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

B07_b

# Messenger框架与IMessenger接口(b)

By 高煥堂

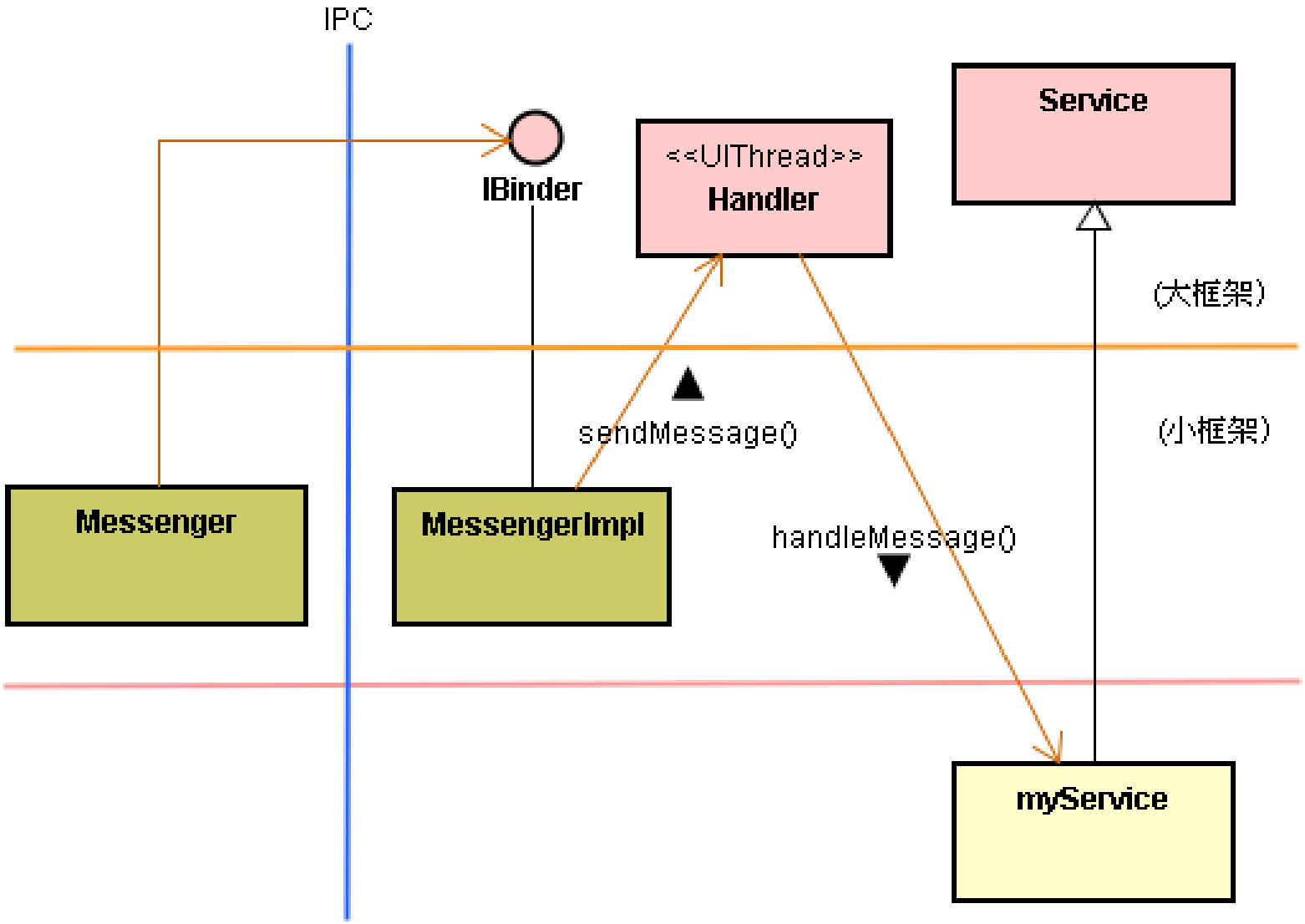
# 2、Android的Messenger框架

# 复习：线程、信息和IBinder接口

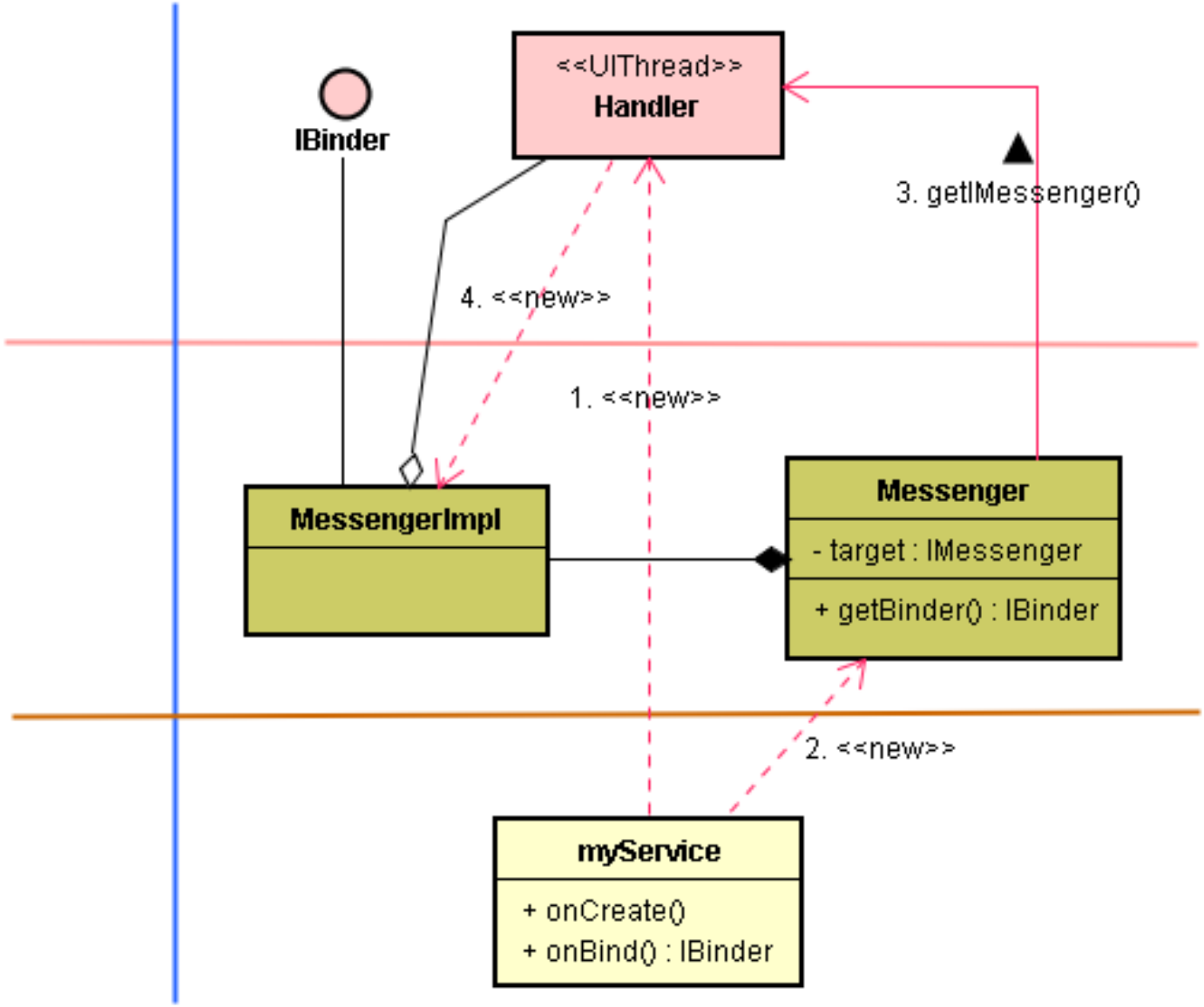
- 在Android框架里，有个IBinder接口来担任跨进程的通讯。
- 在Android框架里，也有一个Message类，两个线程之间能互传Message对象。

- 于是，就能设计一个**Messenger类**来包装IBinder接口机制，让其能跨进程地将Message对象传递到另一个进程里，给其主线程(又称UI线程)。

- 其中，由于Message类实作(Implement)了Parcelable接口，所以**Messenger类**可以透过IBinder接口而将Message对象传送到另一个进程里的**MessengerImpl类**。
- 然后，Messenger透过**Handler**而将Message对象丢入UI线程的MQ里，让UI线程来处理之。

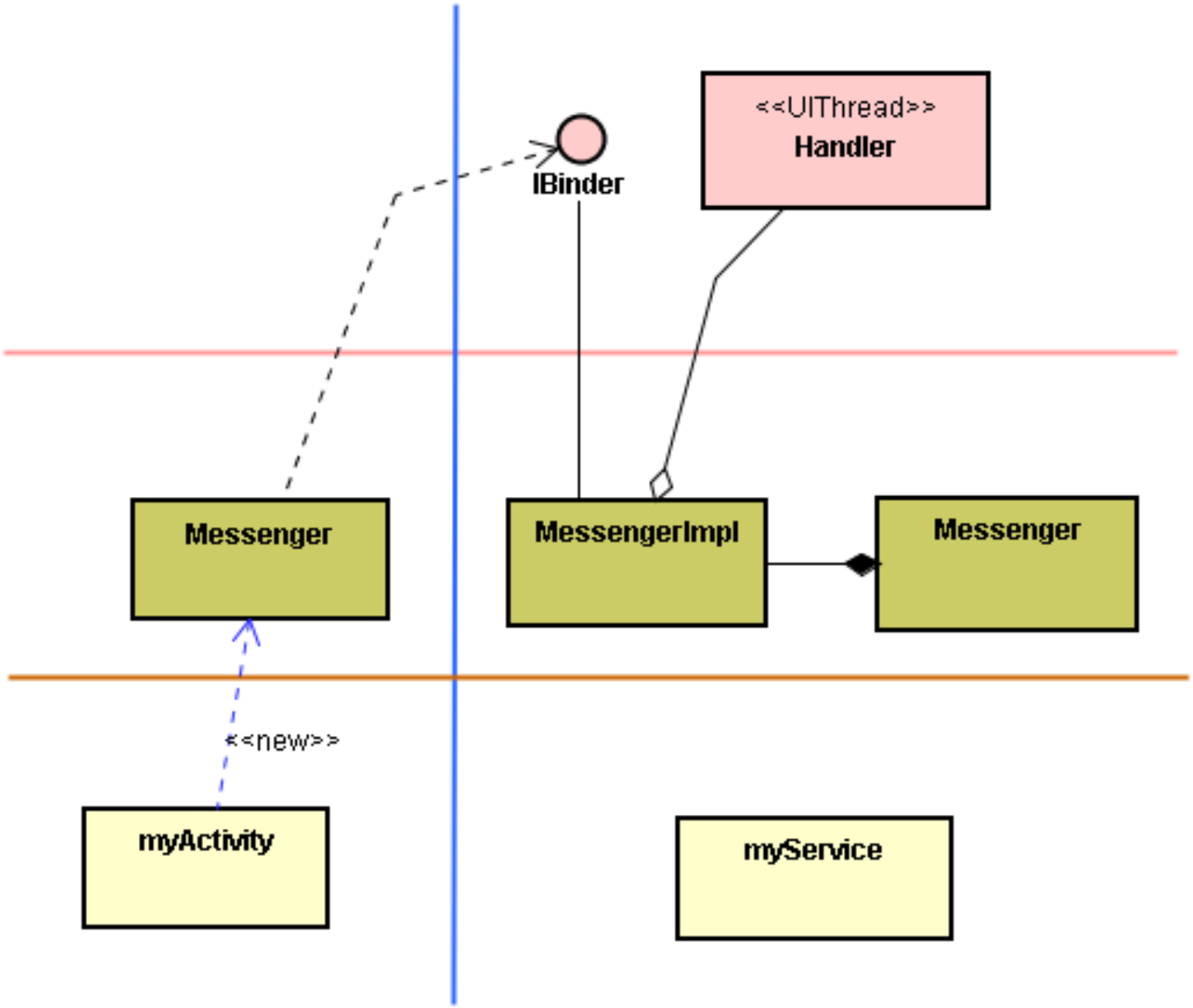

IPC
IBinder
<<UIThread>>
Handler
Service
(大框架)
(小框架)
sendMessage()
handleMessage()
Messenger
MessengerImpl
myService

- 在传送Message对象之前，必须先建立MessengerImpl、Handler和myService三者之间的关系。如下图：


<<UIThread>>
Handler
IBinder
3. getIMessenger()
4. <<new>>
1. <<new>>
MessengerImpl
Messenger
- target : IMessenger
+ getBinder() : IBinder
2. <<new>>
myService
+ onCreate()
+ onBind() : IBinder

- 首先myService诞生一个Handler对象，并诞生一个Messenger对象，并让Messenger指向该Handler对象。
- 于是，Messenger对象调用Handler的getIMessenger()函数去诞生一个MessengerImpl对象，并让Messenger对象指向MessengerImpl对象。
- 此时，MessengerImpl对象也指向Handler对象。

- 建构完毕后，在另一个进程里的myActivity就能透过Messenger类而将Message对象传递给MessengerImpl对象。
- 然后，MessengerImpl继续将Message对象放入主线程(main thread)的MQ里，如下图所示：


<<UIThread>>
Handler
IBinder
Messenger
MessengerImpl
Messenger
<<new>>
myActivity
myService

# 步骤是：

- myActivity调用bindService()去绑定myService，取得IBinder接口。
- 以Messenger类包装IBinder接口。
- myActivity透过Messenger类接口将Message信息传给远方的MessengerImpl类。

- MessengerImpl类将信息丢入对方主线程的MQ里。
- 主线程从MQ里取得信息，并调用myService的函数来处理信息

# 程序代码

```
// myService.java
// ……..
public class myService extends Service {
   class myHandler extends Handler {
        @Override public void handleMessage(Message msg) {
            //……..
            Toast.makeText(getApplicationContext(),msg.obj.toString(),
                  Toast.LENGTH_SHORT).show();
            //……..
            }
        }
    }
   final Messenger mMessenger = new Messenger(new myHandler());
```

```
    @Override
     public IBinder onBind(Intent intent) {
                return mMessenger.getBinder();
     }
}
```

```
// myActivity.java
// ……
public class myActivity extends Activity {
    Messenger mMessenger = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);  bindService(new Intent(this,
                MessengerService.class), mConnection,
                Context.BIND_AUTO_CREATE);  }
    private ServiceConnection mConnection =
        new ServiceConnection() {
            public void onServiceConnected(ComponentName
                        className, IBinder ibinder)
            {
                mMessenger = new Messenger(ibinder);
            }
        }
```

```
public void onClick() {
        Message msg = Message.obtain(null, 0, “Hello”);
        mMessenger.send(msg);
    }
 }
```

- 一开始，框架会诞生myService对象，此时也执行指令：

```
final Messenger mMessenger =
    new Messenger(new myHandler());
```

- 就诞生一个myHandler对象，并且诞生一个Messenger对象，并把myHandler对象指针存入Messenger对象里。

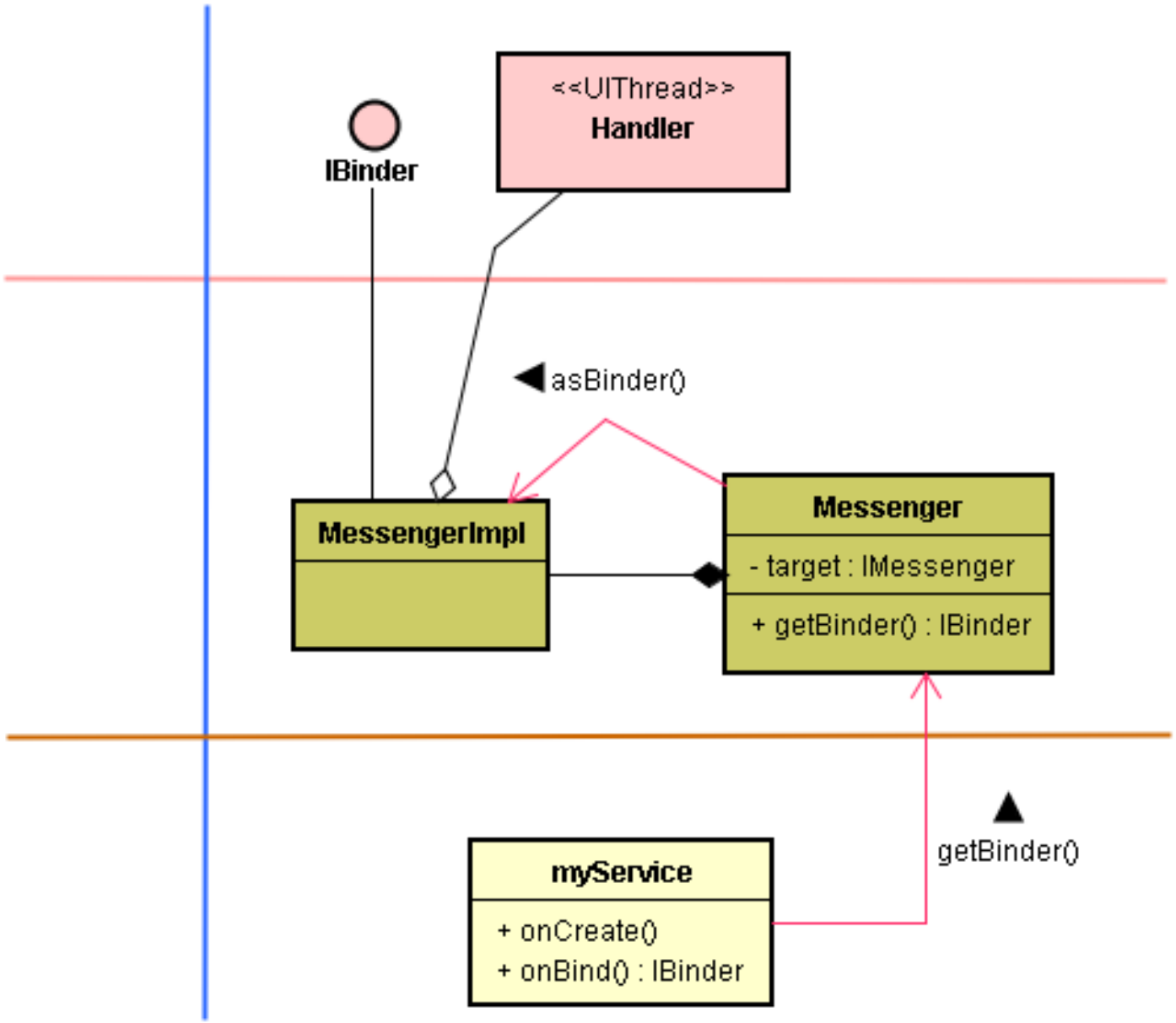
- 一旦myActivity执行到指令：

```
bindService(new Intent(this, MessengerService.class),
    mConnection, Context.BIND_AUTO_CREATE) ;
```

- 框架会调用myService的onBind()函数，其内容为：

```
public IBinder onBind(Intent intent) {
    return mMessenger.getBinder();
}
```

- 此时，调用Messenger的getBinder()函数来取的MessengerImpl的IBinder接口，并回传给Android框架。如下图：


<<UIThread>>
Handler
IBinder
asBinder()
MessengerImpl
Messenger
- target : IMessenger
+ getBinder() : IBinder
getBinder()
myService
+ onCreate()
+ onBind() : IBinder

- 接着，框架就调用myActivity的onServiceConnected()函数：

```
public void onServiceConnected(ComponentName
    className, IBinder ibinder) {
     mMessenger = new Messenger(ibinder);
    }
```

- 此时，就让Messenger对象指向IBinder接口了。

- 一旦myActivity执行到指令：

```
public void onClick() {
    Message msg = Message.obtain(null, “hello”, 0, 0);
    mMessenger.send(msg);
}
```

- 就诞生一个Message对象，然后调用Messenger的send()函数，此send()函数则调用IBinder接口的transact()函数，将Message对象传递给MessengerImpl，再透过myHandler将Message对象放入主线程的MQ里。

# 再谈线程的角色

- 在Android文件里，写道：

  “… if you want to perform IPC, but do *not* need to handle multithreading, implement your interface using a Messenger.”

- 但是，有许多人看不懂其涵意。

- 其实，它的涵意很简单。如果你并不考虑让多个线程(thread)同时来执行你的Service，你就可以透过这个机制，将多个Client端(如myActivity1, myActivity2等)送来的Message对象存入单一线程的MessageQueue里，由该线程依序逐一地处理各Client传来的Message对象。

- 虽然多个并行的Client端线程在调用IBinder接口时，会触发多个Binder驱动线程(Binder Thread)而进入MessengerImpl，然而它们则依序将Message丢入同一个(即主线程的)MessageQueue里。因此，对于Service而言，还是单线程的情境，你在撰写myService程序代码时，不必担心多线程之间的数据冲突问题。

~ Continued ~