코드 중복 없이 안드로이드 컴포넌트에 예외 처리를 추가하는 클래스 설계 방법에 관한 연구

김현순 이승휘 이화중 최광훈

연세대학교 컴퓨터정보통신공학부
강원도 원주시 흥업면 연세대길 1
{coli,sunkus711,hj_tl22,kwanghoon.choi}@yonsei.ac.kr

창병모

숙명여자대학교 컴퓨터과학과
서울 용산구 청파로 47 길 100
chang@sookmyung.ac.kr

요약: 안드로이드 앱의 취약점으로 비정상 종료되는 경우 발생하는 예외를 처리하기 위해 안드로이드 컴포넌트 구조에 적합한 예외처리 방법이 필요하다. 안드로이드 컴포넌트 기반 예외처리 방법에 관한 이전 연구에서 해결책을 제시한 바가 있으나 유사한 컴포넌트 클래스들이 동일한 코드를 중복해서 갖는 문제가 있었다. 이 논문은 중복 코드 없이 안드로이드 컴포넌트에 예외처리 기능을 추가한 클래스를 설계하는 방법을 제안한다.

핵심어: 안드로이드, 예외, 클래스 설계, 디자인 패턴

1. 서론

안드로이드[1]는 모바일 디바이스를 위한 구글의 오픈소스 플랫폼이다. 안드로이드 앱은 안드로이드 플랫폼 API 를 사용하는 자바 프로그램으로, 액티비티, 서비스, 브로드캐스트 리시버, 컨텐트프로바이더라 부르는 네 가지 유형의 안드로이드 컴포넌트로 구성된다.

최근에 안드로이드 앱의 취약점으로 비정상 종료되는 사례가 빈번하게 발생한다고 보고된 바 있고[2], 안드로이드 예외에 관한 연구[3]에서 비정상 종료 시 발생하는 예외를 안드로이드 앱의 구조에 적합한 새로운 방식으로 처리함으로써 예상하지 못한 예외 발생에 적극적으로 대응하는 방법을 제안하였다.

이 연구에서 안드로이드 컴포넌트 클래스를 확장한 안드로이드 플랫폼과 앱의 경계에 방어벽을 놓아 앱에서 발생한 예외가 플랫폼으로 전달되어 비정상 종료되는 것을 막는 방법을 제안하였다. 이 방법을 클래스 라이브러리로 제공하여 개발자가 이 라이브러리를 사용하여 앱을 개발하면 비정상 종료되는 상황을 막을 수 있다. 안드로이드 플랫폼과 앱의 구조는 전혀 변경하지 않고 기존 앱 소스를 최소로 변경만 해도 적용 가능한 장점이 있다[3].

그림 1 에서 안드로이드 액티비티 컴포넌트 클래스(Activity, ListActivity, PreferenceActivity)에 예외 기능을 추가한 새로운 컴포넌트 클래스(ExceptionActivity, ExceptionListActivity, ExceptionPreferenceActivity)간의 클래스 다이어그램을 보여준다. 기존 앱에서 Activity, ListActivity, PreferenceActivity 클래스를 상속받아 액티비티를 만들지만, 안드로이드 예외를 고려한 액티비티를 사용하려면 Exception-으로 시작하는 해당하는 클래스를 상속받는다.

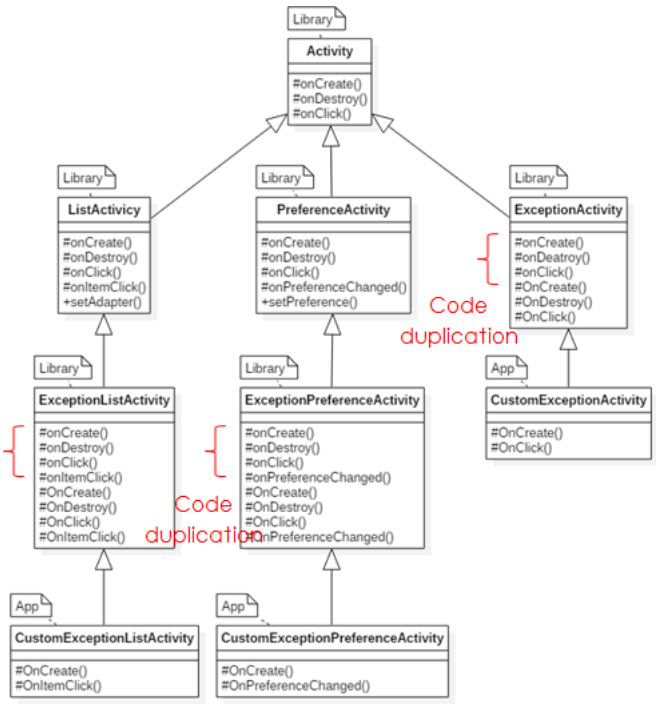

그림 1 클래스 다이어그램

하지만 안드로이드 예외 클래스 라이브러리[3]의 문제점은 동일한 유형의 컴포넌트 클래스들에서 앱과 플랫폼 사이의 방어벽에 해당하는 코드를 제대로 공유하지 못하고 각 클래스마다 방어벽 코드를 중복해서 포함시켜야 했다. 그 이유는 기존 안드로이드 컴포넌트 클래스들의 상속 관계가 예외를 추가한 컴포넌트 클래스들 사이의 상속 관계로 유지되지 않는 방법으로 설계했기 때문이다. 즉, Activity 클래스를 상속받아 예외 기능을 추가해서 ExceptionActivity 클래스를 정의하고, Activity 클래스의 자식 클래스

ListActivity 를 상속받아 예외 기능을 추가하여 ExceptionListActivity 클래스를 정의했지만[3], 새로 정의한 두 예외 클래스는 서로 상속 관계가 아니다. 따라서 ExceptionActivity 에 포함된 방어벽 코드를 ExceptionListActivity 에서 공유하지 못하고 중복해서 코드를 포함시켰다. 이와 유사한 상황이 PreferenceActivity 뿐만 아니라 Activity 의 모든 자식 클래스에 대해서도 벌어진다.

이 논문은 안드로이드 예외 클래스에서 앞에서 설명한 중복된 코드가 없는 설계 방법을 제안한다. 2 장에서 안드로이드 예외 클래스를 소개하고 그 문제점을 상세히 설명한다. 3 장에서는 새로운 클래스 설계 방법을 해결책으로 제시한다. 4 장에서 향후 진행할 연구 주제와 함께 결론을 맺는다.

2. 안드로이드 예외 처리 방법

안드로이드 앱의 취약점으로 예상하지 못한 이유로 비정상 종료되는 사례가 빈번하게 발생하고, 이에 대응하고자 안드로이드 앱의 구조에 적합한 예외 처리 방법을 [3]의 연구에서 제안하였다.

2.1 안드로이드 앱의 컴포넌트 구조와 예외 처리 방법

안드로이드 앱은 4 가지 유형의 컴포넌트들로 구성되어 있다. UI 화면을 담당하는 액티비티, 백그라운드 태스크를 제어하는 서비스, 시스템 전체 알림 메시지를 받는 브로드캐스트리시버, 여러 앱이 데이터를 공유하기 위한 컨텐트프로바이더가 있다.

안드로이드 앱은 보통 자바 프로그램 형태로 개발하고, 자바 언어는 비정상 상황에 대응하는 두 가지 예외 처리 방법을 제공한다. 기본적으로, 문장이나 블록 단위에서 예외가 발생하면 try/catch 로 대응하고, 여기에서 처리하지 못한 예외는 함수 호출 역순으로 전파된다. 쓰레드 안에서 처리하지 못한 예외(예를 들어, main 메소드까지 전파되었으나 처리하지 못한 예외)는 Thread.UncaughtExceptionHandler 인터페이스를 통해 대응할 수 있다.

표 1 예외 처리 방법

예외처리 단위	자바 프로그램	안드로이드 프로그램
문장/블록	● try/catch ● 예외 전파 (함수호출 역순)	동일
안드로이드 컴포넌트	n/a	예외처리를 추가한 컴포넌트 [3]
스레드	Thread.UncaughtExceptionHandler	동일

하지만 안드로이드 앱의 컴포넌트에 적합한 예외 처리 방법은 자바 언어에서 제공하지 않는다. 안드로이드 컴포넌트는 문장이나 블록을 포함하는 더 큰 코드 단위이고, 여러 컴포넌트를 포함하는 스레드 보다는 더 작은 코드 단위이다. 따라서, 안드로이드 컴포넌트 단위에 적합한 예외처리 방법이 필요하다.

안드로이드 컴포넌트 기반 예외처리는 두 가지 방법으로 구성되어 있다[3].

- 컴포넌트 내 예외처리: 각 컴포넌트에 독립적인 catch 블록을 정의하여 컴포넌트를 실행하다 발생했으나 그 안에서 처리하지 못한 예외를 모두 여기로 전달된다.
- 컴포넌트 간 예외처리: 컴포넌트간 호출 관계의 역순으로 예외를 전파시킨다.

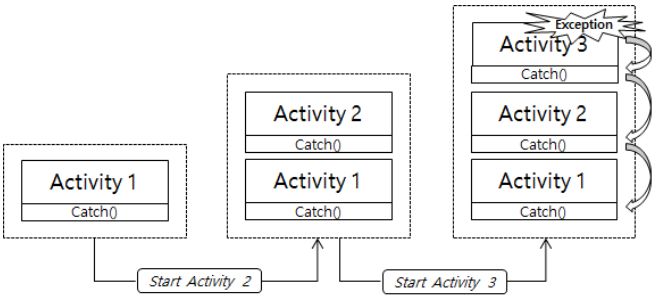

그림 2 안드로이드 컴포넌트 기반 예외처리

액티비티 컴포넌트를 예로 들어 예외 처리하는 방법을 그림 2 에서 보여준다. 각 액티비티에 개별적으로 Catch() (메소드로 표현한 catch 블록)을 두어 컴포넌트 내 예외처리를 한다. 그림과 같이 액티비티 1 에서 액티비티 2 를 거쳐 액티비티 3 을 호출한 상황에서 발생한 예외는 호출 역순으로 액티비티 3 의 Catch(), 액티비티 2, 액티비티 1 로 전파된다. 만일 이 전파 경로에서 아무도 처리하지 못하면 마침내 안드로이드 플랫폼으로 예외를 전달한다.

안드로이드 컴포넌트 예외처리 방법을 실현하기 위해서 다음과 같은 방법으로 구현하였다.

- 기존 안드로이드 컴포넌트 클래스를 상속받아 예외처리를 추가한 새로운 클래스를 정의하였다.
- 안드로이드 앱을 작성할 때 기존 컴포넌트 클래스 대신 새로 정의한 클래스를 상속받는다.
- 예외 기능을 추가한 새 클래스는 기존 클래스의 각 콜백 메소드(onM)에 상응하는 새로운 콜백 메소드(OnM)을 새로 추가한다. 기존 콜백 메소드를 오버라이딩하여 새로 정의한 콜백 메소드를 호출하고, 이 호출을 감싸는 try/catch 블록을 두어 예외 장벽을 쌓는다.
- 안드로이드 앱을 작성할 때 기존 콜백 메소드 대신 새로운 콜백 메소드를 오버라이딩한다.

예를 들어, 기존 액티비티 클래스 Activity 를 상속

받아 예외 기능을 추가한 액티비티 클래스 ExceptionActivity 를 정의한다. 기존 콜백 메소드 onCreate, onDestroy, onClick 등에 대해 새로운 콜백 메소드 OnCreate, OnDestroy, OnClick 을 새 클래스에 추가한다. ExceptionActivity 클래스에서 Activity 의 onCreate, onDestroy, onClick 메소드를 오버라이딩하고, 각각 OnCreate, OnDestroy, OnClick 을 호출한다. 이 호출을 감싸는 try/catch 블록을 두어 예외를 장벽 코드를 쌓는다. 안드로이드 앱은 ExceptionActivity 를 상속받고 OnCreate, OnDestroy, OnClick 메소드를 오버라이딩한다.

그림 1 은 액티비티 컴포넌트 클래스와 [3]의 연구에서 설계한 예외 기능을 추가한 액티비티 컴포넌트 클래스의 상속 관계를 설명하는 클래스 다이어그램이다. 설명에 필요한 정도의 클래스와 메소드 만 이 다이어그램에 포함했고, 그 외 상세 내용에 대해서는 안드로이드 플랫폼 API 문서[1]을 참고한다.

그림 3 은 예외 기능으로 확장한 액티비티 클래스를 상속받아 안드로이드 앱의 클래스 CustomExceptionActivity 를 정의했을 때 객체들 간의 상호작용을 표현한 순차 다이어그램이다.

- 안드로이드 앱을 띄우면 플랫폼에서 (CustomExceptionActivity 클래스가 상속 받은) EA 클래스의 onCreate 메소드를 호출한다.
- 그 다음, (ExceptionActivit 의 OnCreate 메소드를 오버라이딩한) CustomExceptionActivity 클래스의 OnCreate 메소드를 호출한다.
- CustomExceptionActivity.OnCreate 메소드에서 예외가 발생하면 ExceptionActivity.onCreate 메소드에서 OnCreate 메소드 호출을 감싸는 try/catch 블록으로 이 예외를 잡을 것이다.

그림 1 과 3 의 상세 코드는 아래의 사이트에서 다운받아 확인할 수 있다.

- https://mobilesw.yonsei.ac.kr/paper/androidexceptionredesigned.zip (com.example.java 패키지)

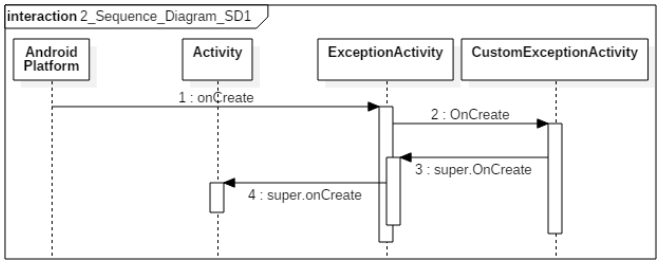

그림 3 순차 다이어그램

2.2 안드로이드 컴포넌트 기반 예외 처리 방법의 코드 중복 문제점

안드로이드 컴포넌트에 예외처리를 추가한 연구의 클래스 설계 방법[3]에서 코드가 중복되는 문제점이 있다. 그림 1 의 클래스 다이어그램을 살펴보면, Activity 클래스를 상속받아 ListActivity 를 정의하였지만 Activity 와 연관된 ExceptionActivity 클래스는 ListActivity 와 연관된 ExceptionListActivity 클래스와 서로 상속 관계에 있지 않다. 만일 이 예외 기능을 갖춘 두 클래스들이 상속 관계에 있었다면, ExceptionListActivity 클래스는 ListActivity 에서 새로 정의한 onItemClick 과 OnItemClick 메소드만을 새로 정의하고 Activity 에서 정의한 onCreate, onDestroy, onClick 메소드는 ExceptionActivity 클래스로부터 상속받아 재사용했을 것이다. 따라서 ExceptionActivity 에서 정의한 예외 장벽코드를 포함하는 onCreate 메소드를 ExceptionListActivity 에서 동일한 코드를 갖는 onCreate 메소드를 다시 정의해야 했다.

이러한 코드 중복 문제는 매우 심각하다. 첫째, 그림 1 에서 나열한 것 보다 Activity 클래스의 더 많은 메소드(onActivityResult 등)[1]에서 발생한다. 둘째, Activity 를 상속받아 정의한 모든 액티비티 클래스에 예외 기능을 추가할 때 똑같은 문제가 일어난다. 유사한 상황이 다른 부류의 컴포넌트 클래스 Service, BroadcastReceiver, ContentProvider 에 예외를 추가할 때도 생긴다. 셋째, 작은 화면을 차지하는 액티비티인 Fragment 클래스와 액티비티 화면 내의 UI 를 구성하는 View 클래스에 예외를 추가할 때도 비슷한 문제가 발생한다.

ExceptionListActivity 에서 두 가지 클래스 ListActivity 와 ExceptionActivity 를 모두 상속받는 다중 상속으로 안드로이드 예외 클래스의 코드 중복 문제를 쉽게 해결할 수 있을 것으로 보이지만 그렇지 않다. 다음 절에서 그 이유를 설명한다.

2.3 다중 상속을 사용한 코드 중복을 해결하는 접근 방법의 문제점

다이아몬드 형태의 다중 상속을 가정하고 코드 중복 문제를 해결할 수 있는지 살펴보자. 즉, Activity 를 상속받아 ExceptionActivity 클래스를 정의하고 Activity 의 자식클래스 ListActivity 와 함께 ExceptionActivity 를 상속받아 ExceptionListActivity 클래스를 정의한다고 가정한다. 물론 자바 언어로 이 가정대로 안드로이드 프로그램을 작성할 수 없다. 왜냐하면 자바 언어는 단일 상속만 허용하고 (메소드 선언만을 허용하는) 인터페이스에 대해서만 다중 상속을 지원한다. 비록 클래스 다중 상속은 안드로이드 프로그램에 적용할 수 있는 가정은 아니지만, 다중 상속으로 정의한 ExceptionListActivity 클래스의 onCreate 메소드는 ExceptionActivity 클래스에서 물려 받은 것이다. 이 onCreate 메소드는 super.onCreate()를 통해 부모 클래스의 onCreate 메

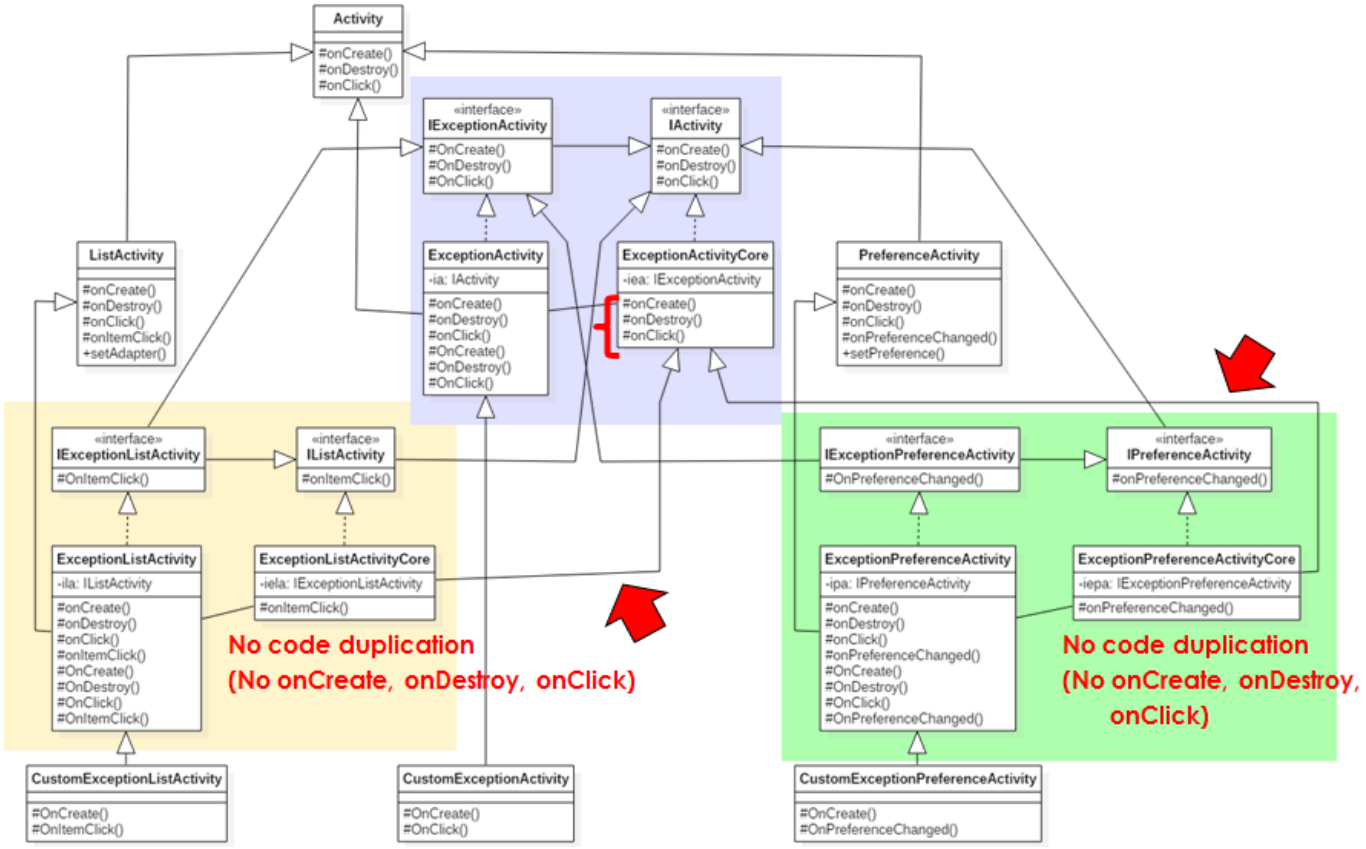

그림 4 새로운 방법으로 설계한 클래스 다이어그램

소드를 먼저 호출해야 한다[1]. 이때 super 는 Activity 객체를 가리키거나 ListActivity 객체를 가리킨다. ExceptionActivity 를 상속받아 앱의 액티비티를 구성한다면 Activity 객체이고, ExceptionListActivity 를 상속받는 경우라면 ListActivity 객체이다. 자바 언어는 단일 상속만 허용하므로 동적으로 두 가지 클래스의 객체를 선택해서 메소드를 호출하는 방법을 제공하지 않는다.

다중 상속을 제공하는 C++언어에서도, 두 가지 이상의 경로를 통해 동일한 이름의 메소드를 상속받는 경우 호출하고자 하는 메소드 이름(m)에 구체적인 클래스 이름(C)을 지정해서(C::m) 구분한다. C++ 언어에서도 동적으로 객체에 따라 이 클래스 이름을 다르게 지정해서 다중 상속에서 동일한 이름의 여러 메소드를 구분하는 방법을 제공하지 않는다.

논의한 사항은 사실 다중 상속 연구에서 해결하지 못한 문제이다. 다중 상속 문제를 해결하면 안드로이드 컴포넌트 기반 예외 처리 방법의 코드 중복 문제를 해결할 수 있지만, 이 다중 상속 문제를 제대로 해결한 연구는 없다.

이 논문에서 안드로이드 컴포넌트에 예외 기능을 추가할 때 코드 중복이 발생하지 않는 클래스 설계 방법을 제안한다. 다중 상속을 사용하지 않았고, 아직 다른 연구에서 시도된 바가 없다.

3. 안드로이드 컴포넌트 예외 기능을 공유하는 클래스 설계 방법

앞 절에서 설명한 코드 중복 문제를 해결하면서 안드로이드 컴포넌트에 예외 기능을 확장하는 클래스 설계 방법을 제안한다.

- 안드로이드 컴포넌트 클래스를 예외 처리 기능 관점에서 핵심(Core) 클래스와 이를 사용하는 껍데기(Shell) 클래스로 나눈다.
- 두 클래스를 서로 연관 관계(association)로 구성하여 각 클래스의 객체를 합하면 하나의 객체처럼 동작하도록 구성한다.
- 기존 클래스를 상속받아 껍데기 클래스를 정의하고 핵심 클래스들끼리 서로 상속 관계를 구성한다. 이로써 코드 중복 문제를 해결한다.

.

그림 4 는 앞에서 설명한 아이디어에 따라 설계한 클래스 다이어그램이다. Activity 클래스를 상속받아

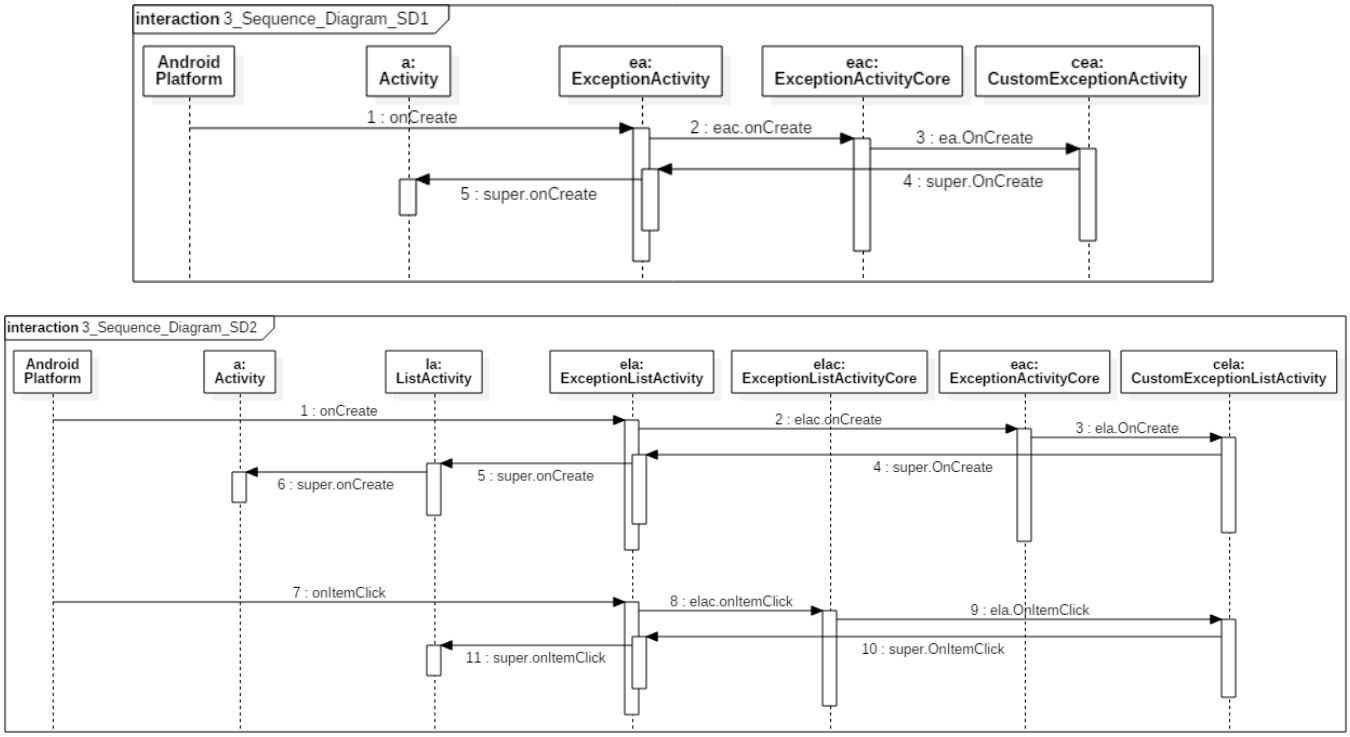

그림 5 순차 다이어그램

ExceptionActivity(Shell) 클래스를 정의하고, 이와 별개로 ExceptionActivityCore 클래스를 도입한다. 이 두 클래스는 객체 생성시 서로의 객체를 가리키는 멤버 변수를 포함하는 연관 관계를 갖도록 구성한다.

핵심 클래스는 onM 메소드만 가지고 있고, 껍데기 클래스는 onM 과 OnM 을 모두 갖추고 있다. 단, 껍데기 클래스의 onM 메소드는 단순히 핵심 클래스의 onM 을 호출하는 코드로 작성한다. 핵심 클래스의 onM 메소드는 껍데 클래스의 OnM 메소드를 호출하고 이때 발생할 수 있는 예외를 잡는 try/catch 블록을 지닌다. 이렇게 상호 객체간 호출을 하기 위해 두 클래스를 연관 관계로 구성해야 한다.

두 클래스가 서로 연관 관계를 갖도록 구현하기 위해 그림 4 와 같이, ExcetpionActivity 클래스에 IActivity 인터페이스의 멤버 변수를 선언하고, ExceptionActivityCore 클래스에 IExceptioNActivity 인터페이스의 멤버 변수를 선언한다.

IActivity 는 onCreate, onDestroy, onClick 메소드를 선언한 인터페이스이고, IExceptionActivity 는 IActivity 를 상속받아 이 메소드들을 간접적으로 선언하고 OnCreate, OnDestroy, OnClick 메소드를 추가 선언한 인터페이스이다.

ExceptionActivity 는 IExceptionActivity 인터페이스를 구현하고, ExceptionActivityCore 는 IActivity 인터페이스를 구현한다.

두 클래스의 연관 관계를 맺기 위해 도입된 이 멤버 변수는 ExceptionActivity 클래스의 객체를 생성할 때 기본 생성자에서 ExceptionActivityCore 객체를 생성하고 적절히 초기화한다.

Activity 의 자식 클래스 ListActivity 를 상속받아 정의한 ExceptionListActivity 와 ExceptionListActivityCore 도 유사한 방식으로 작성한다. IListActivity 인터페이스는 IActivity 를 상속받고, onItemClick 메소드 선언을 추가한다. IExceptionListActivity 인터페이스는 IListActivity 를 상속받고, OnItemClick 메소드 선언을 추가한다.

ExceptionListActivityCore 클래스는 문제가 되었던 중복된 코드(onCreate, onDestroy, onClick)을 ExceptionActivityCore 에서 그대로 상속받는다. 오직 onItemClick 메소드 코드를 새로 추가한다. 기존 방법[3]에서 중복해서 추가했던 코드를 클래스 상속으로 공유할 수 있다.

ExceptionPreferenceActivity 클래스에 대해서도 유사한 방식으로 작성하여 중복된 코드를 클래스 상속 관계를 통해 공유할 수 있다.

그림 5 에서 새로운 방식으로 설계한 클래스 다이어그램에 따라 작성한 안드로이드 프로그램을 실행할 때 객체간의 상호작용을 보여준다. 이전 클래스 구성[3]에서는 ExceptionActivity 클래스의 onCreate 메소드에서 앱에서 정의한 액티비티 클래스의 OnCreate 메소드를 직접 호출하였으나, 새로운 클래스 구성에서는 ExceptionActivityCore 클래스의 onCreate 메소드를 거치서 호출한다.

그림 4 와 5 의 상세 코드는 아래의 사이트에서 다운받을 수 있다.

- http://mobilesw.yonsei.ac.kr/paper/androidexceptionredesigned.zip (com.example.java.newdesign 패키지)

제안한 방법을 사용해서 안드로이드 앱을 개발하면 기존의 안드로이드 컴포넌트 예외 클래스를 사용하는 경우의 장점을 그대로 유지한다.

- 안드로이드 컴포넌트 수준에서 예외를 처리하는 능력은 기존 방법[3]과 새로 제안한 방법이 동일하다.
- 기존 방법으로 설계한 예외 클래스를 사용해서 작성한 앱을 변경하지 않아도 새로운 방식의 예외 클래스과 함께 사용할 수 있다.
- 기존 방식에서 안드로이드 플랫폼을 변경하지 않아도 되었던 것처럼 새로운 방식에서도 안드로이드 플랫폼을 수정할 필요가 없다.

4. 결론 및 향후 연구

이 논문에서 안드로이드 컴포넌트에 예외처리를 확장한 클래스를 설계하는 새로운 방법을 제안해서 기존 방법의 코드 중복 문제를 해결하였다. 이 아이디어는, 안드로이드 컴포넌트 클래스들 Activity, Service, BroadcastReceiver, ContentProvider 에 적용 가능할 뿐만 아니라 작은 액티비티를 표현하는 Fragment 클래스와 액티비티 안의 UI 블록을 구성하는 View 클래스에 대해서도 적용 가능하다.

여기에서 제안한 아이디어를 [3] 연구에서 실험한 안드로이드 앱의 벤치마크 소스에 적용하여 코드 중복을 제거한 효과를 실험할 예정이다.

향후 연구 주제로, 논문에서 제안한 클래스 설계 방법이 특별한 도메인의 구현 방법이 아니라 일반적인 도메인에서 활용할 수 있는 클래스 구조인지 살펴볼 필요가 있다. 예를 들어, 자바의 GUI 프로그램에서 사용하는 Swing 클래스나 자바 스레드 클래스에 예외처리를 확장하는 클래스를 설계할 때 응용 가능할 것으로 판단한다.

또 다른 향후 연구 주제로, 데커레이터 디자인 패턴(Decorator design pattern) [4,5]과 이 논문에서 사용한 아이디어를 비교하고자 한다. 이 디자인 패턴에서 다양한 기능의 조합을 구현할 때 연관 관계를 사용해서 코드를 공유하는 아이디어가 이 논문에서 사용한 껍데기 클래스와 핵심 클래스를 연관 관계로 구성하여 핵심 클래스의 코드를 공유하는 아이디어와 유사하다. 다만, 데커레이터 디자인 패턴 적용 사례에서는 조합하려는 기능의 클래스들을 객체 생성 시 필요에 따라 임의로 지정하는데, 이 논문에서 제안한 아이디어에서는 이미 정해진 상속 관계(예를 들어, Activity 와 ListActivity)에 따라 조합하도록 고정되어 있는 점이 다르다.

참고문헌

[1] Android API, http://developer.android.com, 2014

[2] Amiya K, Maji, Fahad A. Arshad, Saurabh Bagchi, and Jan S. Rellermeyer, “An empirical study of the robustness of Inter-component Communication in Android”, In proceedings of the 2012 42nd Annual IEEE/IFIP International comference on Dependable Systems and Networks (DSN) (DSN’12), IEEE Computer society, Washington, DC, USA, 1-12, 2012

[3] Kwanghoon Choi, Byeong-Mo Chang, A Lightweight Approach to Component-level Exception Mechanism for Robust Android Apps, Computer Languages, Systems, and Structures, Vol.44, Part C, P.283-298, December 2015.

[4] 정인상, 채흥석, Java 객체지향 디자인 패턴, 한빛미디어, 2014 년 3 월.

[5] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.

5. 감사의 글

이 논문은 2014 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.NRF-2014R1A1A2053446).