MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

B06_a

活用IBinder接口于短(近)程通信(a)

By 高煥堂

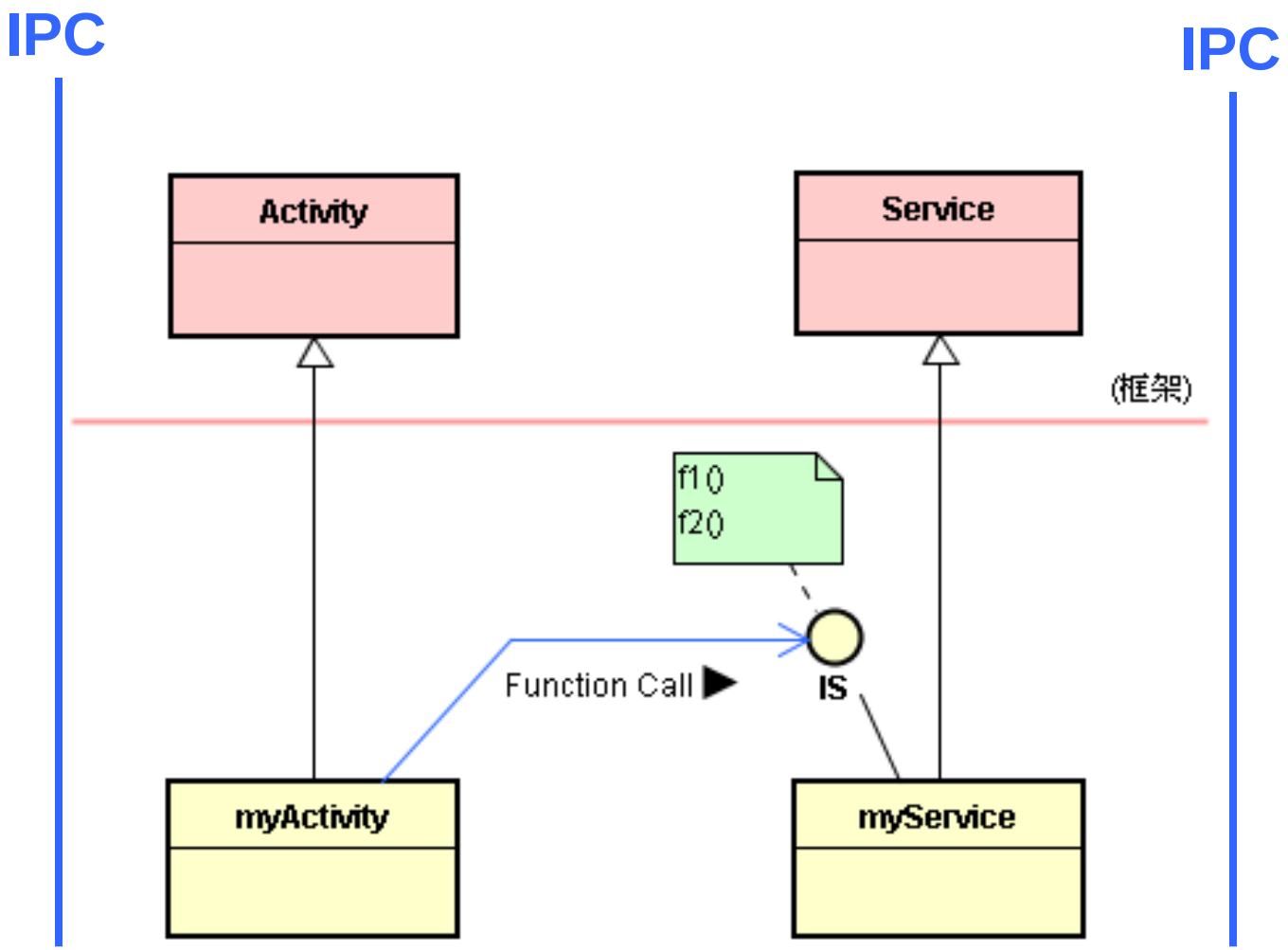
内容

1. 在同一进程里，活用IBinder接口
2. 目的、议题与方法
3. 留意线程的角色

1、在同一进程里，活用IBinder接口

前言

下图：myActivity与myService两类别是在同一个进程里执行



议题

1. myActivity对象是谁创建的呢?
2. myService对象是谁创建的呢?

3. 当myService类里有个f1()函数，如何去调用它呢?

4. 必须先取得myService对象的指针，才能调用f1()函数去存取对象的属性(Attribute)值。

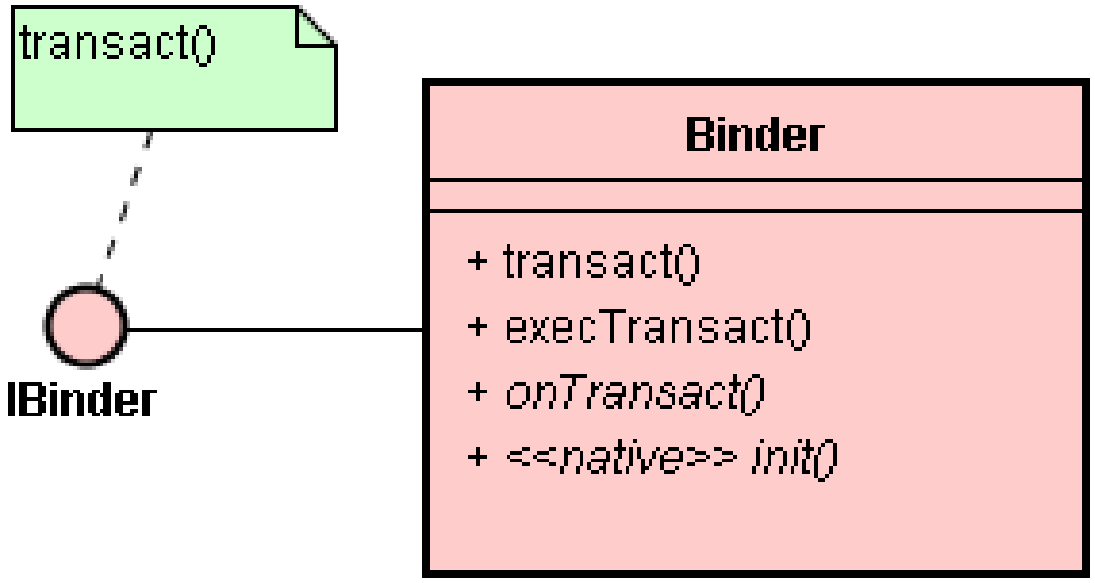
5. 那么，该如何才能取得myService对象的指 针呢?

6. 想一想，可以透过myService类的静态(static)属性或函数来取得myService对象的指针吗?

7. 可以透过IBinder接口来取得myService对象的指针吗?

- IBinder接口的重要目的是支持跨进程的远程调用。然而，它也应用于同一进程里的近程调用。
- 例如，当Activity远程调用Service时，我们常用bindService()函数去绑定Service，取得对方的IBinder接口。
- 在近程(同一进程内)调用时也可以使用bindService()函数去绑定Service，并取得对方的IBinder接口。

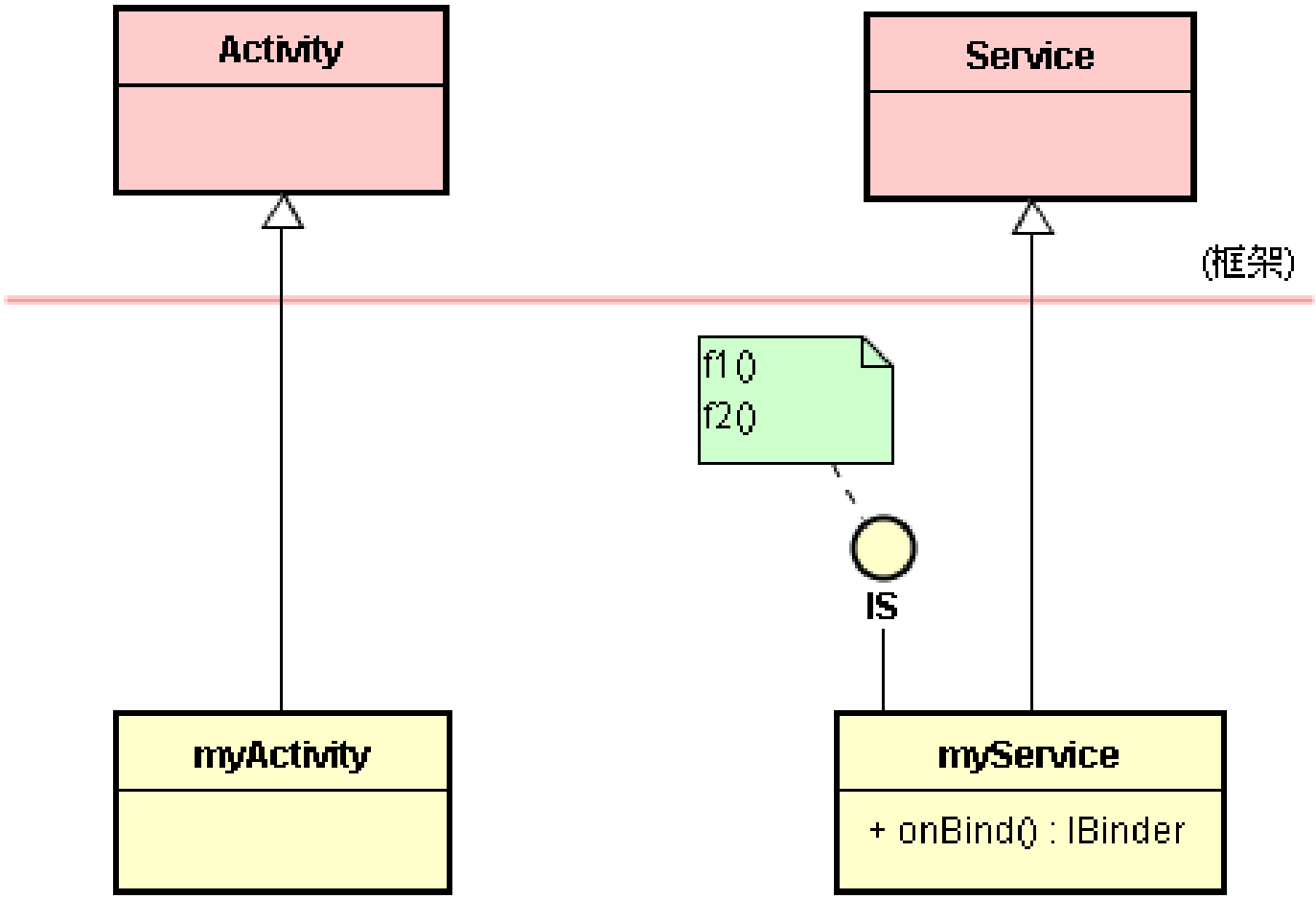
- IBinder接口的典型实现类是Binder基类，其定义于Binder.java档案里。



- 近程通信(同一进程里)如何使用IBinder接口呢?
- 举例说明之。
- 例如，myActivity和myService两者都执行于同一个进程(process)里，而且myActivity提供一个IS接口，其定义如下：

```
interface IS {
    void f1();
    void f2();
}
```

现在，myActivity想要透过此IS接口来调用mySevice的函数；如下图：


Activity
Service
(框架)
f1()
f2()
IS
myActivity
myService
+ onBind() : IBinder

2、目的、议题与方法

目的、议题与方法

- **目的**：myActivity想去直接(近程)调用myService类的函数， 例如IS接口里的f1()函数
- **议题**：如何取的myService对象的IS接口呢?
- **方法**：先取得myService对象的IBinder接口

步驟是：

Step-1. myActivity透过bindService()函数来绑定(Bind)此myService。

Step-2. myService回传myBinder类的IBinder接口给myActivity。

Step-3. myActivity将IBinder接口转换为myBinder类的接口

Step-4. myActivity调用myBinder类的getService()函数，取得myService的IS接口。

Step-5. 于是，myActivity就能调用IS接口(由myService类实现)的函数了。

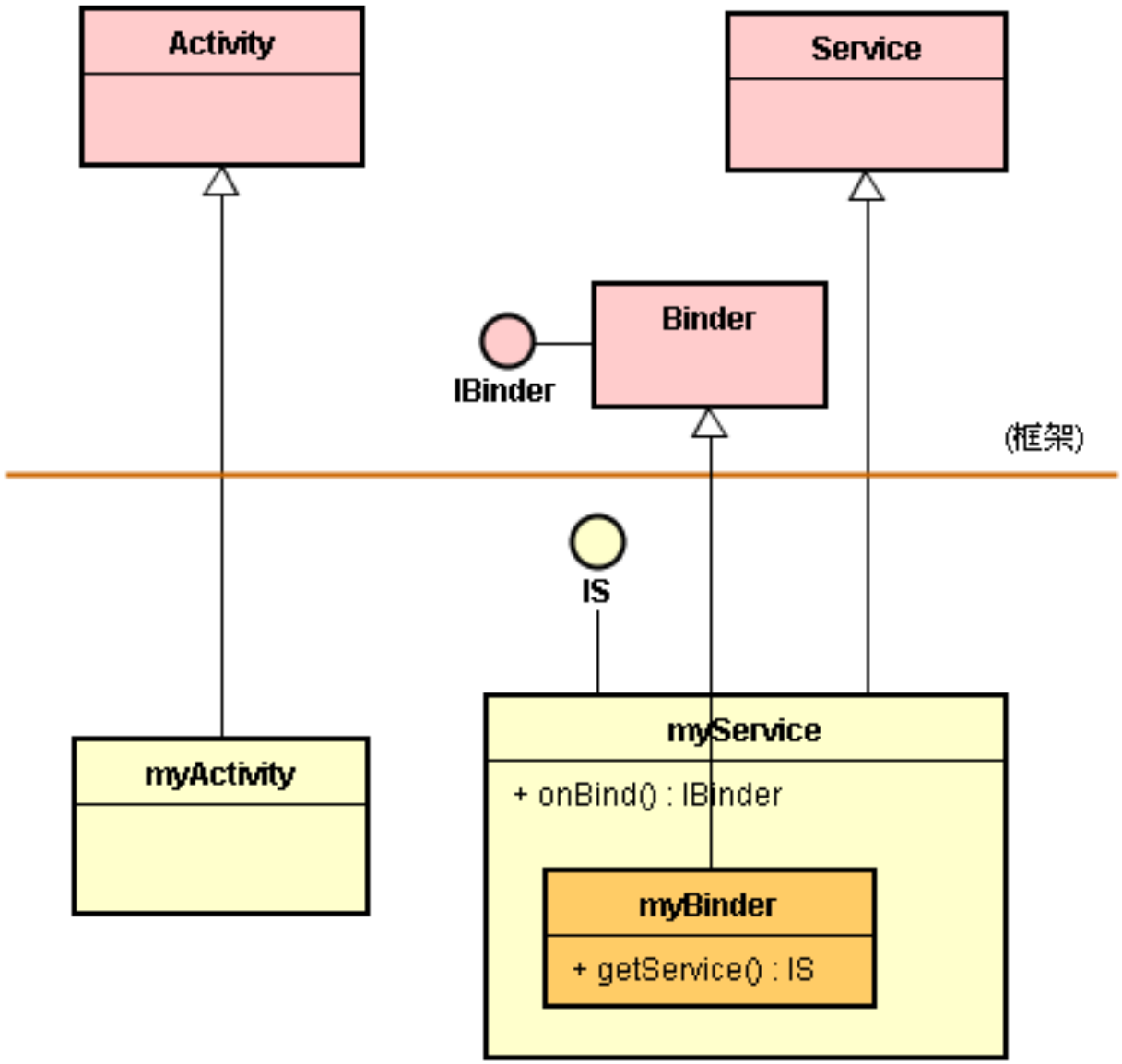
在Android 说明文件里，说明道：

- “If your service is private to your own application and runs in the same process as the client (which is common), you should create your interface by extending the **Binder** class and returning an instance of it from **onBind()**.

- The client receives the **Binder** and can use it to directly access public methods available in either the **Binder** implementation or even the **Service**.”

- 依据上述文件的说明：

  “… you should create your interface by extending the Binder class and returning an instance of it from **onBind()**.”


Activity
Service
Binder
IBinder
(框架)
IS
myActivity
myService
+ onBind() : IBinder
myBinder
+ getService() : IS

- 依据这个设计图，就来撰写myService类别如下：

```
// myService.java
// ……...
public class myService extends Service implements IS {
     private final IBinder mBinder = new myBinder();
      //……
     @Override
      public IBinder onBind(Intent intent) {
              return mBinder;
      }
      //……
      public class myBinder extends Binder {
            IS getService() {
                  return myService.this;
      }
      public void f1(){  //…….   }
      Public void f2() {  //……  }
}
```

```
// myActivity.java
//……..
public class myActivity extends Activity {
  IS isv;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
        //……..
        Intent intent = new Intent(this, myService.class);
         bindService(intent, mConnection,
                          Context.BIND_AUTO_CREATE);
   }
```

```
private ServiceConnection mConnection = new ServiceConnection() {
      @Override
        public void onServiceConnected(ComponentName className,
                                       IBinder ibinder) {
        myBinder ib = (myBinder)ibinder;
         isv = ib.getService();
     }}
    public void onClick(View v) {
    // ..... 例如：isv.f1()
    }
}
```

第1步

- 当Android框架启动myService时，就立即执行：

```
private final IBinder mBinder = new myBinder();
```

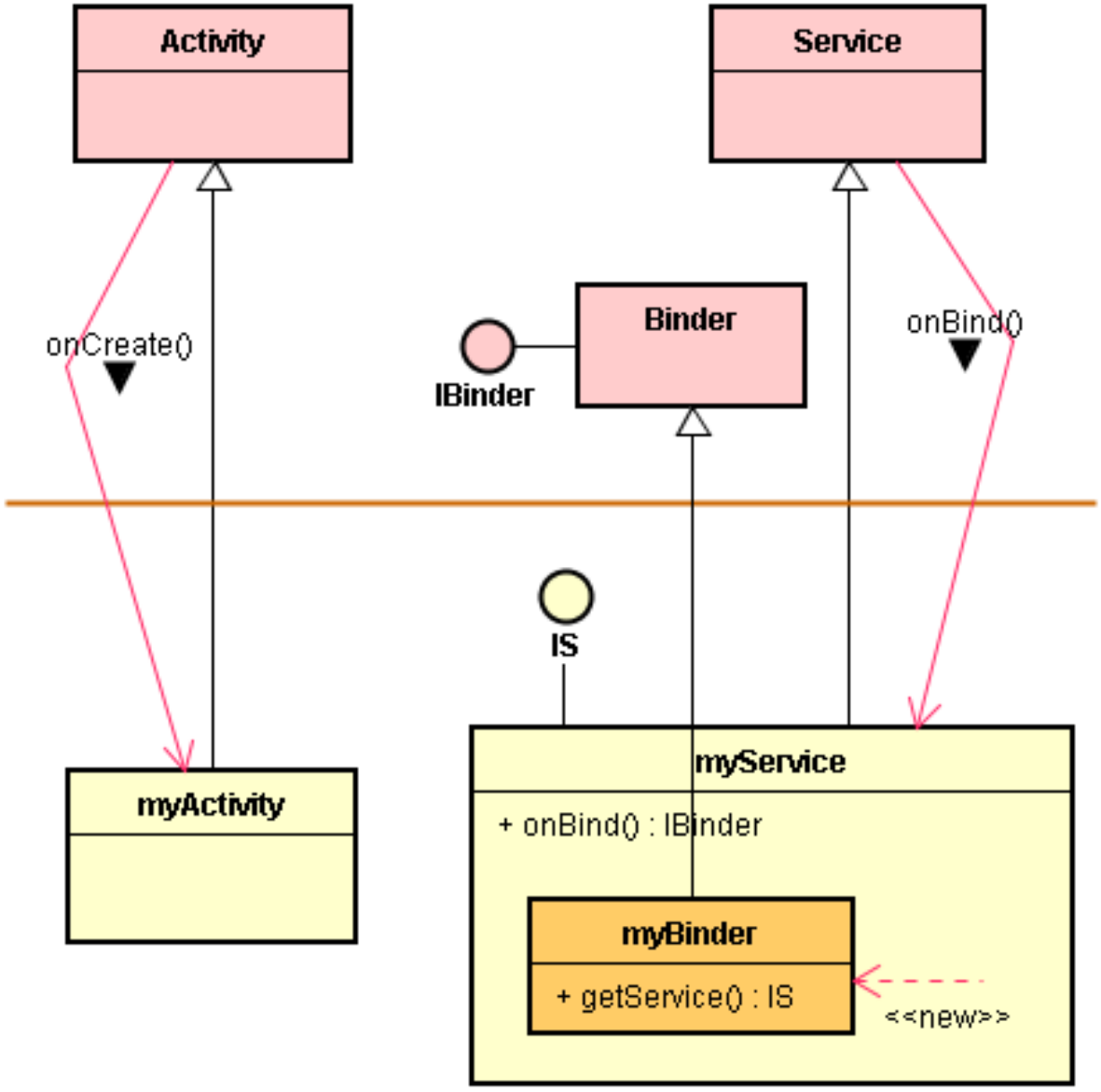
這诞生了myBinder对象。

第2步

- 随后，当myActivity调用bindService()时，框架会反向调用到myService的onBind()函数：

```
public IBinder onBind(Intent intent)
    {   return mBinder;   }
```

其将 myBinder的IBinder接口回传给框架，并由框架调用onServiceConnected()函数，将此接口回传给myActivity。


Activity
Service
onCreate()
Binder
IBinder
onBind()
IS
myActivity
myService
+ onBind() : IBinder
myBinder
+ getService() : IS
<<new>>

第3步

- 由于myActivity与myService在同一个进程里执行，myActivity所获得的就是myBinder的真正接口(不是它的Proxy的)；于是，执行：

```
myBinder ib = (myBinder) ibinder;
```

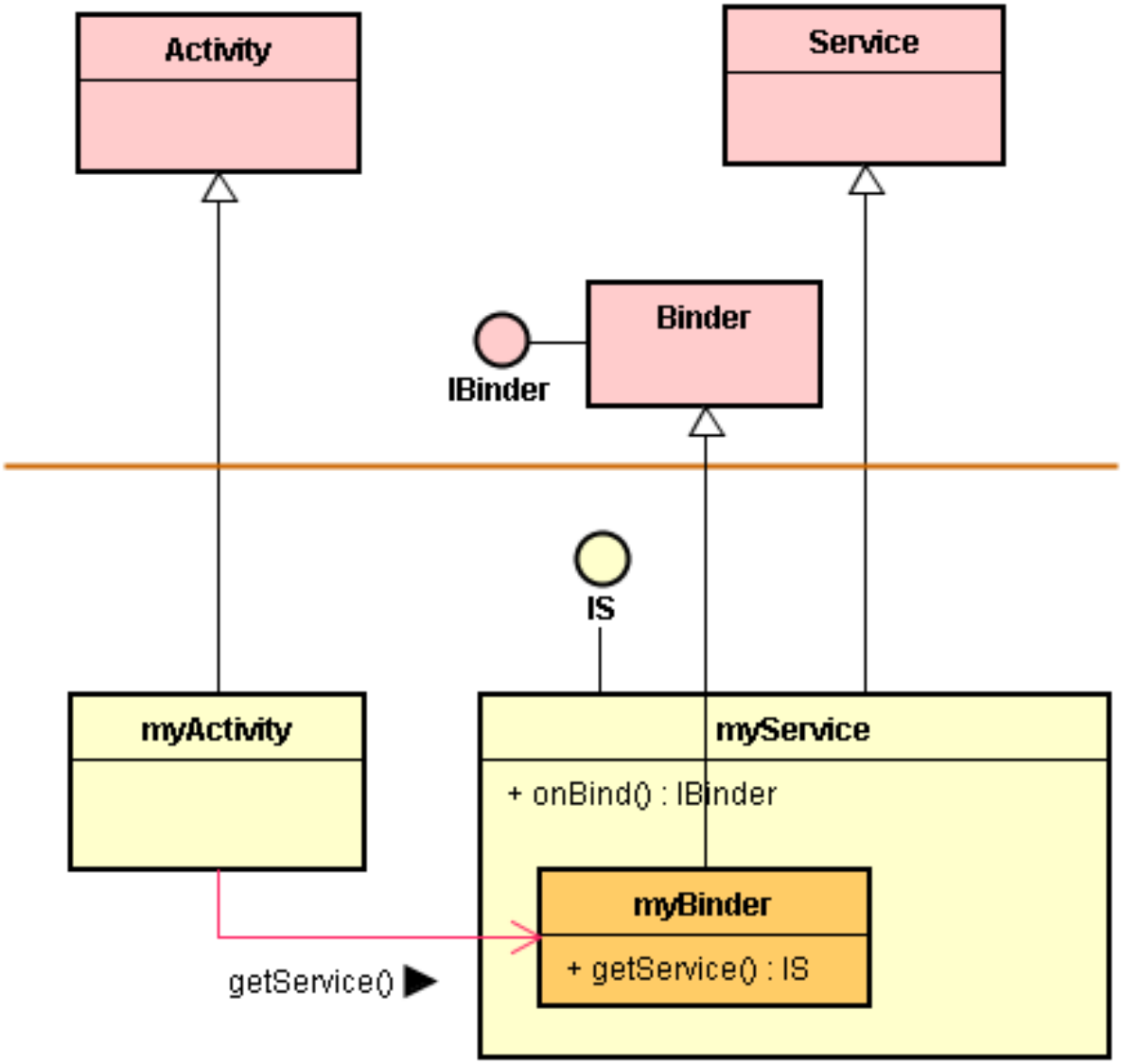
- 就从接获的IBinder接口转型(casting)为myBinder本身接口了。

第4步

- 接着，执行：

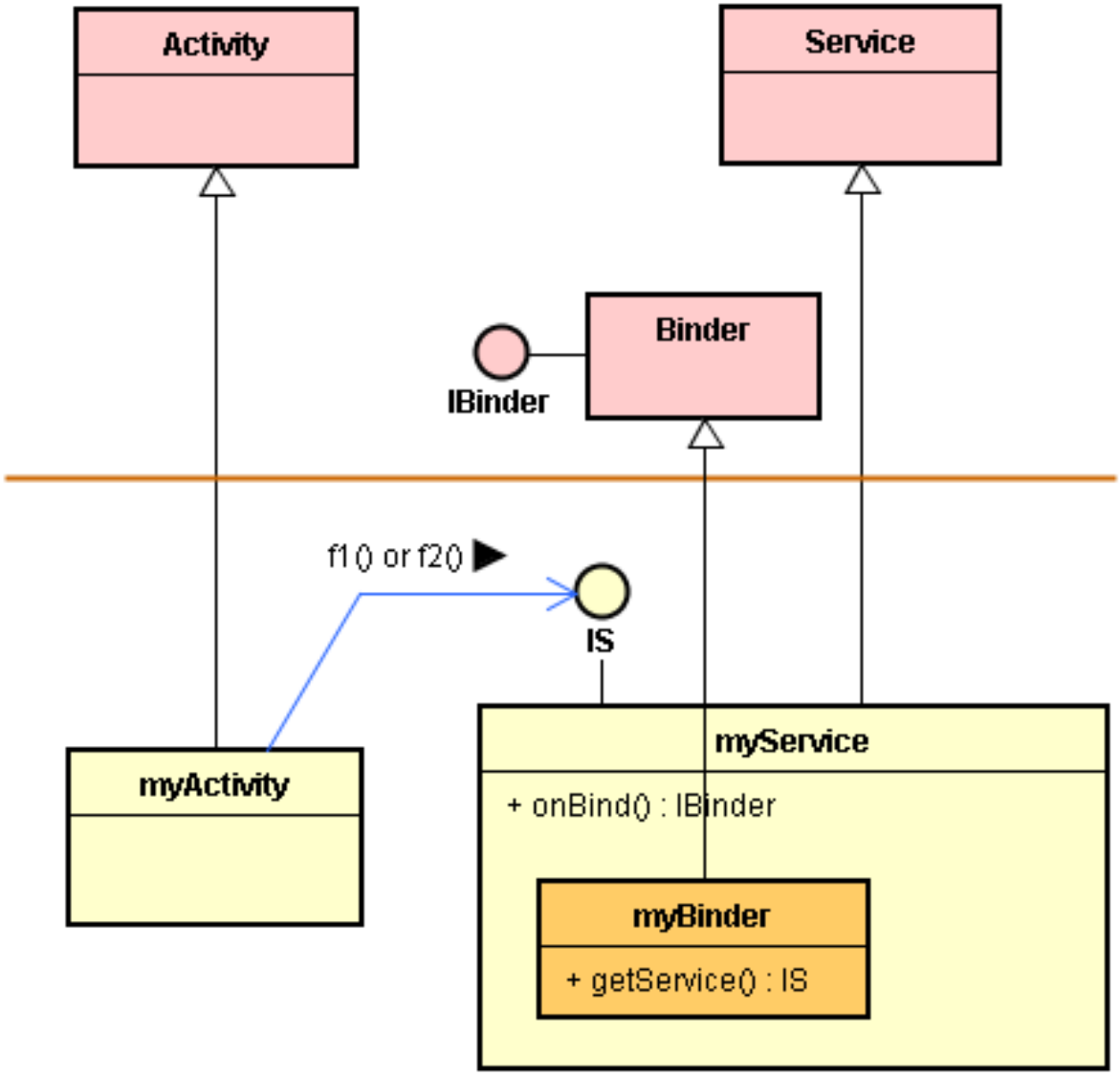
```
isv = ib.getService();
```

- 这透过myBinder本身接口来调用getService()函数，取得了myService的IS接口。


Activity
Service
Binder
IBinder
IS
myActivity
myService
+ onBind() : IBinder
myBinder
+ getService() : IS
getService()

第5步

- 最后，myActivity就能透过IS接口来调用myService的f1()或f2()函数了。


Activity
Service
Binder
IBinder
f1() or f2()
IS
myActivity
myService
+ onBind() : IBinder
myBinder
+ getService() : IS

~ Continued ~