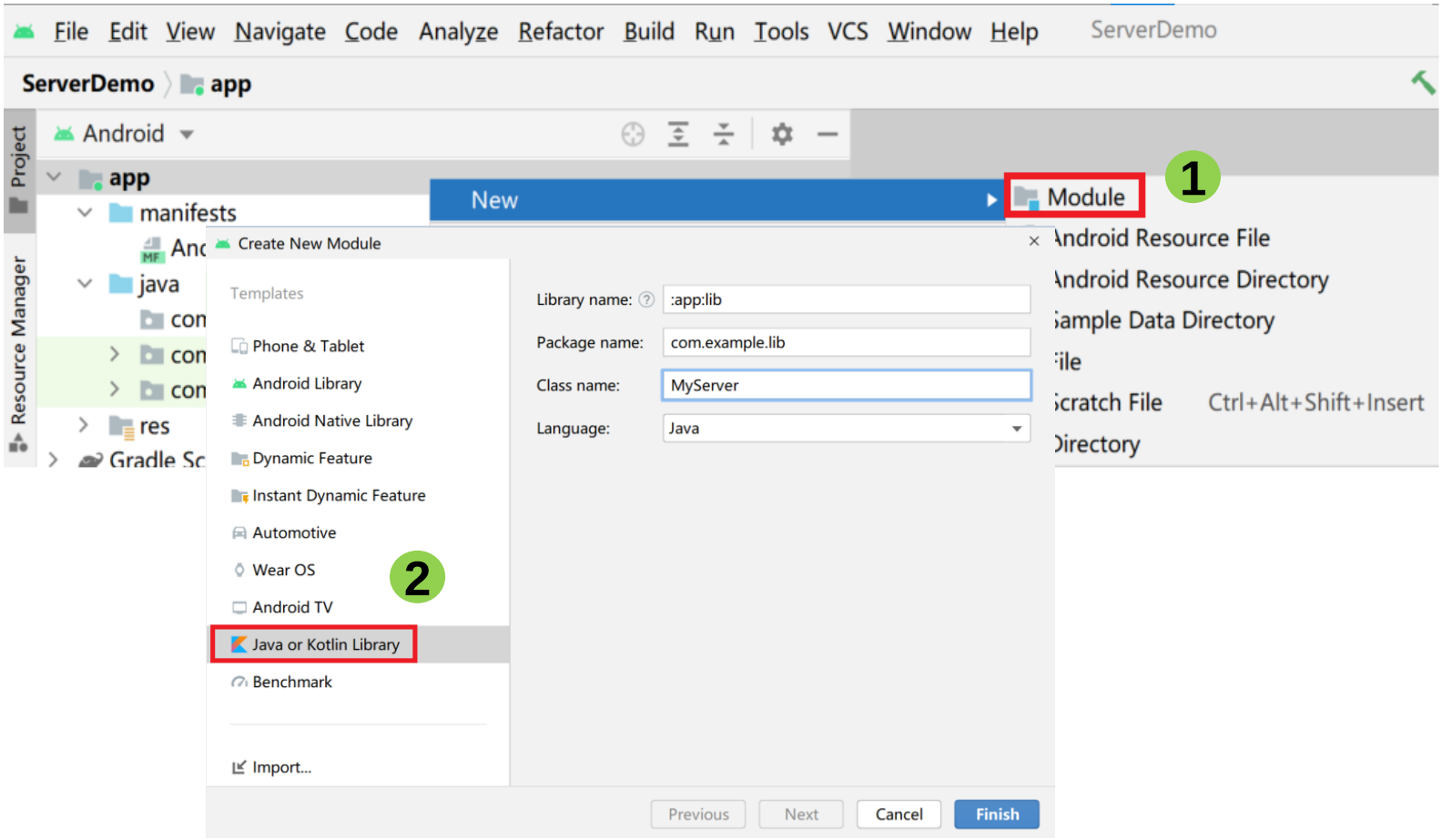
- 在Android Studio中选择No Activity模板，输入项目名称ServerDemo，其它选项可选择默认。



实验任务

1 创建服务器程序

- 创建Java模块



实验任务

1 创建服务器程序

- 创建服务器代码

```
public  MyServer(){
    try{
        InetAddress addr = InetAddress.getLocalHost();
        System.out.println("local host:" + addr);

        //创建server socket
        ServerSocket serverSocket = new ServerSocket( port: 9999)
        System.out.println("listen port 9999");

        while(true){
            System.out.println("waiting client connect");
            Socket socket = serverSocket.accept();
            System.out.println("accept");
            new Thread(new Service(socket)).start();
        }
    }catch (Exception ex){
        ex.printStackTrace();
    }
}
```

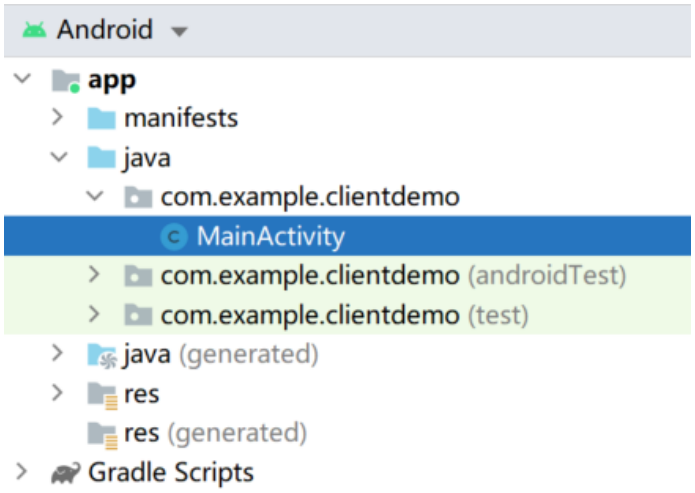
Service类

```
@Override
public void run() {
    System.out.println("wait client message " );
    try {
        while ((content = in.readLine()) != null) {
            System.out.println("message from client:"+content);
            if(content.equals("bye")){
                System.out.println("disconnect from client,close socket");
                socket.shutdownInput();
                socket.shutdownOutput();
                socket.close();
            }else {
                this.sendMessge(socket);
            }
        }
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
public void sendMessge(Socket socket) {
    PrintWriter pout = null;
    try{
        String message = "hello,client!";
        System.out.println("messge to client:" + message);
        pout = new PrintWriter(new BufferedWriter(
                new OutputStreamWriter(socket.getOutputStream())), autoFlush: true);
        pout.println(message);
    }catch (IOException ex){
        ex.printStackTrace();
    }
}
```

实验任务

2 创建客户端程序

- 创建ClientDemo项目，设计布局，在res/layout目录下的activity_main.xml中增加两个按钮



实验任务

2 创建客户端程序

- 在AndroidManifest.xml中声明网络访问权限

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.clientdemo">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ClientDemo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ClientDemo">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

实验任务

2 创建客户端程序

- 开发客户端代码

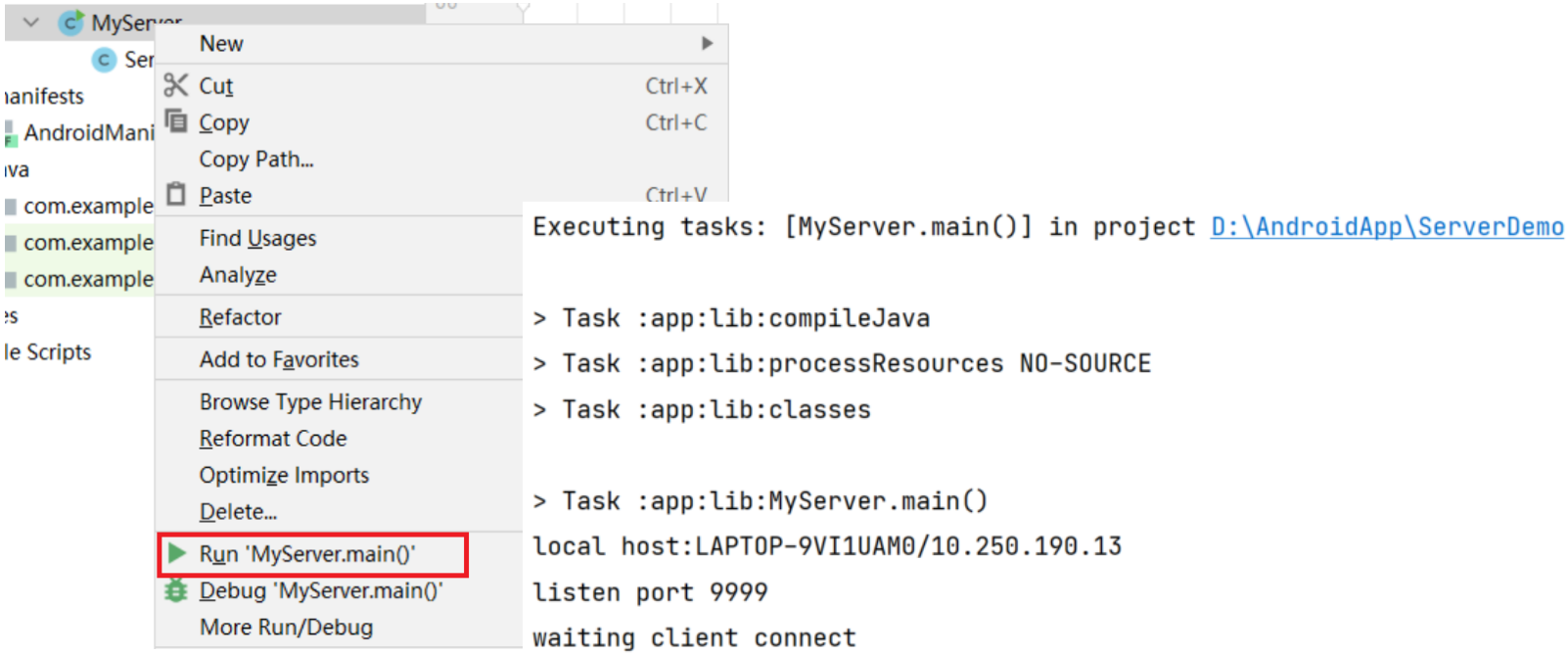
```
@Override
public void onClick(View view){
    switch (view.getId()){
        case R.id.btnConn:
            new Thread(new NetConn()).start();
            break;
        case R.id.btnSend:
            new Thread(){
                @Override
                public void run(){
                    Log.i( tag: "client",  msg: "send message to server");
                    writer.println("hello,server!");
                }
            }.start();
            break;
        case R.id.btnDiscon:
            new Thread(){
                @Override
                public void run(){
                    Log.i( tag: "client", msg: "disconnect to server");
                    writer.println("bye");
                }
            }.start();
            break;
    }
}
```

```
protected class NetConn extends Thread{
    @Override
    public void run(){
        try{
            socket = new Socket();
            socket.connect(new InetSocketAddress
                    ( hostname: "10.250.190.13", port: 9999), timeout: 5000);
            writer = new PrintWriter(new BufferedWriter(
                    new OutputStreamWriter(
                            socket.getOutputStream(), charsetName: "UTF-8")), autoFlush: true);
            Log.i( tag: "client", msg: "connect to server");
        }catch(UnknownHostException ex){
            ex.printStackTrace();
        }catch(IOException ex){
            ex.printStackTrace();
        }
    }
}
```

实验任务

3 启动服务器程序

- 右键MyServer → Run，控制面板显示服务器监听9999端口，等待客户端连接



实验任务

4 启动客户端程序

- 点击Connect Server建立与服务器的连接，点击Send Message向服务器发送消息

```
Executing tasks: [MyServer.main()] in project D:\AndroidApp\ServerDemo

> Task :app:lib:compileJava
> Task :app:lib:processResources NO-SOURCE
> Task :app:lib:classes

> Task :app:lib:MyServer.main()
local host:LAPTOP-9VI1UAM0/10.250.190.13
listen port 9999
waiting client connect
accept client connectSocket[addr=/10.250.190.13,port=63735,localport=9999]
waiting client connect
wait client message
message from client:hello,server!
messge to client:hello,client!
```

实验任务

5 断开Socket连接

- 点击Disconnect断开Socket连接

```
> Task :app:lib:MyServer.main()
local host:LAPTOP-9VI1UAM0/10.250.190.13
listen port 9999
waiting client connect
accept client connectSocket[addr=/10.250.190.13,port=55726,localport=9999]
waiting client connect
wait client message
message from client:hello,server!
messge to client:hello,client!
message from client:bye
disconnect from client,close socket
```

同学们
请开始实验吧！