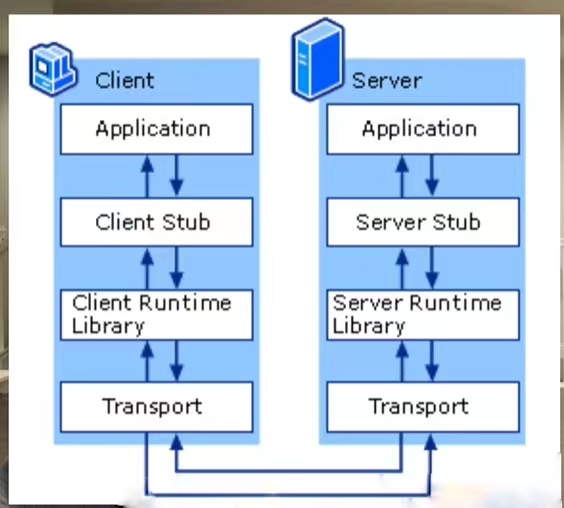
RPC（remote procedure call）：



Stub：将这个函数和变量编码成一种通用的数据格式，然后给服务端。服务端对这个数据进行解码，然后调用已经注册过的函数。这个函数运行之后的结果返回给这个client stub。然后这个Client sub就会进行解析返回的数据和错误。然后汇报给客户端。

GRPC：（基于HTTP/2.0开发的，性能高、接口规范、适合团队开发）

IDL(Interface Definition Language)：接口描述文件（proto buffer）

**proto buffer文件定义**：设置每个接口支持的请求，响应的type类型；通过grpc提供的toolbox来生成代码：比如说这个client stub和server stub。

**Grpc有四种不同的生命周期(类型)：**  
 1. Unary：客户发送请求，服务器直接进行回应

1. Client-side streaming：客户端不断上传请求到服务端（像一个流一样），服务端在这 个结束后只反应一个响应回去。
2. Server-side streaming：用户只上传一个请求，但是服务器源源不断的把这个响应传回给客户端。
3. Bidirectional streaming：双向都是以这个流的形式来进行通信。

**任务：**

描述一个向导服务，服务名为RouteGuide

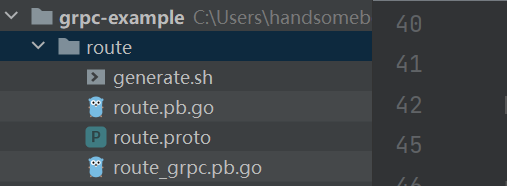
定义四种不同的信息类型（msg），分别为Point，Rectangle，Feature（某个点相关的信息），RouteSummary（用户上传的点会有一个路线图） 以及Chat。

定义四个方法：

1. GetFeature（输入一个Point， 返回这个点的Feature）
2. ListFeature（输入一个Rectangle， 输入流这个区域内所有的Feature）
3. RecordRoute（输入流为每个时间点的位置Point，返回一个RouteSummary）
4. Recommend（输入流RecommendationRequest，输出流Feature）

通过创建好的proto来创建server和client的代码桩：

protoc --go\_out=. --go\_opt=paths=source\_relative --go-grpc\_out=. --go-grpc\_opt=paths=source\_relative route.proto



然后自己写了个sh，不用每次都敲命令来调用proto

Proto里主要定义这个**message**和**Service**。

**生成的这两个文件的含义：**

**route.pb.go**定义对应一个message的代码桩；

**Route\_grpc.pb.go**定义你的**Server要实现的接口**以及你的**client怎么去调用这些接口**；

然后定义两个东西，一个route-client和route-server。在这里定义这个客户端和服务端。

**Client.go**

runFiret() // 上传一个点，返回在server里的点的feature（route\_grpc.pb.go里RouteGuideClient的GetFeature）

runSencond() // 给一个包含着点信息的list，判断哪个点在服务端的Rectangle里，返回在的点的feature（route\_grpc.pb.go里RouteGuideClient的ListFeature）

runThird() // 给服务端上传点的值,返回所有上传点的连接距离，并返回获取的时间（route\_grpc.pb.go里RouteGuideClient的RecordRoute）

runForth() // 双向流，既向服务端问询，同时监听服务端信息（route\_grpc.pb.go里RouteGuideClient的Recommend）

main() 开启dial

readIntFromCommandLine() 读取流

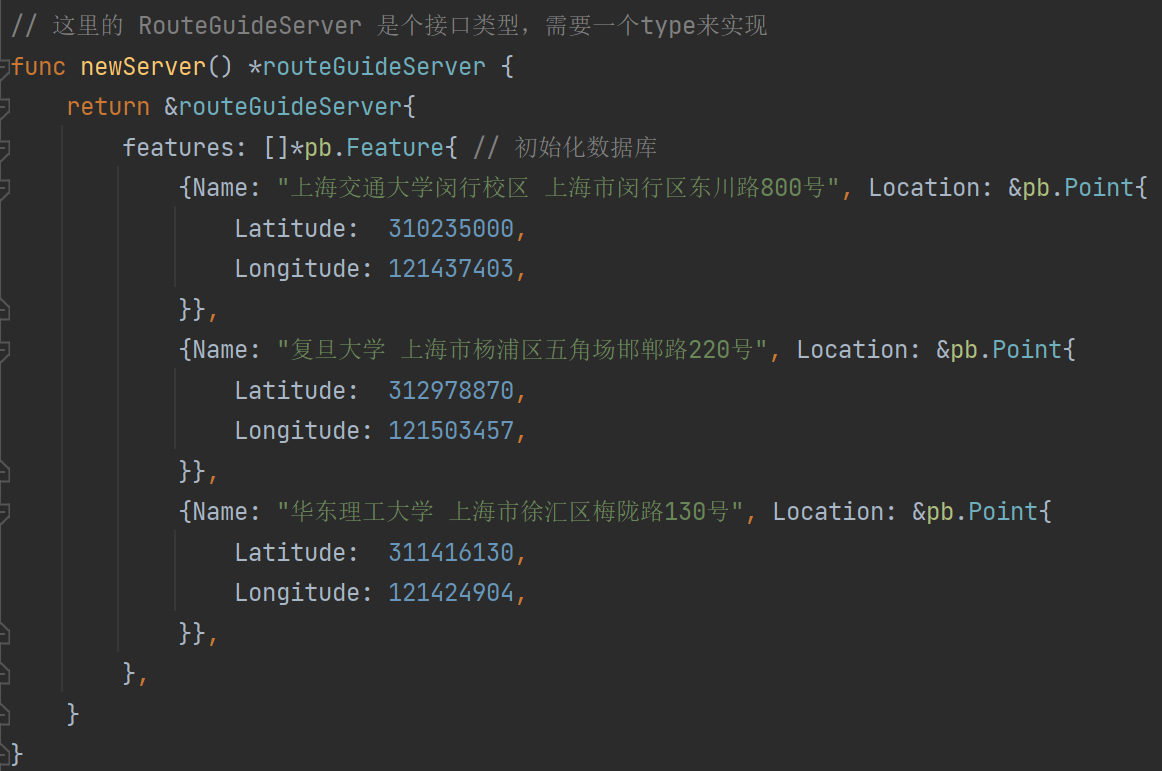
**Server.go:实现上述client需要调用的方法（对应四种生命周期）；**

GetFeature()**-----**Unary

ListFeature()-----Client-side streaming

RecordRoute()-----Server-side streaming

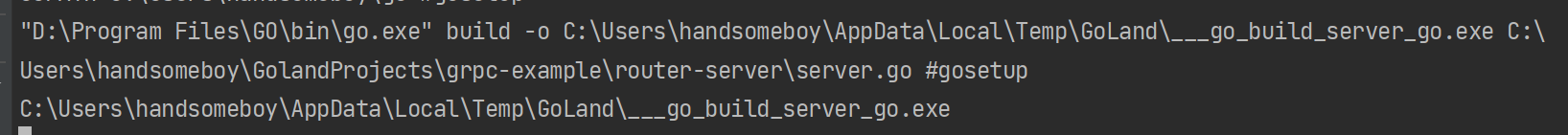
Recommend()-----Bidirectional streaming



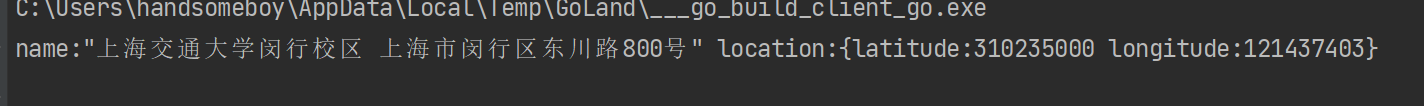
**程序运行过程截图：**

1. 使用GetFeature：

开启服务：



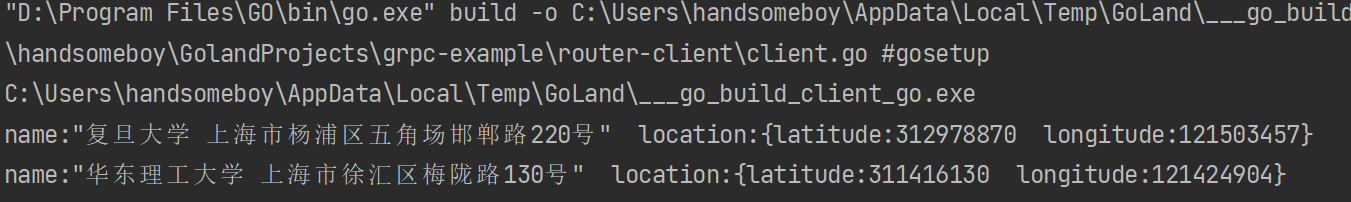
启动client：



返回这个featuer的原因是client的runFirest传输的这个point对应着server上的数据；

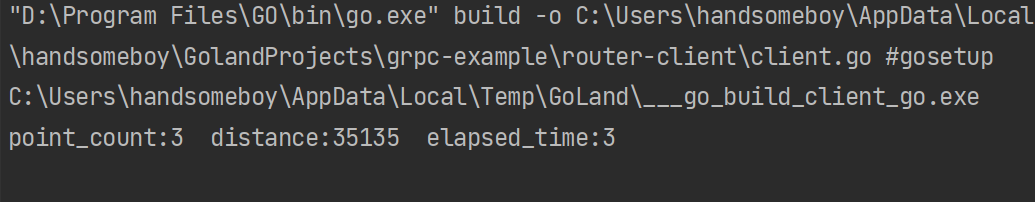


1. 使用ListFeature；

Client：

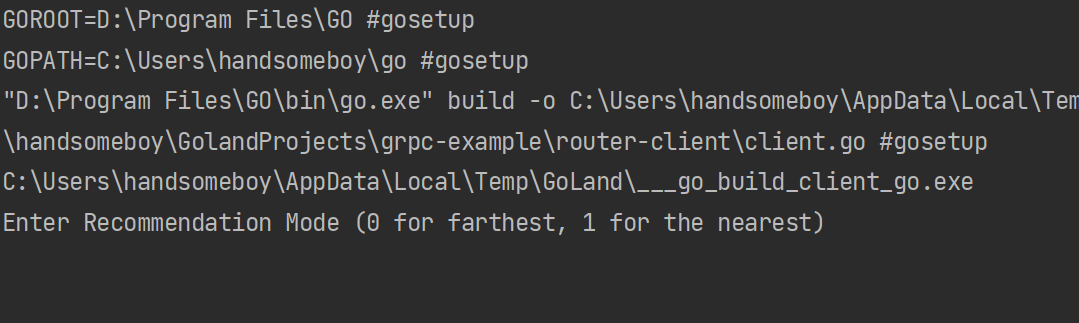
根据runSecond里的使用这个listFeatures，传输list point，返回list point的fature

1. 使用RecordRoute；



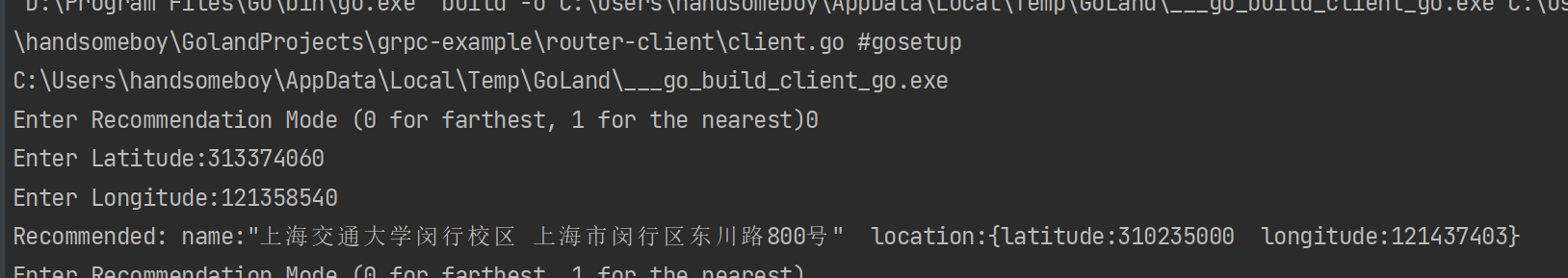
根据runThrid里的使用这个点，这边测试只使用了三个point。理论上可以传输很多个point，然后计算这些点的point\_count,distance,elapsed\_time. client上传的的是流。

1. 使用Recommend;

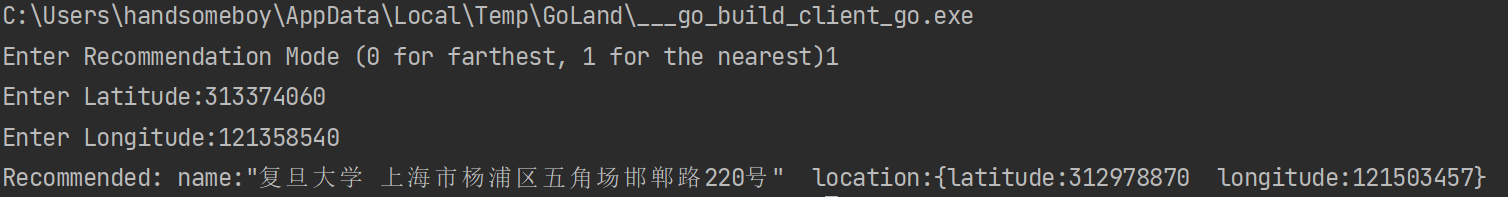


这里就是输入数字，想要那种模式：模式0就是寻找输入点最远的feature然后返回。模式1是寻找输入点最近的feature返回。

先输入0：（输入了坐标后，在这个features里离输入点最远的feature如下图所示）



先输入1：（输入了坐标后，在这个features里离输入点最近的feature如下图所示）



因为是个双向流，所以可以一直查询；

