UML 建模與 Android 應用程序開發

By 高煥堂

前言

就一個完整的軟件系統而言，程序碼只是系統(本體)的一個觀點(View)而已，而模型(Model)也是系統(本體)的一個觀點。當 Android 應用開發者來說，若既能從程序碼看應用系統，又能從模型看它，就相當於人們都有兩隻眼睛來看前方的一切事物。一旦發現兩者有不一致的情形，就表示兩者可能有所失真(即遠離本體)了。這樣的訊息，可讓開發者提前知道未來開發路子上，可能發生的錯誤，以便防患未然。同樣地，在模型大觀點裡，也含有許多小觀點，例如：

- 架構觀點：一般採用 UML 類別圖(Class Diagram)
- 使用觀點：一般採用 UML 用例圖(Use Case Diagram)
- 順序觀點：一般採用 UML 順序圖(Sequence Diagram)
- 狀態觀點：一般採用 UML 狀態圖(Statechart Diagram)

在本文裡，將說明如何就上述的 4 個小觀點，來構成模型大觀點，然後再與程序碼觀點匯合，成爲一個穩定可靠、簡潔高雅的 Android 應用系統。然而，特別留意的是：模型觀點與程序碼觀點兩者不一定要有明確的先後順序關係。兩者之間，到底何者先，而何者後，並非重點。因爲最好的狀態是：在腦海裡先兩者並存，先領悟構思，然後才畫出 UML 模型圖，也寫出程序碼，但都不一定是完美的。隨著兩個觀點的對比，發現不一致現象，就像兩隻眼睛發現前方物體的呈像不一致時，兩者自然而然會逐漸修正(Iterative & Incremental)，止於至善。

本文範例

本文舉一個簡單範例：一個 Activity 的子類別，以及一個遠程的(Remote)的 Service 子類別。兩者透過 Android 的 IPC 機制相互溝通。

多種 UML 類別圖呈現各種架構觀點

所有的模型圖都是人們對某項事物本體認知的心智觀點，隨著觀點和抽象的角度之不同而改變其所呈現之面貌。例如，當我們覺得 Android 框架裡的基類(即抽象類)是最重要的，只要呈現它即可，此時類別圖就呈現如下：

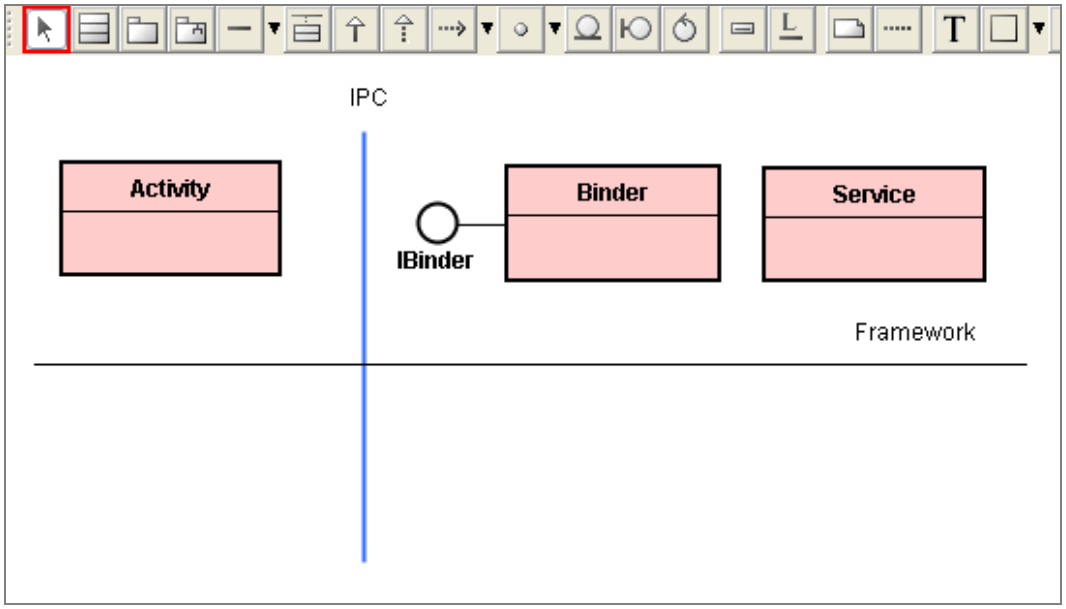


圖 1　獨尊 Android 框架的類別圖

如果覺得應用子類別也是架構裡的重要元素，需要與框架裡的基類別一起呈現出來，則此時類別圖就呈現如下：

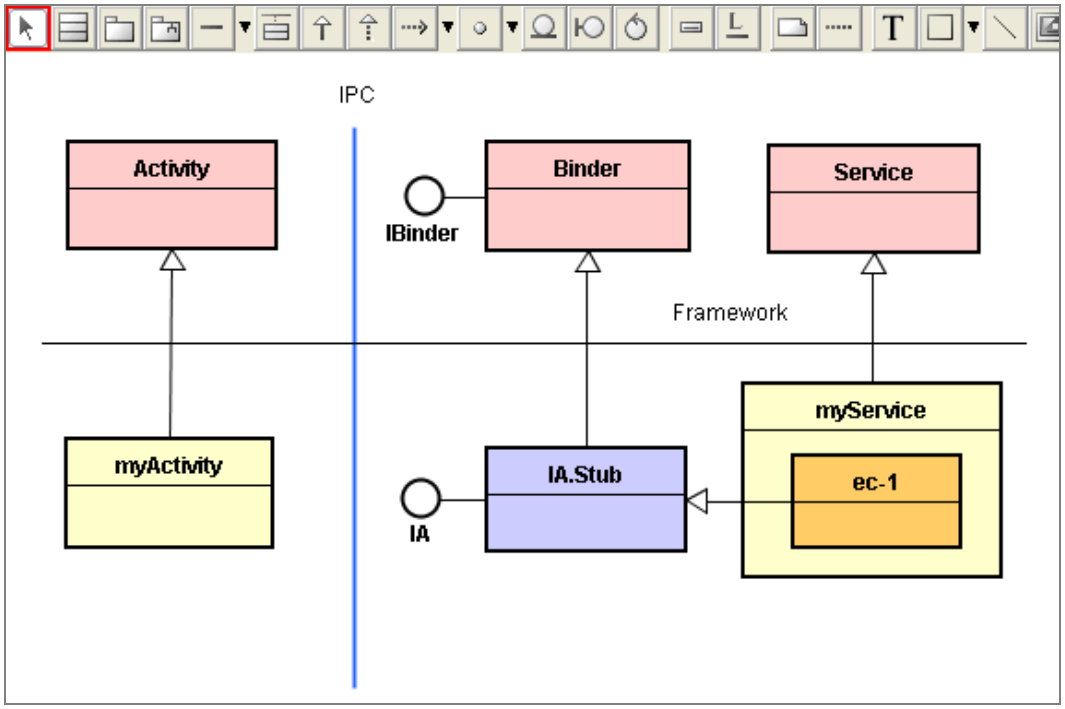


圖 2　兼具框架與應用的類別圖

當然也有許多人習慣於獨尊應用子類別，而認爲不需要呈現幕後的框架基類

別，則此時類別圖就呈現如下：

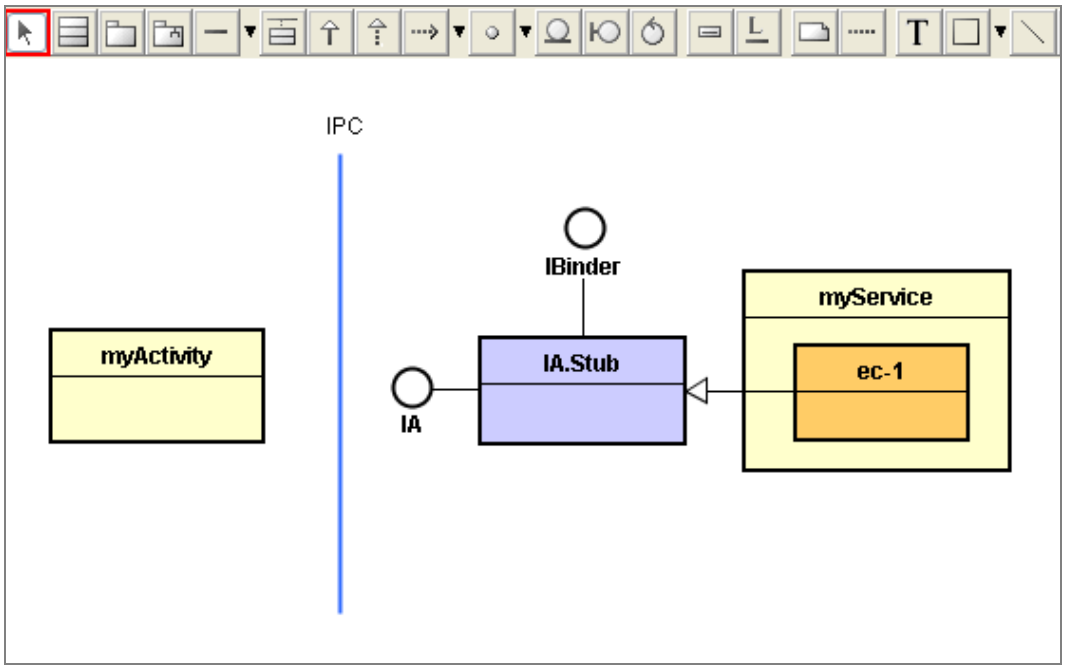


圖 3　獨尊應用子類別的類別圖

此外，還有人習慣於獨尊接口(即接口)，而對幕後實作類別視而不見，則此時類別圖就呈現如下：

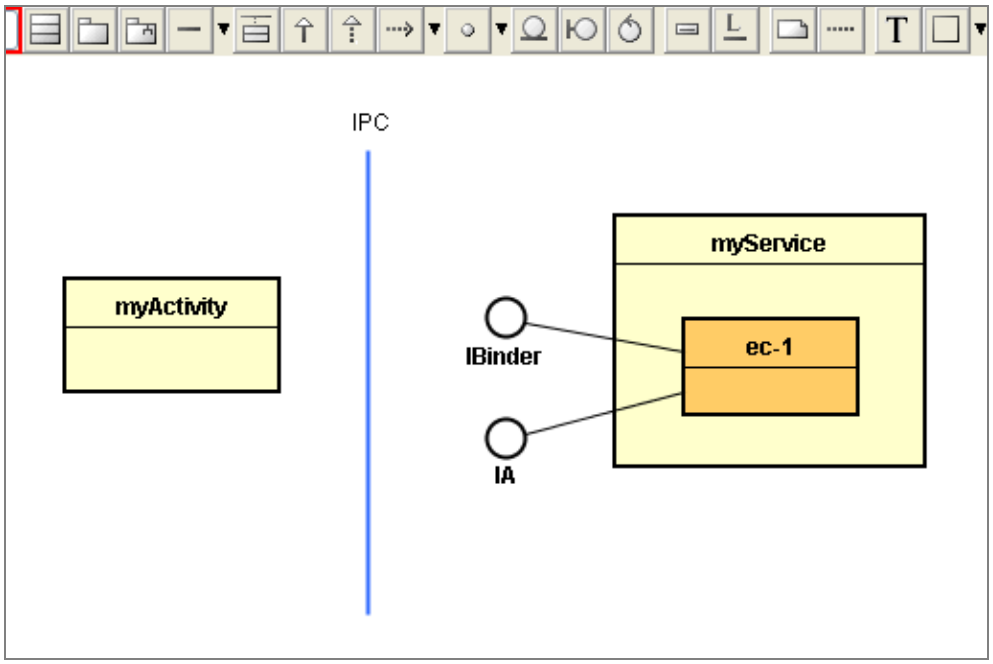


圖 4　強調接口(即接口)的類別圖

以上都只強調架構裡的元素(如類別和接口)，還有人認爲這些元素之間的互動(Interaction)與合作(Collaboration)是非常重要的，需要表現出來，此時類別圖

就呈現如下：

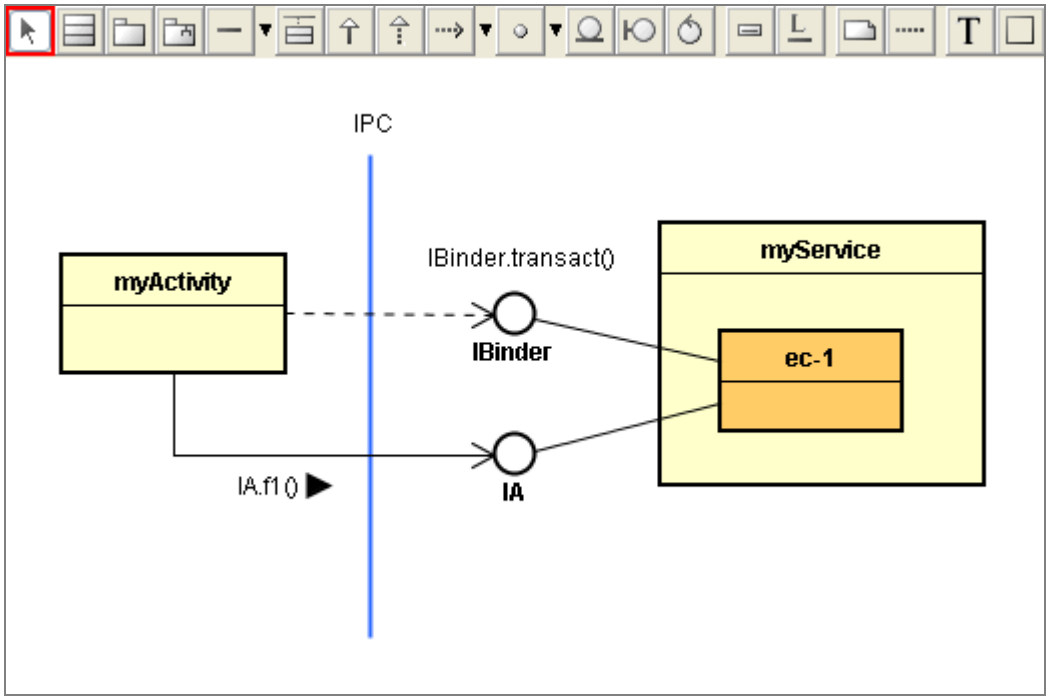


圖 5　強調互動的類別圖

UML 用例圖呈現使用觀點

類別圖是基於架構師的觀點，偏向系統的內觀(Internal View)。至於用例圖(Use Case Diagram)則是基於使用者(即用戶)的觀點，偏向系統的外觀(External View)。許多人堅持需求至上(Requirement-based)的開發者，非常重視這項 UML 圖，終究用戶是買家，就行銷的角度來看，用戶觀點當然非常重要囉。例如，針對上述範例的 UML 用例圖呈現如下：

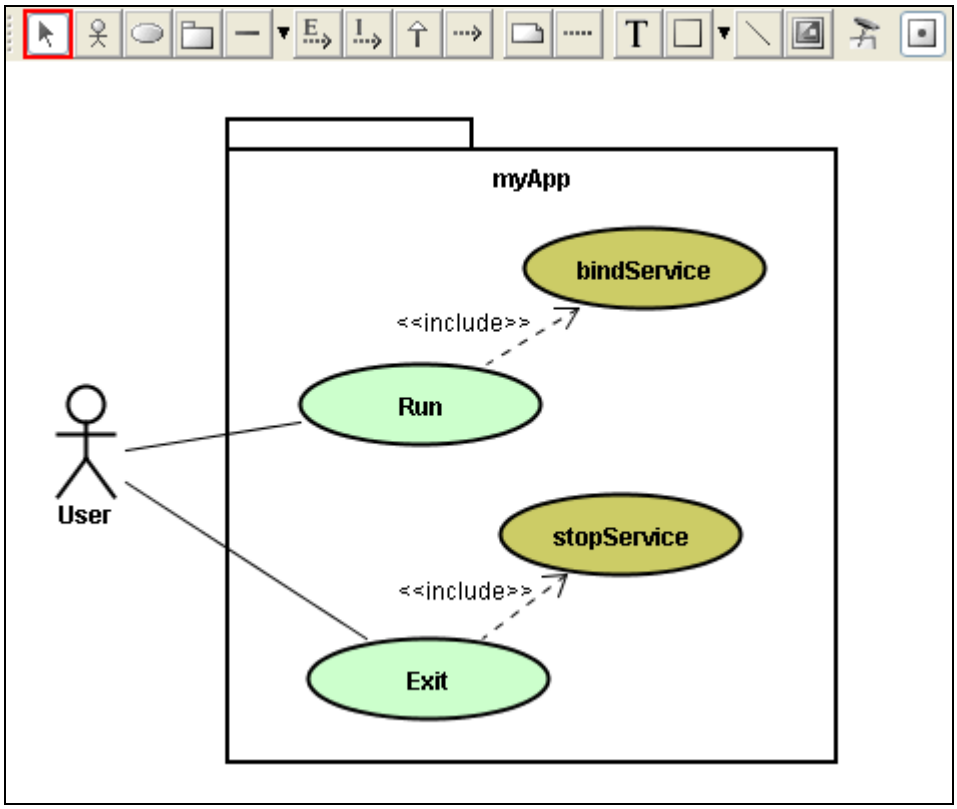


圖 6　強調用戶觀點的 UML 用例圖

UML 順序圖呈現各項活動發生順序

如何去實現上述的用例呢? 爲了實現上述的用例，系統必須執行一連串的活動(Action)。有人認爲這些用例幕後的活動執行順序是很重要的，就使用 UML 順序圖來表達之，例如，針對用例「Run」可繪製其幕後活動的執行順序，如下述的 UML 順序圖：

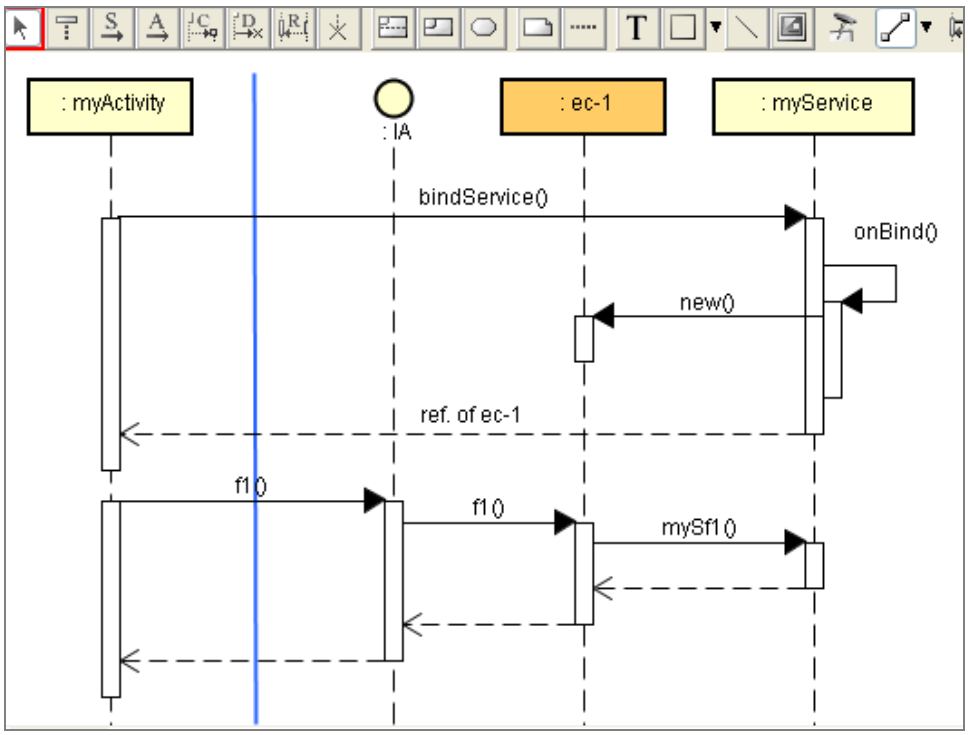


圖 7　Run 用例幕後的活動順序圖

再如，針對用例「Exit」可繪製其幕後活動的執行順序，如下述的 UML 順序圖：

Use Case: Exit

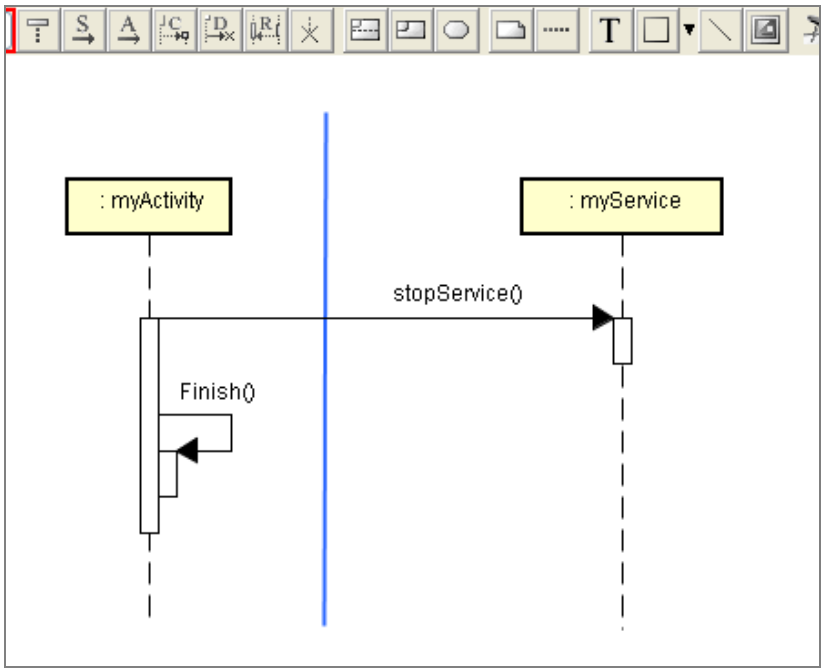
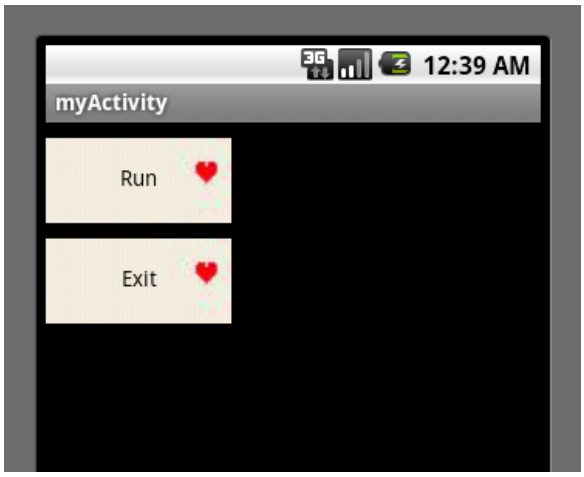


圖 8　Exit 用例幕後的活動順序圖

UML 狀態圖呈現 UI 畫面的千變萬化

由於 Android 是屬於事件驅動(Event-Driven)的平台系統，有許多人主張善用 UML 狀態圖可對眾多事件分而治之，於是在清晰的狀態之下，會執行明確的活動。例如，下述的畫面可接受來自 Android 和用戶所引發的事件。



此時，可以採用 UML 狀態圖呈現 UI 變化之觀點，如下圖：

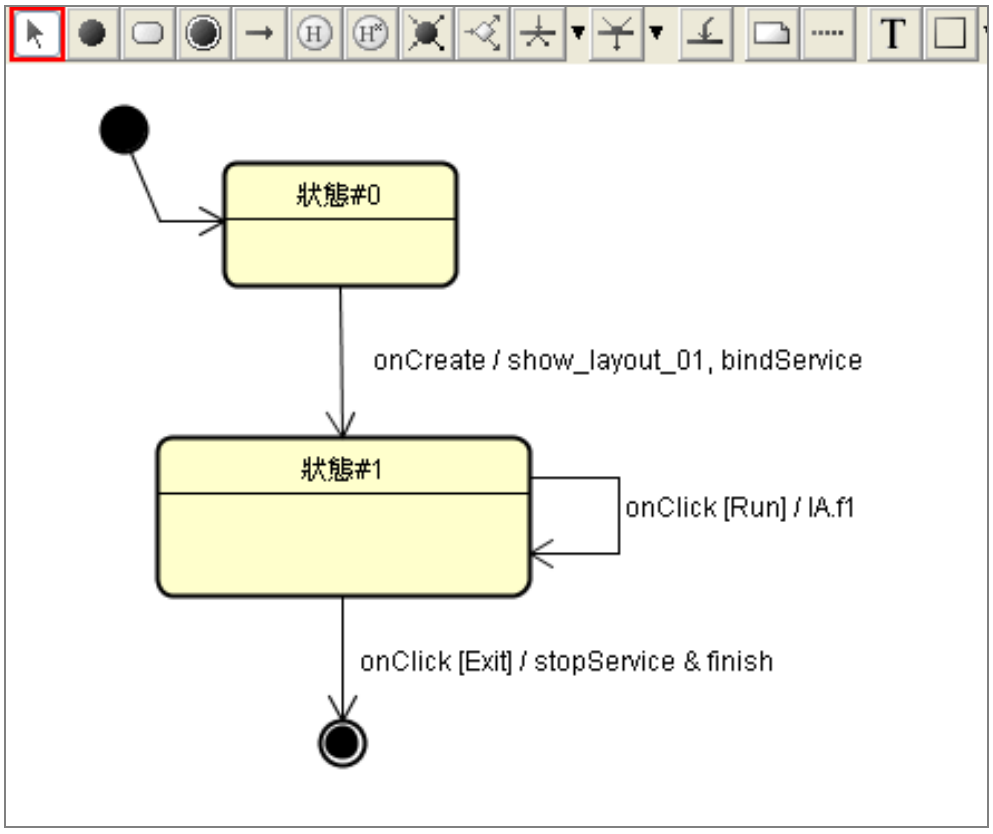


圖 9　呈現 UI 畫面狀態變化的 UML 狀態圖

以 Java 語言表達程序碼的觀點

至今天，還是有許多人維持傳統的觀點：

- 畫 UML 模型圖在先，撰寫程序碼在後。
- 程序碼是 UML 模型的實踐。
- UML 模型較爲抽象，程序碼較爲具體。

這項傳統觀點並沒有對與錯。但是，近年來，愈來愈多的人們持著新的觀點：

- UML 模型圖與 Java 程序碼是兩個同位階的觀點。
- 兩個觀點的一致性是確保系統穩定可靠、簡潔高雅的重要途徑。
- 傑出的 Android 開發者應該兼具兩個觀點。

經過兩個觀點的互相核對與逐步修正後，的確呈現出極爲完美的程序碼，如下：

Android 應用程序 Project

這包含了 1 個 IA.java 接口定義檔，及兩個應用子類定義檔：

一致化的程序碼如下所示：

// IA.java 接口

```
package com.misoo.pk01;
import android.os.Binder;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;

public interface IA {
	int f1(int x)throws RemoteException;
	public static abstract class Stub extends Binder implements IA
	{
	 @Override
		public boolean onTransact(int code, Parcel data, Parcel reply, int flags)
		throws android.os.RemoteException{
			 int x = data.readInt();
			int y = this.f1(x);
			reply.writeInt(y);
			 return true;
		}
```

```
        public abstract int f1(int x) throws RemoteException;
      public static IA asInterface(IBinder obj){
        return new Proxy(obj);
      }
       //-----------------------------------------
      private static class Proxy implements IA{
    private IBinder mRemote;

    public Proxy(IBinder ibinder){
          mRemote = ibinder;
    }

    public int f1(int x) throws RemoteException {
          // TODO Auto-generated method stub
          Parcel data = Parcel.obtain();
          data.writeInt(x);
          Parcel reply = Parcel.obtain();
          mRemote.transact(0, data, reply, 0);
          return reply.readInt();
    }
   }
  }
}
```

// myService.java

```
package com.misoo.pk01;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;

public class myService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
          return mBinder;
    }
    public int mySf1(int x){
          return x + 1000;
    }
    //-----------------------------------------
    private final IA.Stub mBinder = new IA.Stub() {
          @Override
          public int f1(int x) throws RemoteException {
                return mySf1(x);
          }
    };
}
```

// myActivity.java

```
package com.misoo.pk01;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
```

```
import android.graphics.Color;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class myActivity extends Activity implements OnClickListener {
        private final int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
        private final int FP = LinearLayout.LayoutParams.FILL_PARENT;
        private Button btn, btn2;
        private TextView tv;
        private IA ia;
        private int state_var_A = 0;

       public void onCreate(Bundle icicle) {
                        super.onCreate(icicle);
                        if(state_var_A == 0){
                            show_layout_01();
                            goto_state_01();
                        }
       }
       private void show_layout_01(){
                    LinearLayout layout = new LinearLayout(this);
                        layout.setOrientation(LinearLayout.VERTICAL);

                        btn = new Button(this);
                        btn.setBackgroundResource(R.drawable.heart);
                        btn.setId(101);
                        btn.setText("Run");
                        btn.setOnClickListener(this);
                        LinearLayout.LayoutParams param =
                                new LinearLayout.LayoutParams(120, 55);
                        param.topMargin = 10;
                        layout.addView(btn, param);

                        btn2 = new Button(this);
                        btn2.setBackgroundResource(R.drawable.heart);
                        btn2.setId(102);
                        btn2.setText("Exit");
                        btn2.setOnClickListener(this);
                        layout.addView(btn2, param);

                        tv = new TextView(this);
                    tv.setTextColor(Color.WHITE);
                    tv.setText("");
                    LinearLayout.LayoutParams param2 =
                                new LinearLayout.LayoutParams(FP, WC);
                    param2.topMargin = 10;
                    layout.addView(tv, param2);
                        setContentView(layout);
                }
     private void goto_state_01(){
            state_var_A = 1;
           bindService(new Intent("com.misoo.pk01.REMOTE_SERVICE"),
```

```
                        mConnection, Context.BIND_AUTO_CREATE);
    }
    private ServiceConnection mConnection = new ServiceConnection() {
            public void onServiceConnected(ComponentName className, IBinder ibinder) {
                                ia = IA.Stub.asInterface(ibinder);
                        }
             public void onServiceDisconnected(ComponentName className) { }
     };

     public void onClick(View v) {
      int ret=0;
            switch(v.getId()){
               case 101:
                     if(state_var_A == 1){
                           try {
                                   ret = ia.f1(188);
                            } catch (Exception e) {
                              e.printStackTrace();
                            }
                           tv.setText(String.valueOf(ret));
                     }
                  break;
               case 102:
                     if(state_var_A == 1) {
                          stopService(new Intent("com.misoo.pk01.REMOTE_SERVICE"));
                         finish();
                     }
                   break;
            }
      }
    }
```

結語

在傳統觀點裡，大多先繪製 UML 模型圖，然後才開始構思程序碼的撰寫，使得 UML 建模成爲撰寫程序碼的前置工作，因此許多程序員將 UML 建模視爲多餘的負擔。爲了節省開發成本，就將省略掉 UML 建模的工作了。

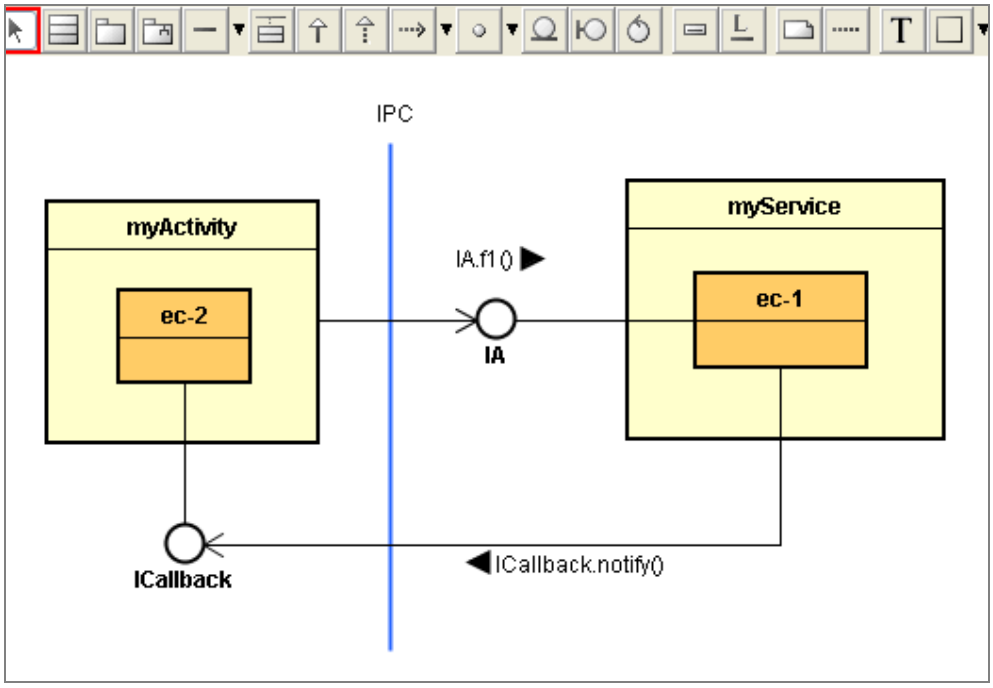
在新潮的觀點裡，UML 模型與程序碼是軟件系統本體的兩個觀點(或面向)，兩者沒有先後順序關係，而是並存和兼具於同一個人的腦海裡。這就像兩隻眼睛看到的景象並存於一個人的腦海裡一般，如此才能看到更真實的世界，也能做出更完美的軟件系統來。從本文的範例，你可看到當 UML 模型與程序碼兩個觀點一致時，真的能讓軟件系統既可靠又高雅，不亦美哉!◆

附錄：演練範例 1

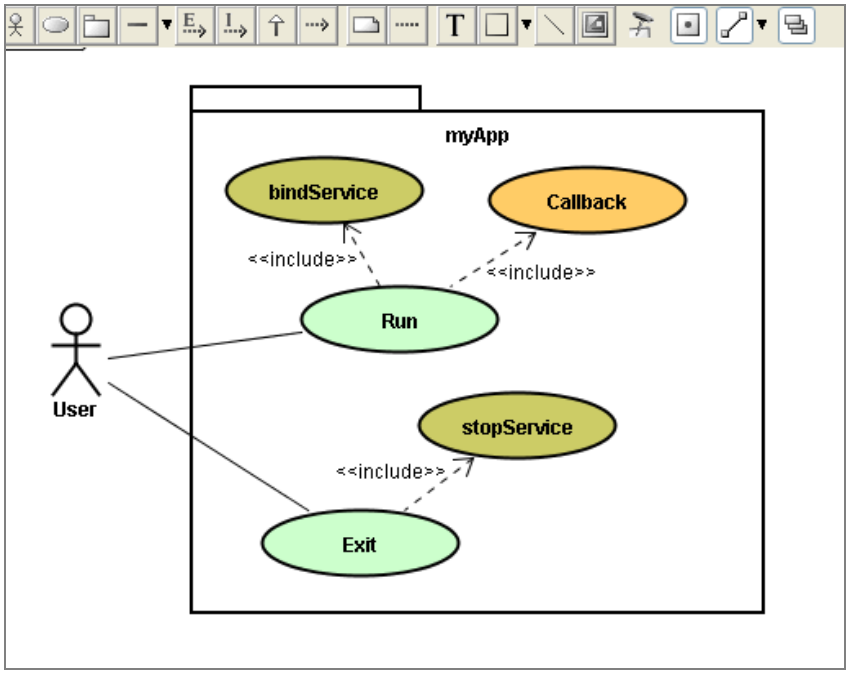
上述的範例是單向的 IPC 呼叫，線再來演練一個雙向的 IPC 範例。

模型觀點：

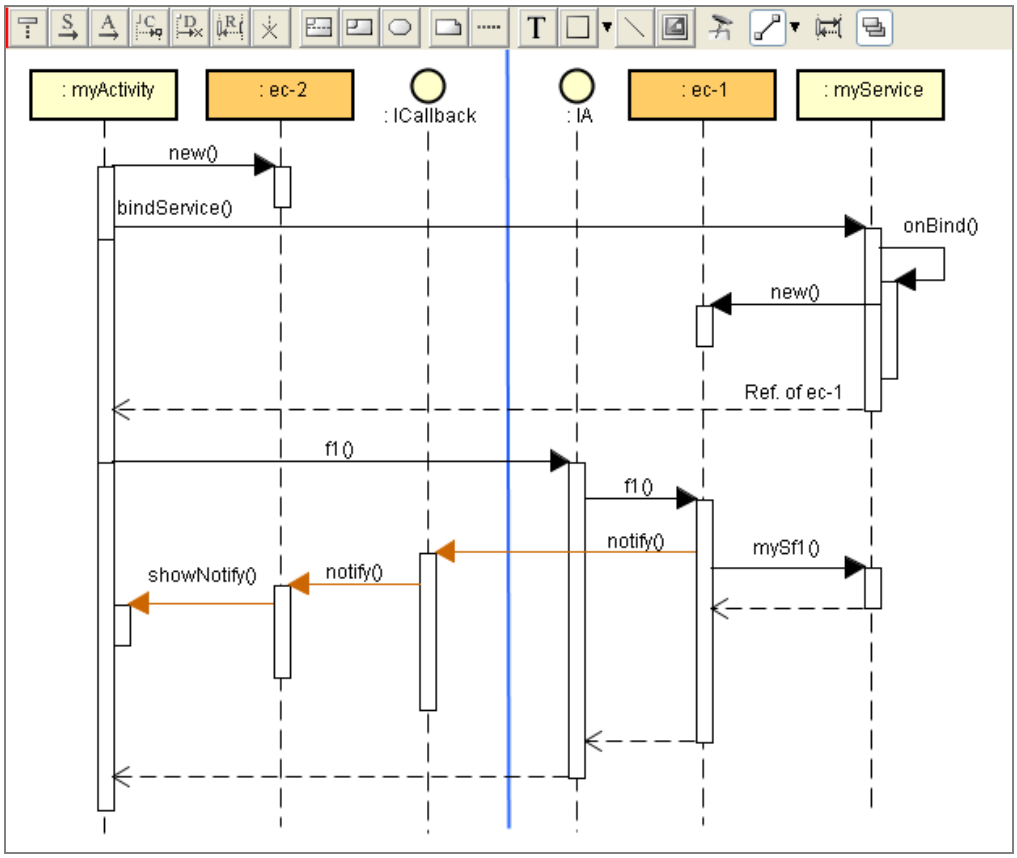
UML 類別圖：



UML 用例圖：



UML 順序圖：



程序碼觀點：

Android 開發 Proj: IB-13


```
// IA.java
package com.misoo.pk01;
import android.os.Binder;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;

public interface IA {
	int f1(IBinder cb, int k)throws RemoteException;
	//-----------------------------------------------------
```

```
    public static abstract class Stub extends Binder implements IA {
     @Override
            public boolean onTransact(int code, Parcel data, Parcel reply, int flags)
            throws android.os.RemoteException{
            IBinder cb = data.readStrongBinder();
            int k = data.readInt();
                  int ret = f1(cb, k);
                  reply.writeInt(ret);
                   return true;
            }
       public abstract int f1(IBinder ibinder, int k) throws RemoteException;
      public static IA asInterface(IBinder obj){
        return new Proxy(obj);
      }
       //-----------------------------------------
      private static class Proxy implements IA{
     private IBinder mRemote;

     public Proxy(IBinder ibinder){
            mRemote = ibinder;
     }

     public int f1(IBinder cb, int k) throws RemoteException {
            // TODO Auto-generated method stub
            Parcel data = Parcel.obtain();
            data.writeStrongBinder(cb);
            data.writeInt(k);
            Parcel reply = Parcel.obtain();
            mRemote.transact(0, data, reply, 0);
            return reply.readInt();
     }
    }
   }
}
```

// ICallback.java

```
package com.misoo.pk01;
import android.os.Binder;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;

public interface ICallback {
          void notify(String info)throws RemoteException;
          //-----------------------------------------------------
          public static abstract class Stub extends Binder implements ICallback
          {
           @Override
                public boolean onTransact(int code, Parcel data, Parcel reply, int flags)
                throws android.os.RemoteException{
                String info = data.readString();
                      this.notify(info);
                       return true;
                }
             public abstract void notify(String tinfo) throws RemoteException;
            public static ICallback asInterface(IBinder obj){
              return new Proxy(obj);
            }
             //-----------------------------------------
```

```
        private static class Proxy implements ICallback{
            private IBinder mRemote;

            public Proxy(IBinder ibinder){
                mRemote = ibinder;
            }

            public void notify(String info) throws RemoteException {
                // TODO Auto-generated method stub
                Parcel data = Parcel.obtain();
                data.writeString(info);
                Parcel reply = Parcel.obtain();
                mRemote.transact(0, data, reply, 0);
            }
        }
    }
}
```

// myService.java

```
package com.misoo.pk01;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;

public class myService extends Service {
    private ICallback ci;

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
    public int mySf1(int x){
        return x + 3000;
    }
    //-------------------------------------------
    private final IA.Stub mBinder = new IA.Stub() {
        @Override
        public int f1(IBinder cb, int k) throws RemoteException {
            ci = ICallback.Stub.asInterface(cb);
            k += 100;
            try {
                ci.notify("callback: " + String.valueOf(k));
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            return mySf1(k);
        }
    };
}
```

// myActivity.java

```
package com.misoo.pk01;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
```

```
import android.graphics.Color;
import android.os.Bundle;
import android.os.IBinder;
import android.os.RemoteException;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class myActivity extends Activity implements OnClickListener {
        private final int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
        private final int FP = LinearLayout.LayoutParams.FILL_PARENT;
        private Button btn, btn2;
        private TextView tv;
        private IA ia;
        private int state_var_A = 0;

      public void onCreate(Bundle icicle) {
                        super.onCreate(icicle);
                        if(state_var_A == 0){
                             show_layout_01();
                             goto_state_01();
                        }
      }
      private void show_layout_01(){
                        LinearLayout layout = new LinearLayout(this);
                        layout.setOrientation(LinearLayout.VERTICAL);

                        btn = new Button(this);
                        btn.setBackgroundResource(R.drawable.heart);
                        btn.setId(101);
                        btn.setText("Run");
                        btn.setOnClickListener(this);
                        LinearLayout.LayoutParams param =
                              new LinearLayout.LayoutParams(120, 55);
                        param.topMargin = 10;
                        layout.addView(btn, param);

                        btn2 = new Button(this);
                        btn2.setBackgroundResource(R.drawable.heart);
                        btn2.setId(102);
                        btn2.setText("Exit");
                        btn2.setOnClickListener(this);
                        layout.addView(btn2, param);

                        tv = new TextView(this);
                    tv.setTextColor(Color.WHITE);
                    tv.setText("");
                    LinearLayout.LayoutParams param2 = new LinearLayout.LayoutParams(FP, WC);
                    param2.topMargin = 10;
                    layout.addView(tv, param2);
                      setContentView(layout);
      }
      private void goto_state_01(){
                  state_var_A = 1;
                       bindService(new Intent("com.misoo.pk01.REMOTE_SERVICE"),
                                        mConnection, Context.BIND_AUTO_CREATE);
      }
      private ServiceConnection mConnection = new ServiceConnection() {
```

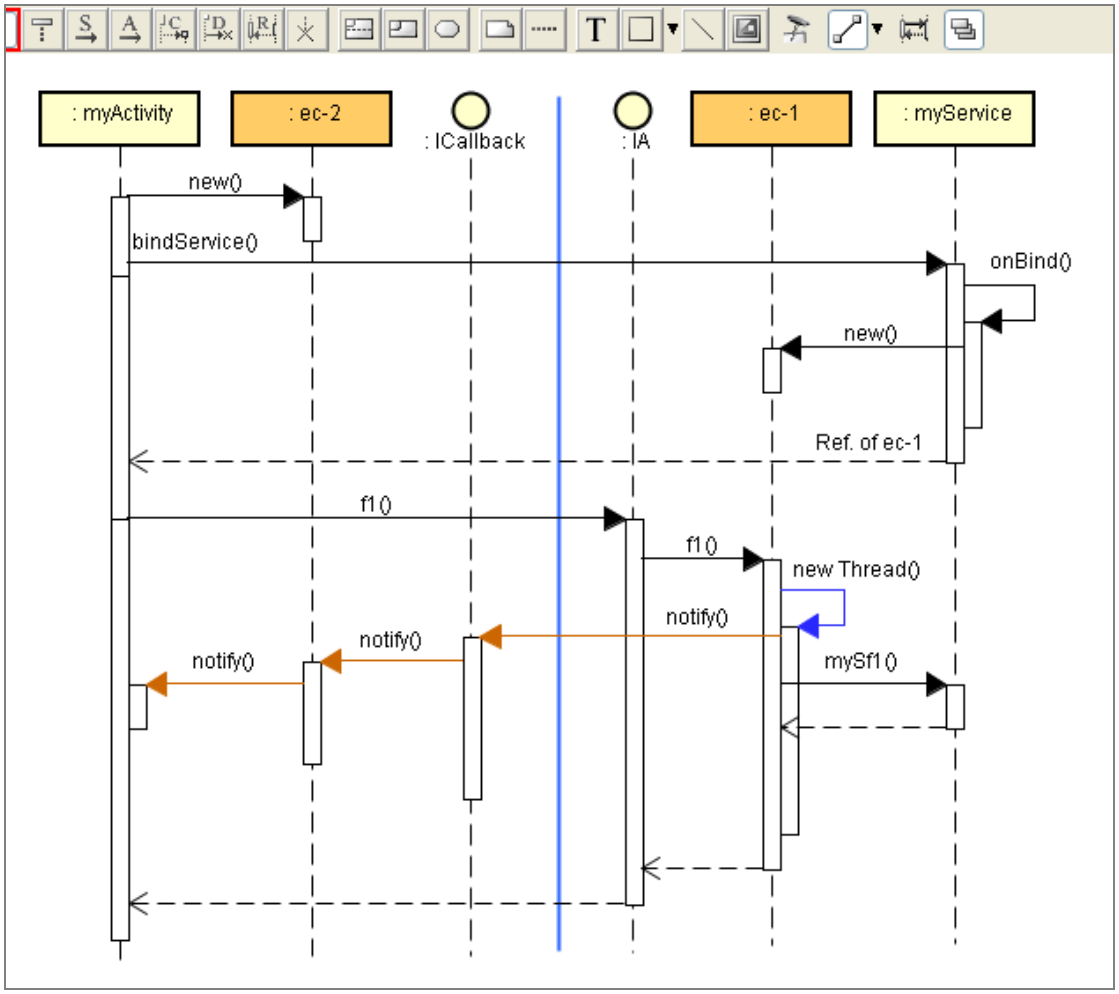
```
                        public void onServiceConnected(ComponentName className, IBinder ibinder)
{
                                ia = IA.Stub.asInterface(ibinder);
                        }
                            public void onServiceDisconnected(ComponentName className) { }
    };

    public void onClick(View v) {
            switch(v.getId()){
            case 101:
                    if(state_var_A == 1){
                            try {
                                int ret = ia.f1(cbBinder, 555);
                                tv.setText(String.valueOf(ret));
                           } catch (Exception e) {
                            e.printStackTrace();
                            }
                    }
                   break;
            case 102:
                    if(state_var_A == 1){
                            stopService(new Intent("com.misoo.pk01x.REMOTE_SERVICE"));
                             finish();
                    }
                    break;
            }
     }
    //---------------------------------
    public void showNotify(String info){
     setTitle(String.valueOf(info));
    }
    //-----------------------------------------
     private final ICallback.Stub cbBinder = new ICallback.Stub() {
            @Override
            public void notify(String info) throws RemoteException {
                    showNotify(info);
                    }
            };
   }
```

附錄：演練範例 2

上述的範例是雙向的 IPC 範例，而且都是由主線程負責執行；現在來練習遊子線程來執行 callback 的動作。

模型觀點：

UML 順序圖：



程序碼觀點：

Android 開發 Proj: IB-14


```
// IA.java
package com.misoo.pk01;
import android.os.Binder;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;

public interface IA {
	int f1(IBinder cb, int k)throws RemoteException;
	//-----------------------------------------------------
```

```
    public static abstract class Stub extends Binder implements IA {
     @Override
            public boolean onTransact(int code, Parcel data, Parcel reply, int flags)
            throws android.os.RemoteException{
            IBinder cb = data.readStrongBinder();
            int k = data.readInt();
                  int ret = f1(cb, k);
                  reply.writeInt(ret);
                   return true;
            }
       public abstract int f1(IBinder ibinder, int k) throws RemoteException;
      public static IA asInterface(IBinder obj){
        return new Proxy(obj);
      }
       //---------------------------------------------
      private static class Proxy implements IA{
     private IBinder mRemote;

     public Proxy(IBinder ibinder){
            mRemote = ibinder;
     }

     public int f1(IBinder cb, int k) throws RemoteException {
            // TODO Auto-generated method stub
            Parcel data = Parcel.obtain();
            data.writeStrongBinder(cb);
            data.writeInt(k);
            Parcel reply = Parcel.obtain();
            mRemote.transact(0, data, reply, 0);
            return reply.readInt();
     }
    }
   }
}
```

// ICallback.java

```
package com.misoo.pk01;
import android.os.Binder;
import android.os.IBinder;
import android.os.Parcel;
import android.os.RemoteException;

public interface ICallback {
          void notify(String info)throws RemoteException;
          //-------------------------------------------------------
          public static abstract class Stub extends Binder implements ICallback
          {
           @Override
                  public boolean onTransact(int code, Parcel data, Parcel reply, int flags)
                  throws android.os.RemoteException{
                  String info = data.readString();
                        this.notify(info);
                         return true;
                  }
             public abstract void notify(String tinfo) throws RemoteException;
            public static ICallback asInterface(IBinder obj){
              return new Proxy(obj);
```

```
            }
             //-------------------------------------------
            private static class Proxy implements ICallback{
            private IBinder mRemote;

            public Proxy(IBinder ibinder){
                    mRemote = ibinder;
            }

            public void notify(String info) throws RemoteException {
                    // TODO Auto-generated method stub
                    Parcel data = Parcel.obtain();
                    data.writeString(info);
                    Parcel reply = Parcel.obtain();
                    mRemote.transact(0, data, reply, 0);
            }
           }
    }
}
```

// myService.java

```
package com.misoo.pk01;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;

public class myService extends Service {
    private ICallback ci;

    @Override
    public IBinder onBind(Intent intent) {
            return mBinder;
    }
    public int mySf1(int x){
            return x + 5000;
    }
    public void callback(int k){
            k += 100;
            try {
                    ci.notify("callback: " + String.valueOf(k));
            } catch (RemoteException e) {
                    e.printStackTrace();
            }
      }
     //-------------------------------------------
    private final IA.Stub mBinder = new IA.Stub() {
            @Override
            public int f1(IBinder cb, final int k) throws RemoteException {
                    ci = ICallback.Stub.asInterface(cb);
                    new Thread(){
                            public void run() {
                                    callback(k);
                            }}.start();
                   return mySf1(k);
            }
    };
}
```

```
// myActivity.java
package com.misoo.pk01;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Looper;
import android.os.Message;
import android.os.RemoteException;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class myActivity extends Activity implements OnClickListener {
        private final int WC = LinearLayout.LayoutParams.WRAP_CONTENT;
        private final int FP = LinearLayout.LayoutParams.FILL_PARENT;
        private Button btn, btn2;
        private TextView tv;
        private IA ia;
        private int state_var_A = 0;
        private Handler h;

      public void onCreate(Bundle icicle) {
                        super.onCreate(icicle);
                        if(state_var_A == 0){
                             show_layout_01();
                             goto_state_01();
                        }
      }
      private void show_layout_01(){
                        LinearLayout layout = new LinearLayout(this);
                        layout.setOrientation(LinearLayout.VERTICAL);

                        btn = new Button(this);
                        btn.setBackgroundResource(R.drawable.heart);
                        btn.setId(101);
                        btn.setText("Run");
                        btn.setOnClickListener(this);
                        LinearLayout.LayoutParams param =
                              new LinearLayout.LayoutParams(120, 55);
                        param.topMargin = 10;
                        layout.addView(btn, param);

                        btn2 = new Button(this);
                        btn2.setBackgroundResource(R.drawable.heart);
                        btn2.setId(102);
                        btn2.setText("Exit");
                        btn2.setOnClickListener(this);
                        layout.addView(btn2, param);

                        tv = new TextView(this);
```

```
                tv.setTextColor(Color.WHITE);
                tv.setText("");
                LinearLayout.LayoutParams param2 = new LinearLayout.LayoutParams(FP, WC);
                param2.topMargin = 10;
                layout.addView(tv, param2);
                  setContentView(layout);
    }
    private void goto_state_01(){
            state_var_A = 1;
                bindService(new Intent("com.misoo.pk01.REMOTE_SERVICE"),
                                mConnection, Context.BIND_AUTO_CREATE);
                //---------------------------------------
                h = new Handler(Looper.getMainLooper()){
                        public void handleMessage(Message msg) {
                            setTitle((String)msg.obj);
                    }
                    };
    }
    private ServiceConnection mConnection = new ServiceConnection() {
                    public void onServiceConnected(ComponentName className, IBinder ibinder)
{
                            ia = IA.Stub.asInterface(ibinder);
                    }
                        public void onServiceDisconnected(ComponentName className) { }
    };

    public void onClick(View v) {
            switch(v.getId()){
            case 101:
                if(state_var_A == 1){
                        try {
                            int ret = ia.f1(cbBinder, 555);
                            tv.setText(String.valueOf(ret));
                        } catch (Exception e) {
                         e.printStackTrace();
                         }
                }
               break;
            case 102:
                if(state_var_A == 1){
                        stopService(new Intent("com.misoo.pk01x.REMOTE_SERVICE"));
                         finish();
                }
                break;
            }
     }
    //---------------------------------
    public void showNotify(String info){
     setTitle(String.valueOf(info));
    }
    //-----------------------------------------
     private final ICallback.Stub cbBinder = new ICallback.Stub() {
            @Override
            public void notify(String info) throws RemoteException {
                 h.removeMessages(0);
                Message m = h.obtainMessage(1, 1, 1, info);
               h.sendMessage(m);
            }
     };
   }
```

~~ END ~~