

6. 애플리케이션 구성하기

1. 레이아웃 인플레이션
2. 화면 구성과 화면 간 이동
3. 인텐트와 데이터 전달
4. 수명주기



학습내용

레이아웃 인플레이션



화면 전환



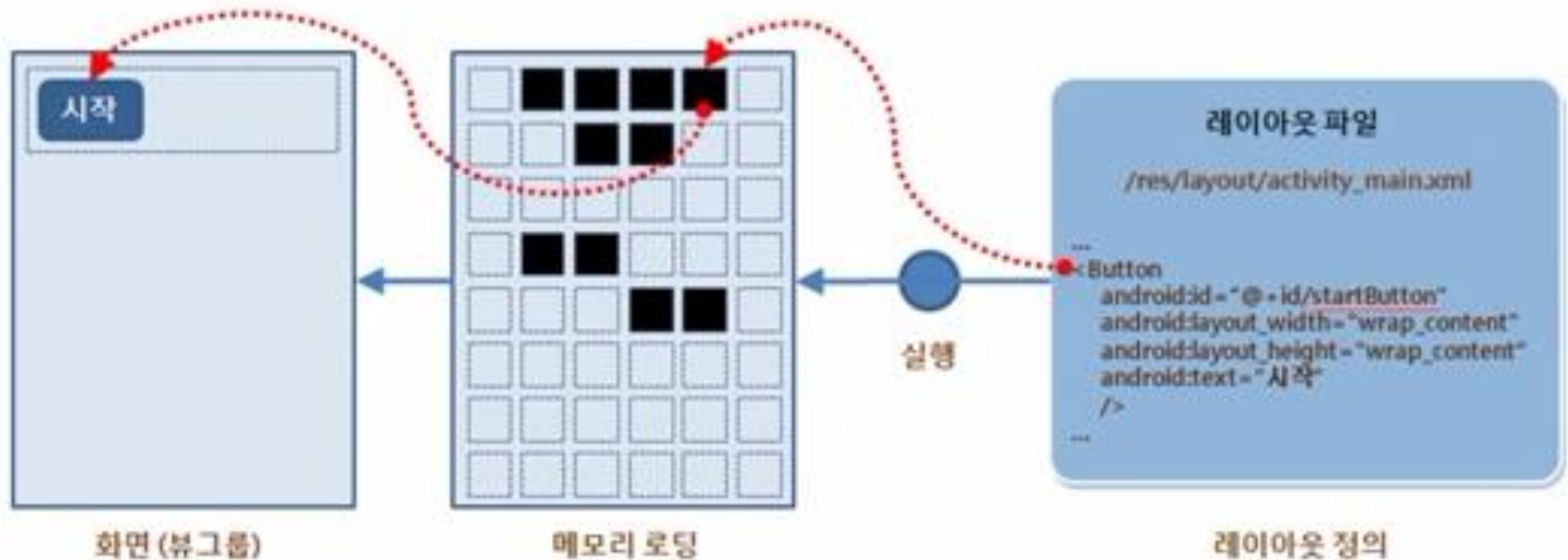
애플리케이션 구성요소



1. 레이아웃 인플레이션

□ 인플레이션이란?

- XML 레이아웃에 정의된 내용이 메모리 상에 객체화되는 과정



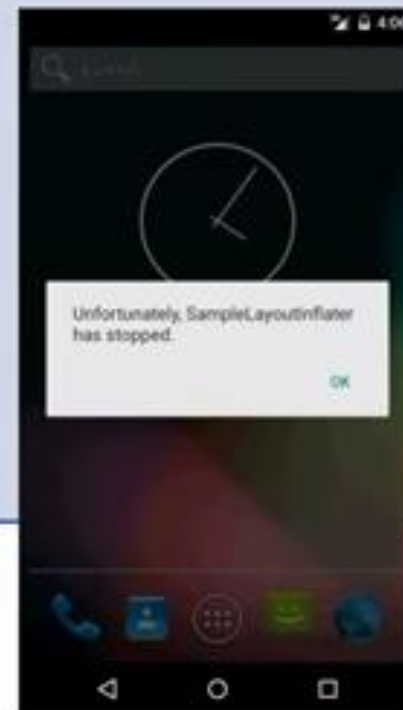
[시작 버튼의 레이아웃 인플레이션 과정]

1. 레이아웃 인플레이션

□ 레이아웃 인플레이션 이해- 호출순서

```
public class MainActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        Button button1 = (Button) findViewById(R.id.button1);  
        button1.setText( " 시작됨 " );  
  
        setContentView(R.layout.activity_main);  
    }  
}
```

[setContentView() 코드와 findViewById() 메소드의 호출 순서를 바꾼 경우]



1. 레이아웃 인플레이션

□ setContentView() 메소드 역할

- ▣ 화면에 나타낼 뷰를 지정하는 역할
- ▣ XML 레이아웃의 내용을 메모리상에 객체화하는 역할

[Reference]

```
public void setContentView(View view)
```

```
public void setContentView(int layoutResID)
```

```
public void setContentView(View view, ViewGroup.LayoutParams params)
```

1. 레이아웃 인플레이션

□ LayoutInflater 클래스

- ▣ 전체화면 중에서 일부분만을 화면구성요소들을 XML 레이아웃에서 로딩하여 보여주는 기능을 제공
- ▣ 시스템 서비스로 제공, LayoutInflater 객체를 참조 후 사용 가능

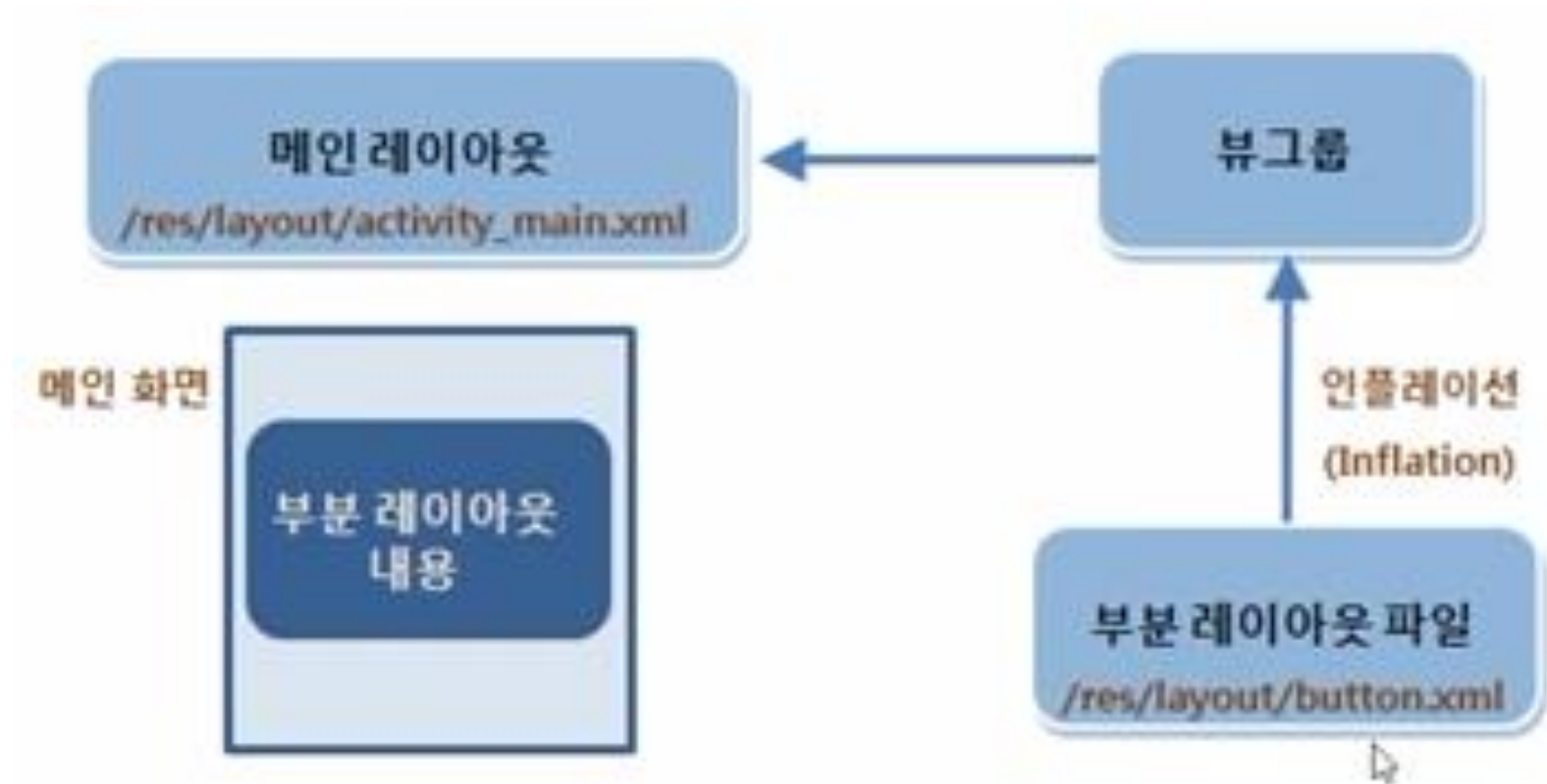
[Reference]

```
getSystemService(Context.LAYOUT_INFLATER_SERVICE)
```

- getSystemService : background 작업

1. 레이아웃 인플레이션

□ 레이아웃 인플레이션 개념도



화면 일부분을 XML 레이아웃 파일의 내용으로 적공하는 과정

1. 레이아웃 인플레이션

□ 화면의 전체와 일부

- ▣ 안드로이드에서 화면 : 소스와 화면 구성이 분리되어 있음
 - 자바 소스 1개
 - XML 레이아웃 1개
- ▣ 화면 전체 : 액티비티-> setContentView에서 인플레이션
 - 액티비티를 위한 자바 소스 1개 : MainActivity.java
 - 액티비티를 위한 XML 레이아웃 1개 : activity_main.xml
- ▣ 부분 화면-> 수동으로 인플레이션
 - 부분화면을 위한 자바 소스 1개 또는 뷰(뷰가 1개의 소스파일로 분리될 수 있음)
 - 부분화면을 위한 XML 레이아웃 1개 : singer.xml

1. 레이아웃 인플레이션

□ LayoutInflater객체의 inflate() 메소드

[Reference]

View inflate(int resource, ViewGroup root)

- resource : XML 레이아웃 리소스를 지정하는 값
- root : 뷰를 객체화하기 위해 추가할 대상이 되는 부모 컨테이너

□ getSystemService() 메소드를 대신하여 객체를 참조하는 메소드

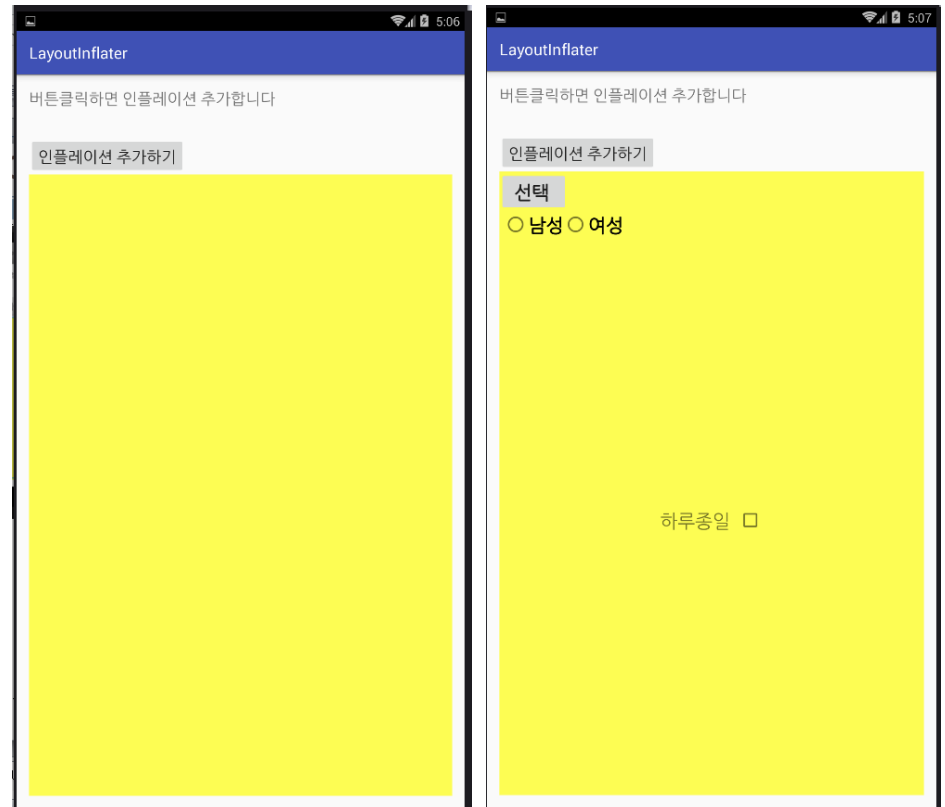
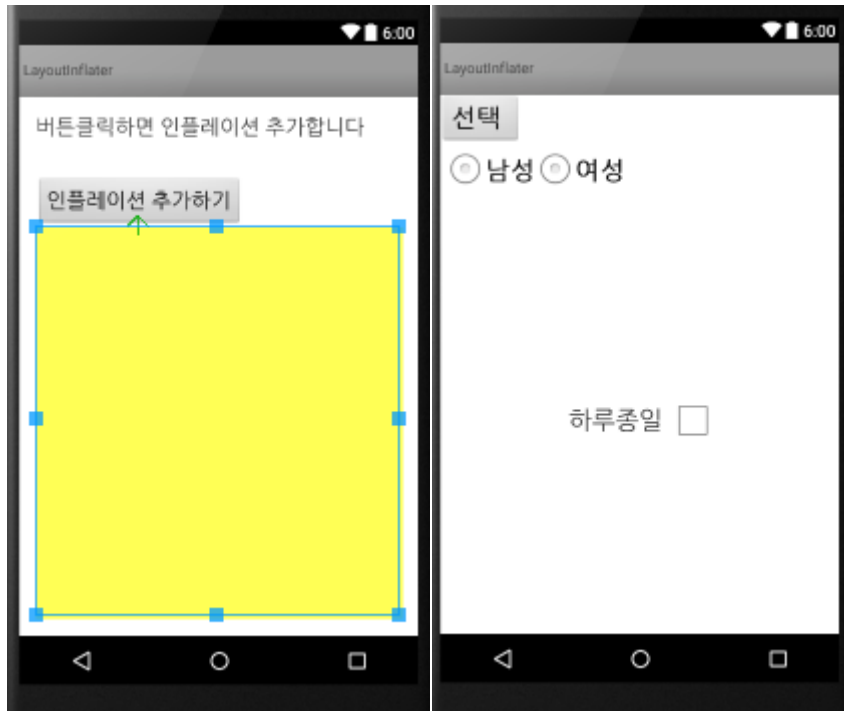
[Reference]

static LayoutInflater inflater.from(Context context)

[Reference]

View inflate(Context context, int resource, , ViewGroup root)

예제



코드 예

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClicked(View v){
        LinearLayout contentsLayout=(LinearLayout)findViewById(R.id.containerLayout);
        LayoutInflater inflater=
            (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        inflater.inflate(R.layout.button_layout, contentsLayout,true);
        Button selectButton=(Button)findViewById(R.id.selectButton);
        final CheckBox allDay=(CheckBox)findViewById(R.id.allDay);
        selectButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(allDay.isChecked()) allDay.setChecked(false);
                else allDay.setChecked(true);
            }
        });
    }
}
```

2. 화면 구성과 화면간 이동

□ 안드로이드 애플리케이션 4가지 구성요소



하나의 화면으로 구성

[안드로이드 애플리케이션을 구성하는 네 가지 구성요소]

2. 화면 구성과 화면간 이동

□ 액티비티(Activity)

- ▣ 화면을 구성하는 가장 기본적인 컴포넌트

□ 서비스(Service)

- ▣ 액티비티와 상관없이 백그라운드에서 동작하는 컴포넌트

□ 브로드캐스트 수신자(Broadcast Receiver)

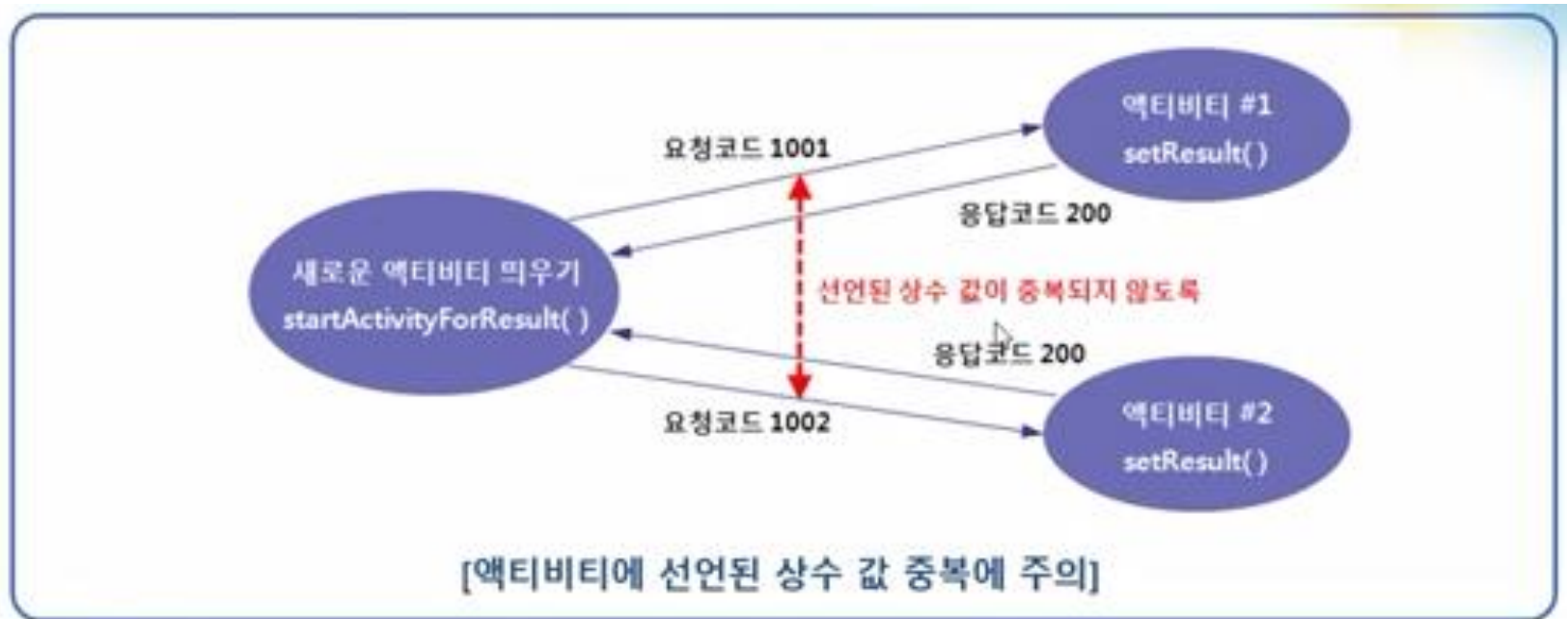
- ▣ 문자 메시지 도착, 배터리 방전, SD카드 탈부착, 네트워크 환경 변화 등이 발생하면 방송 신호 보냄

□ 내용제공자(Content Provider)

- ▣ 응용프로그램 사이에 데이터를 상호 공유하기 위한 컴포넌트

2. 화면 구성과 화면 간 이동

□ 새로운 액티비티 띄우기



2. 화면 구성과 화면 간 이동

□ 액티비티 띄우기

```
protected void startActivity(Intent intent)
```

- 단순히 새 액티비티를 띄우는 메소드

```
protected void startActivityForResult(Intent intent, int requestCode)
```

- 새 액티비티를 띄운 후 되돌아올 때 새 액티비티로부터 응답을 받을 경우

```
protected void onActivityResult(int requestCode, int resultCode, Intent intent)
```

- 띄웠던 액티비티가 응답으로 보내오면 응답을 처리하는 메소드

□ 안드로이드 프로젝트 생성

- ▣ 프로젝트 이름 : Project6_1
- ▣ 패키지 이름 : com.cookandroid.project10_1

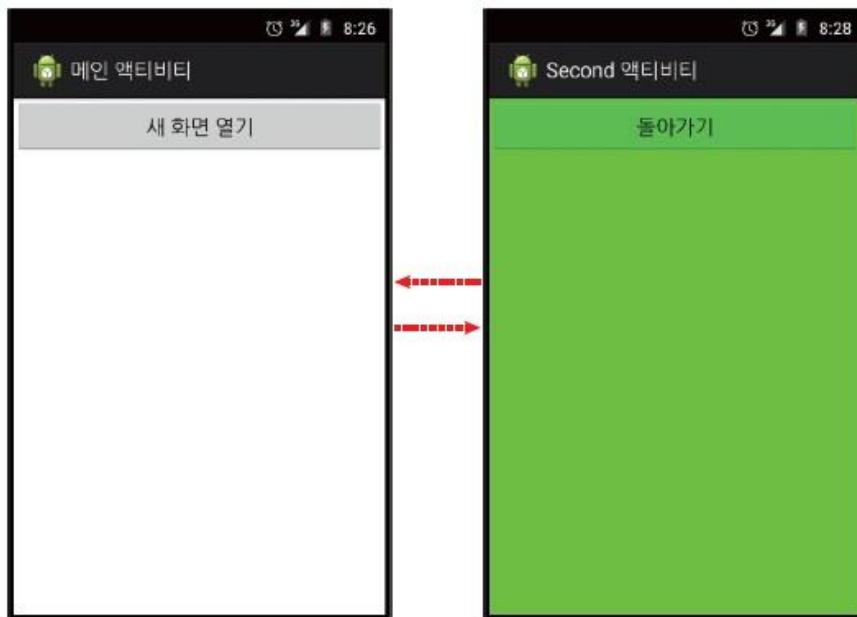


그림 10-2 서브 액티비티 실행 화면

□ 화면 디자인 및 편집

- activity_main.xml에 클릭하면 다른 액티비티가 나오게 하는 버튼을 하나 만듦

```
1  <LinearLayout>
2      <Button
3          android:id="@+id/btnNewActivity"
4          android:text="새 화면 열기" />
5  </LinearLayout>
```

□ 화면 디자인 및 편집

- 새로운 액티비티에서 사용할 second.xml을 만든 후 배경색을 바꾸고 <돌아가기> 버튼을 하나만 만들

```
1 <LinearLayout
2     android:background="#00FF00" >
3     <Button
4         android:id="@+id/btnReturn"
5         android:text="돌아가기" />
6 </LinearLayout>
```

□ Java 코드 작성 및 수정

- 새로운 액티비티인 SecondActivity.java 파일을 만들

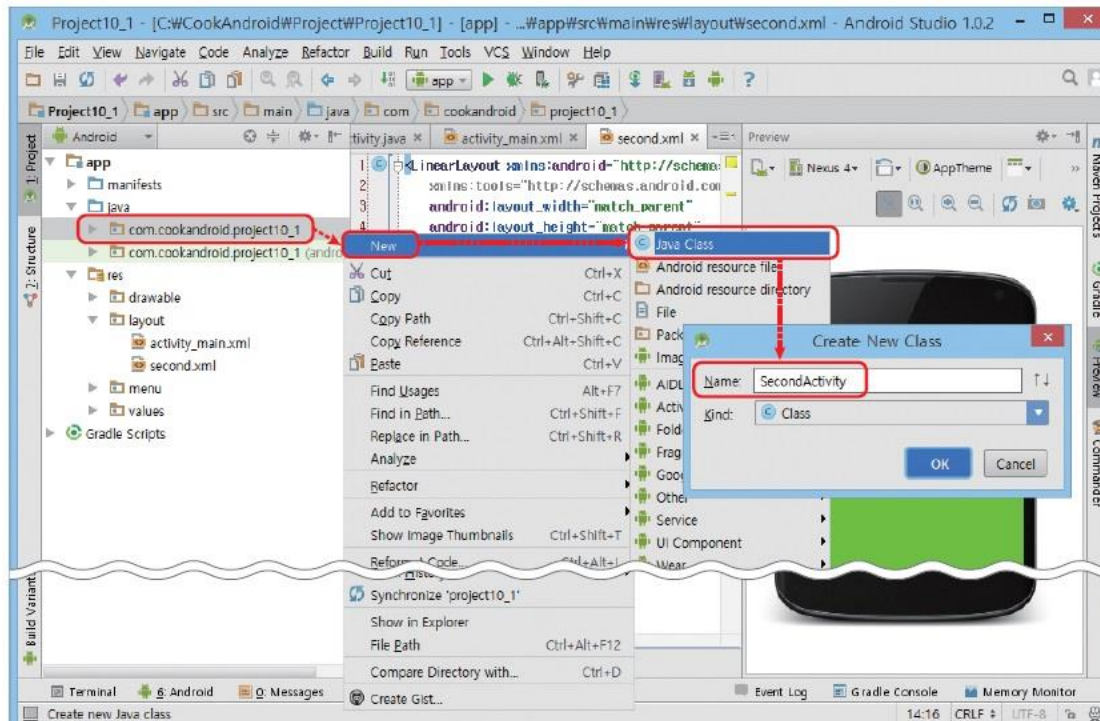


그림 10-3 새로운 클래스 생성 화면

□ Java 코드 작성 및 수정

- ▣ 액티비티의 필수 메소드인 onCreate()를 추가한 후 자동완성시킴
- ▣ second.xml을 화면에 보여주는 코드를 한 행 추가

```
1 public class SecondActivity extends Activity {  
2  
3     @Override  
4     protected void onCreate(Bundle savedInstanceState) {  
5  
6         super.onCreate(savedInstanceState);  
7         setContentView(R.layout.second);  
8  
9     }  
10 }
```

□ Java 코드 작성 및 수정

- second.xml의 <돌아가기>를 클릭하면 현재 액티비티를 끝내는 코드 추가

```
Button btnReturn = (Button) findViewById(R.id.btnReturn);  
btnReturn.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        finish();  
    }  
});
```

- 현재 액티비티를 종료
- SecondActivity는 메인 액티비티에서 호출할 것이므로 SecondActivity를 종료하면 다시 메인 액티비티가 나타남

□ Java 코드 작성 및 수정

- ▣ 메인 액티비티에서 SecondActivity를 호출하는 코드를 추가

```
Button btnNewActivity = (Button) findViewById(R.id.btnNewActivity);
btnNewActivity.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(),
                                   SecondActivity.class);
        startActivity(intent);
    }
});
```

□ AndroidManifest.xml에 SecorActioivity 등록

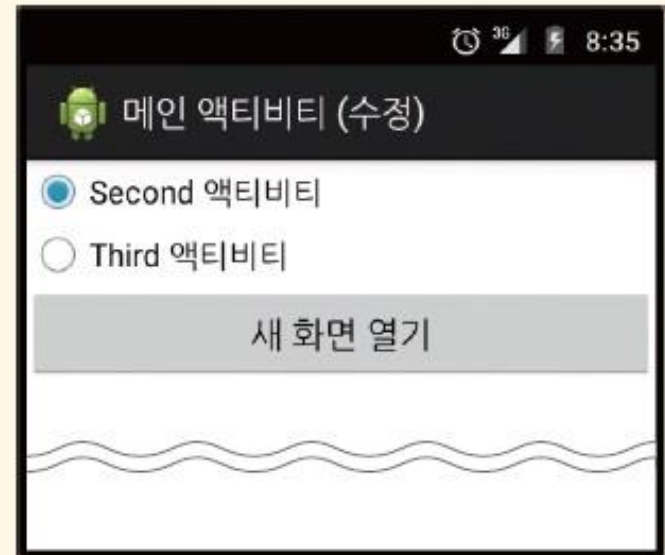
- 안드로이드에서는 사용될 액티비티를 AndroidManifest.xml에 꼭 등록해야 함
- 메인 액티비티(MainActivity)는 자동으로 등록되지만, 추가한 SecondActivity는 별도로 등록해줘야 함
- 아래 코드를 `</application>` 바로 윗 행에 코딩

```
<activity android:name=".SecondActivity" android:label="Second 액티비티"/>
```

▶ **직접 풀어보기 10-1**

[실습 10-1]을 다음과 같이 수정한다.

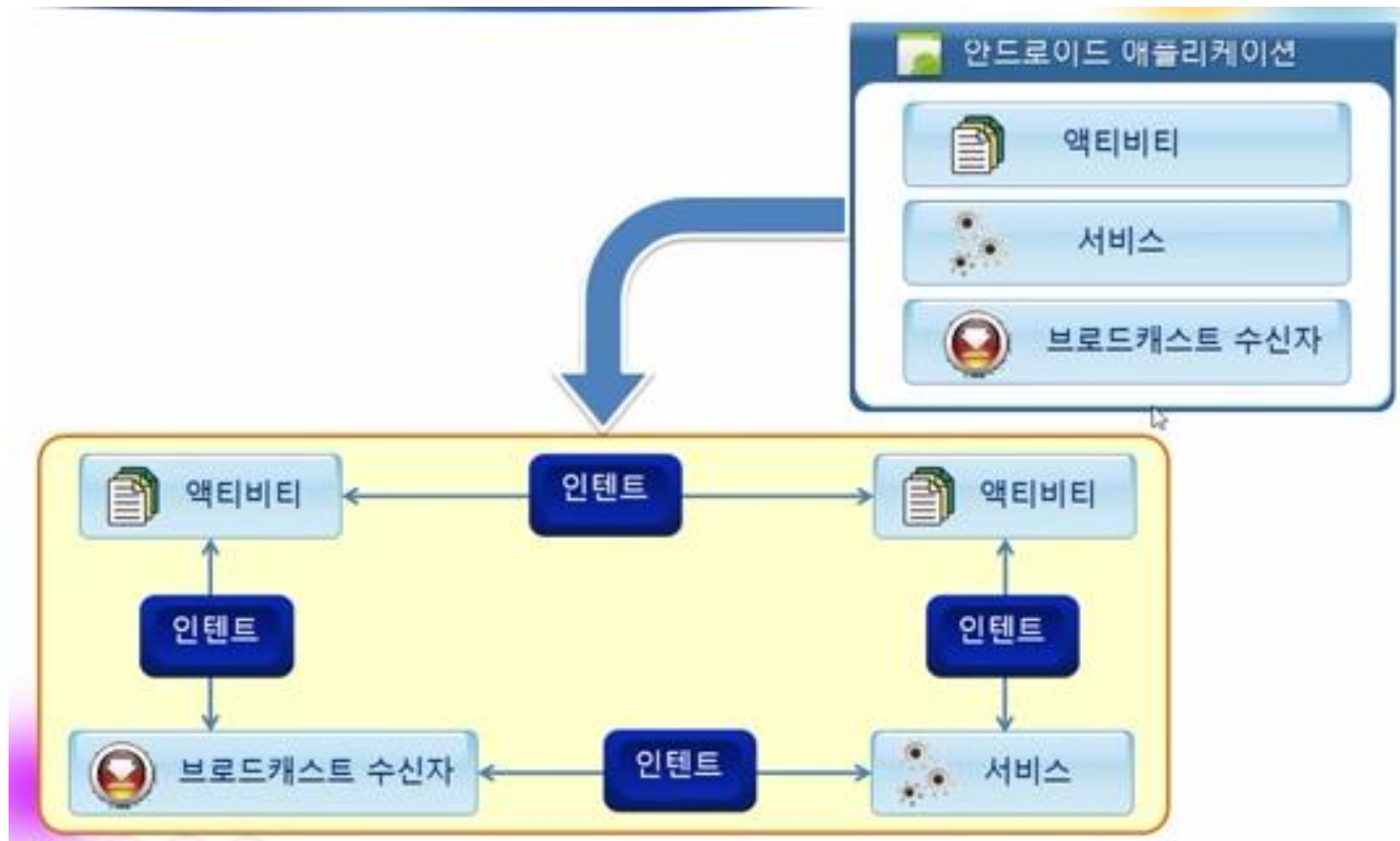
- ThirdActivity를 추가한다.
- 라디오버튼으로 선택된 액티비티가 나오게 한다.



2. 인텐트와 데이터 전달

□ 인텐트(Intent)

- 안드로이드 4대 컴포넌트가 상호 간에 데이터를 주고 받기 위한 메시지 객체



2. 인텐트와 데이터 전달

□ 인텐트의 종류

- ▣ 명시적 인텐트와 암시적 인텐트로 구분

□ 명시적 인텐트와 데이터의 전달

- ▣ 명시적 인텐트 : 다른 액티비티의 이름을 명확히 지정할 때 사용하는 방법

```
Intent intent = new Intent(getApplicationContext(), SecondActivity.class);  
startActivity(intent);
```

□ 명시적 인텐트와 데이터의 전달

- 메인 액티비티에서 인텐트에 데이터를 실어서 넘긴 후, 세컨드 액티비티에서 받은 데이터를 처리

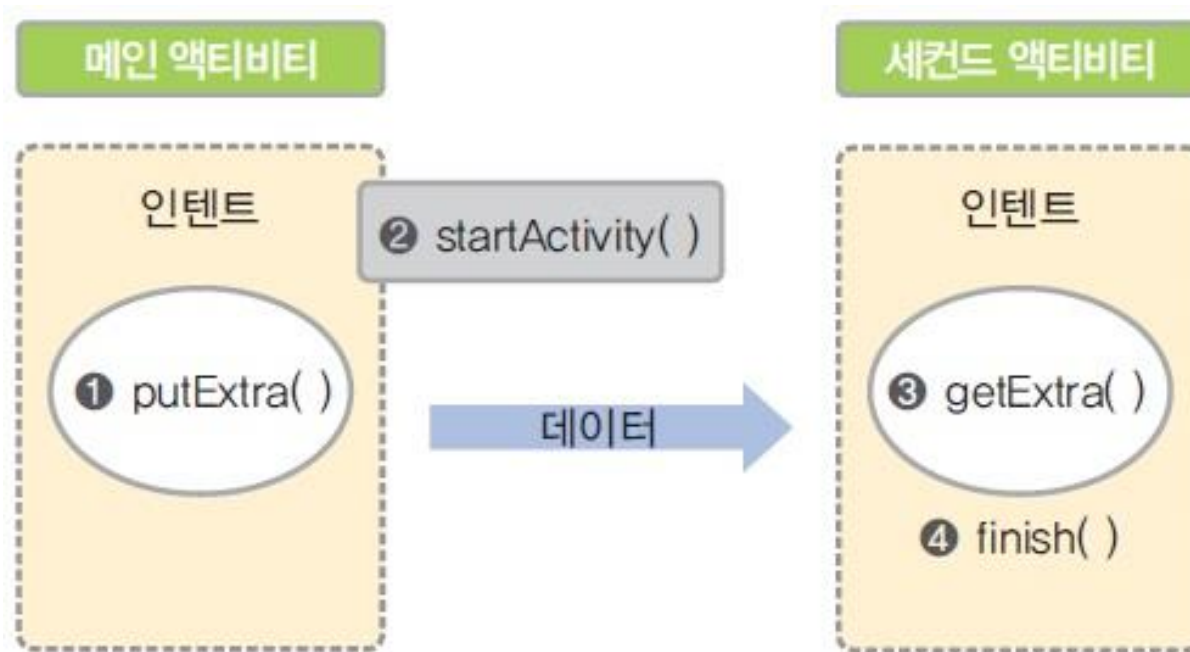


그림 10-5 한쪽 방향으로 데이터를 전달하는 방법

□ 양방향 액티비티와 데이터의 전달

- 메인 액티비티에서 세컨드 액티비티로 데이터를 넘긴 후에 다시 세컨드 액티비티에서 메인 액티비티로 데이터를 돌려주는 경우

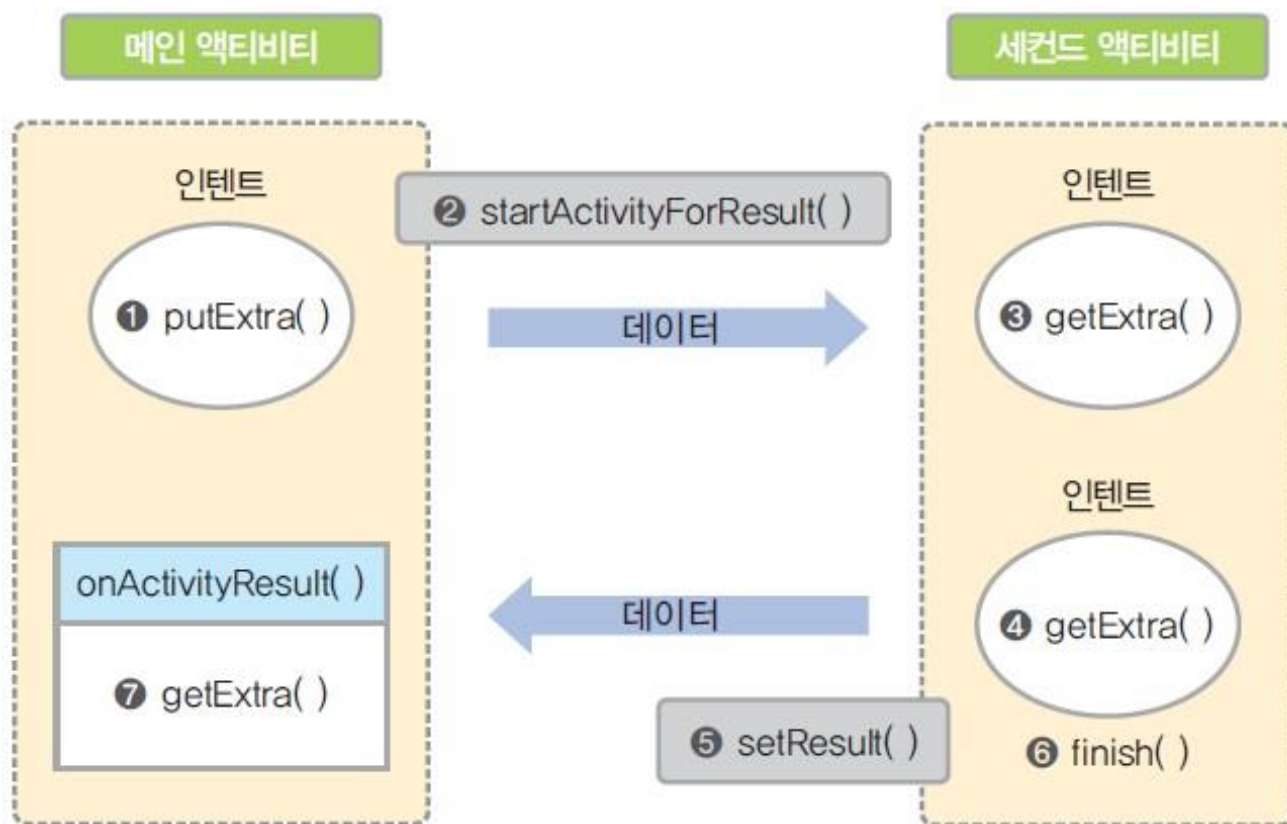


그림 10-7 양방향으로 데이터를 전달하는 방법

양방향 데이터 전달 예제

- 메인 액티비티의 에디트텍스트의 두 수를 세컨드 액티비티에서 더한 후에 다시 메인 액티비티로 돌려줌

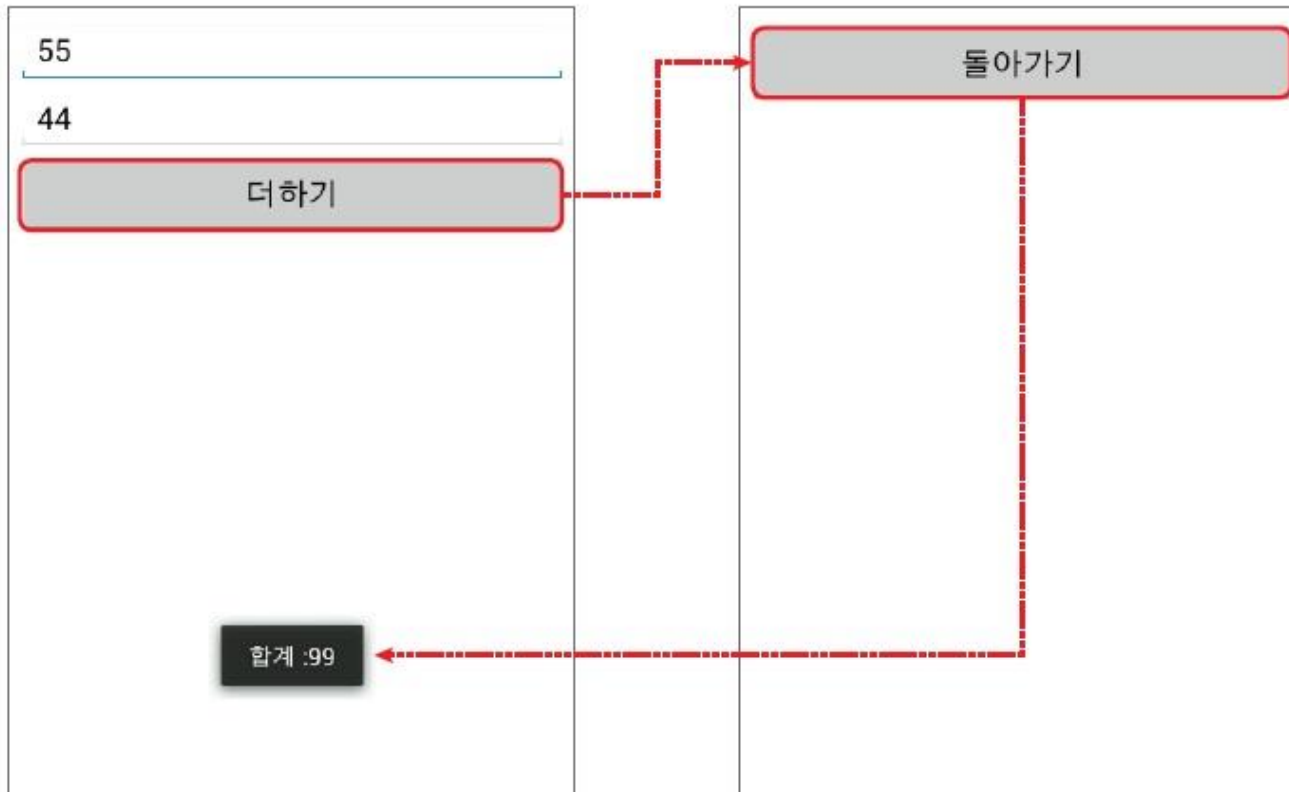


그림 10-8 양방향 데이터 전달 예제 결과 화면

□ 양방향 데이터 전달 예제

```
<LinearLayout>
    <EditText
        android:id="@+id/edtNum1" />
    <EditText
        android:id="@+id/edtNum2" />
    <Button
        android:id="@+id/btnNewActivity"
        android:text="더하기" />
</LinearLayout>
```

```
<LinearLayout>
    <Button
        android:id="@+id/btnReturn"
        android:text="돌아가기" />
</LinearLayout>
```

양방향 데이터 전달 예제- MainActivity

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    setTitle("메인 액티비티");  
  
    Button btnNewActivity = (Button) findViewById(R.id.btnNewActivity);  
    btnNewActivity.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) {  
            EditText edtNum1 = (EditText) findViewById(R.id.edtNum1);  
            EditText edtNum2 = (EditText) findViewById(R.id.edtNum2);  
            Intent intent = new Intent(getApplicationContext(), SecondActivity.class);  
            intent.putExtra("Num1", Integer.parseInt(edtNum1.getText().toString()));  
            intent.putExtra("Num2", Integer.parseInt(edtNum2.getText().toString()));  
            startActivityForResult(intent, 0);  
        }  
    });  
}
```

양방향 데이터 전달 예제- MainActivity

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    if(resultCode == RESULT_OK) {
        int hap = data.getIntExtra("Hap", 0);
        Toast.makeText(getApplicationContext(),
                        "합계 :" + hap, Toast.LENGTH_SHORT).show();
    }
}
```


양방향 데이터 전달 예제- SecondActivity

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.second);  
    setTitle("Second 액티비티");
```

```
    Intent inIntent = getIntent();  
    final int hapValue = inIntent.getIntExtra("Num1", 0)  
        + inIntent.getIntExtra("Num2", 0 );
```

```
    Button btnReturn = (Button) findViewById(R.id.btnReturn);  
    btnReturn.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) {
```

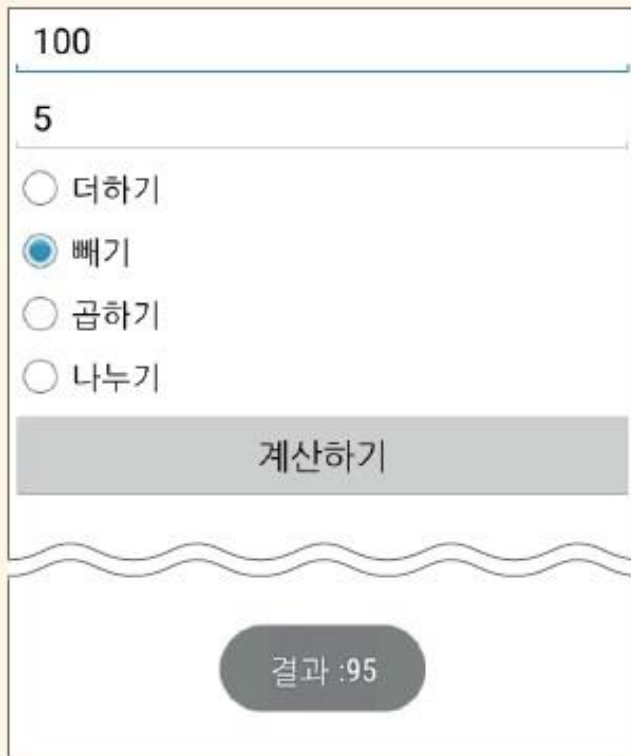
```
            Intent outIntent = new Intent(getApplicationContext(),  
                MainActivity.class);  
            outIntent.putExtra("Hap", hapValue);  
            setResult(RESULT_OK, outIntent);  
            finish();
```

```
        }  
    });
```

```
}
```

▶ **직접 풀어보기 10-3**

두 수를 입력하고 더하기/빼기/곱하기/나누기 라디오 버튼을 선택한 후 <계산하기>를 클릭하면 세컨드 액티비티에서 계산 후 값을 돌려받는다.



100

5

☐ 더하기

☒ 빼기

☐ 곱하기

☐ 나누기

계산하기

결과 :95

□ 암시적 인텐트(Implicit Intent, 묵시적 인텐트)

- 약속된 액션(Action)을 지정하여 안드로이드에서 제공하는 기존 응용프로그램을 실행하는 것
- 전화 거는 것을 예로 들면 전화번호를 인텐트로 넘긴 후에 전화 걸기 응용프로그램이 실행되는 것과 같음

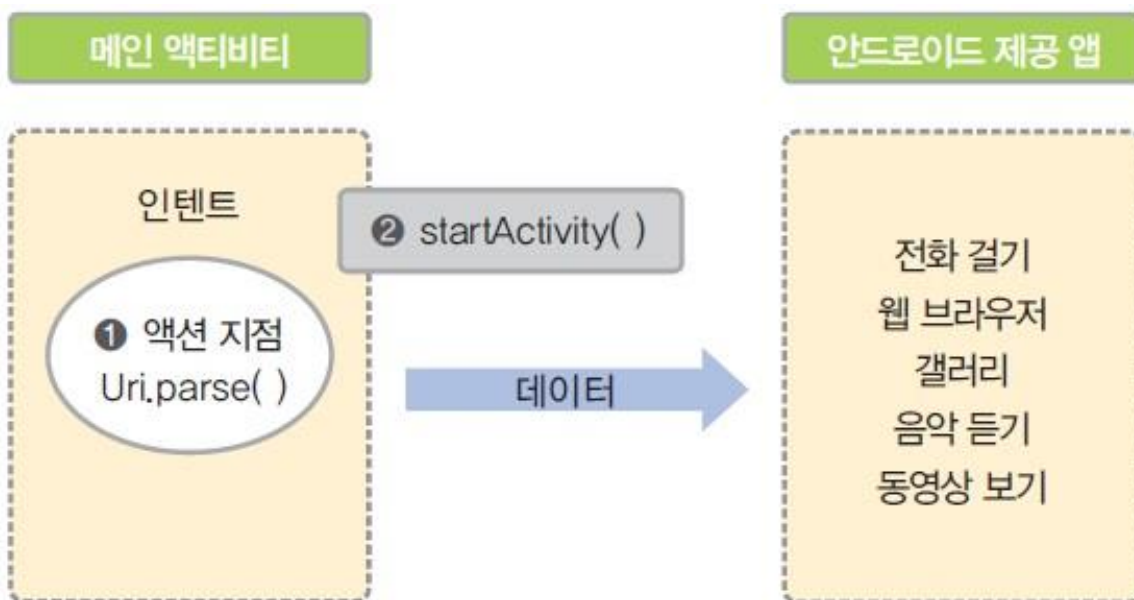


그림 10-9 암시적 인텐트

- **암시적 인텐트(Implicit Intent, 묵시적 인텐트) 예제**
 - ▣ 전화를 거는 형식(교재 : p212-214) 참고
- **인텐트(Intent) 클래스에 정의된 액션 정보**
 - ▣ P215 참고

암시적 인텐트 예제

```
1 <LinearLayout>
2     <Button
3         android:id="@+id/btnDial"
4         android:text="전화 걸기" />
5     <Button
6         android:id="@+id/btnWeb"
7         android:text="홈 페이지 열기" />
8     <Button
9         android:id="@+id/btnGoogle"
10        android:text="구글 맵 열기" />
11    <Button
12        android:id="@+id/btnSearch"
13        android:text="구글 검색하기" />
14    <Button
15        android:id="@+id/btnSms"
16        android:text="문자 보내기" />
17    <Button
18        android:id="@+id/btnPhoto"
19        android:text="사진 보기" />
20 </LinearLayout>
```

전화 걸기
홈 페이지 열기
구글 맵 열기
구글 검색하기
문자 보내기
사진 보기

암시적 인텐트 예제

```
public class MainActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        setTitle("암시적 인텐트 예제");  
  
        Button btnDial = (Button) findViewById(R.id.btnDial);  
        Button btnWeb = (Button) findViewById(R.id.btnWeb);  
        Button btnGoogle = (Button) findViewById(R.id.btnGoogle);  
        Button btnSearch = (Button) findViewById(R.id.btnSearch);  
        Button btnSms = (Button) findViewById(R.id.btnSms);  
        Button btnPhoto = (Button) findViewById(R.id.btnPhoto);  
    }  
}
```

암시적 인텐트 예제

```
btnDial.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        Uri uri = Uri.parse("tel:010-1234-5678");  
        Intent intent = new Intent(Intent.ACTION_DIAL, uri);  
        startActivity(intent);  
    }  
});
```

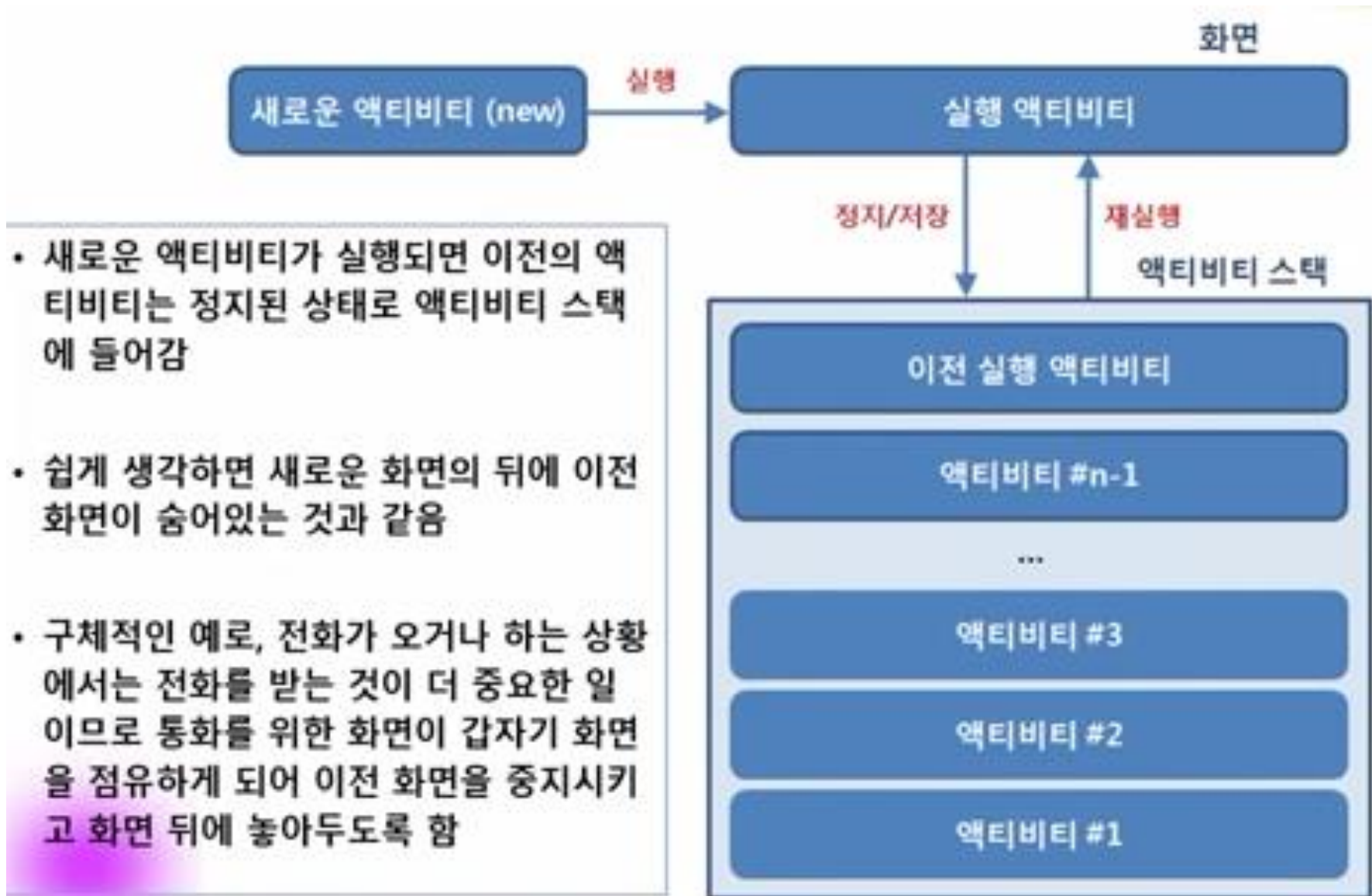
```
btnWeb.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        Uri uri = Uri.parse("http://www.hanb.co.kr");  
        Intent intent = new Intent(Intent.ACTION_VIEW, uri);  
        startActivity(intent);  
    }  
});
```

```
//... 생략
```

```
}
```

```
}
```

액티비티 스택과 플래그



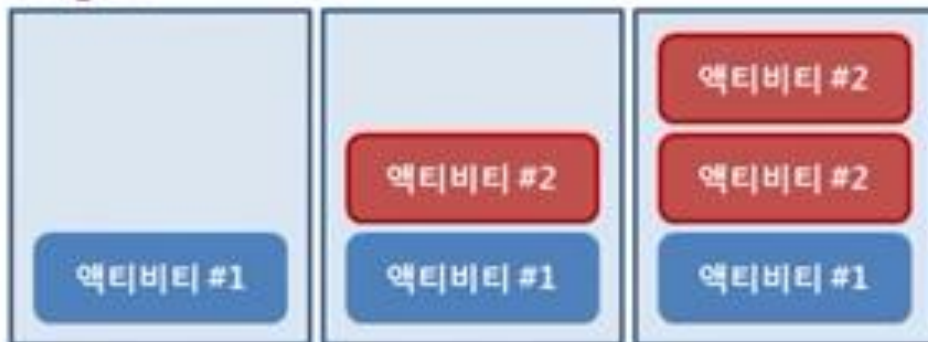
액티비티 스택과 플래그 사용

[Reference]

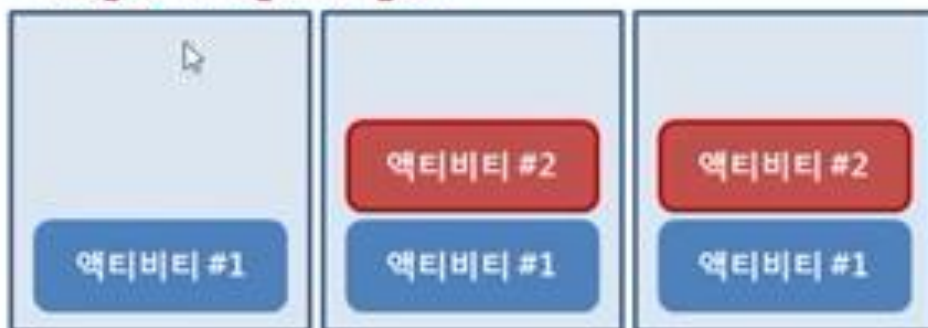
FLAG_ACTIVITY_SINGLE_TOP
FLAG_ACTIVITY_NO_HISTORY
FLAG_ACTIVITY_CLEAR_TOP

- 새로운 액티비티를 실행할 때마다 메모리에 새로운 객체를 만들고 이전 화면 위에 쌓는 방식은 비효율적일 수 있음
- 동일한 화면이 이미 만들어져 있는 경우에는 그 화면을 그대로 보여주고 싶다면 플래그를 사용하면 됨

NO_FLAG



FLAG_ACTIVITY_SINGLE_TOP



[FLAG_ACTIVITY_SINGLE_TOP 플래그를 사용한 경우]

액티비티 스택과 플래그

□ 액티비티 플레그 사용 예

```
Intent intent = new Intent(getBaseContext(), AnotherActivity.class );  
intent.putExtra("startCount", String.valueOf(startCount));  
intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);  
startActivityForResult(intent, REQUEST_CODE_ANOTHER);
```

1 인텐트 객체 생성

2 부가 데이터 넣기

3 인텐트 플래그 설정

4 인텐트 띄우기

액티비티 스택과 플래그

□ 다른 액티비티 플래그 사용



[FLAG_ACTIVITY_NO_HISTORY 플래그를 사용한 경우]



[FLAG_ACTIVITY_CLEAR_TOP 플래그를 사용한 경우]

액티비티 스택과 플래그

□ 부가 데이터 전달하기

[Reference]

```
Intent.putExtra(String name, String value)
Intent.putExtra(String name, int value)
Intent.putExtra(String name, boolean value)

String getStringExtra(String name)
int getIntExtra(String name, int defaultValue)
boolean getBooleanExtra(String name, boolean defaultValue)
```

[Reference]

```
public abstract int describeContents()
public abstract void writeToParcel(Parcel dest, int flags)
```

- 화면과 화면 간에 데이터를 전달하고 싶다면 인텐트의 부가 데이터(Extra)로 넣어 전달하는 방법을 사용함
- 인텐트는 애플리케이션 구성 요소 간에 데이터를 전달하는 방법을 제공하는 것이므로 화면과 화면 간 뿐만 아니라 화면과 서비스 간, 또는 브로드캐스트 수신자와 화면 간 등등 애플리케이션 구성 요소 간에 부가 데이터로 넣어 데이터를 전달할 수 있음

4. 수명 주기

□ 액티비티 생명주기

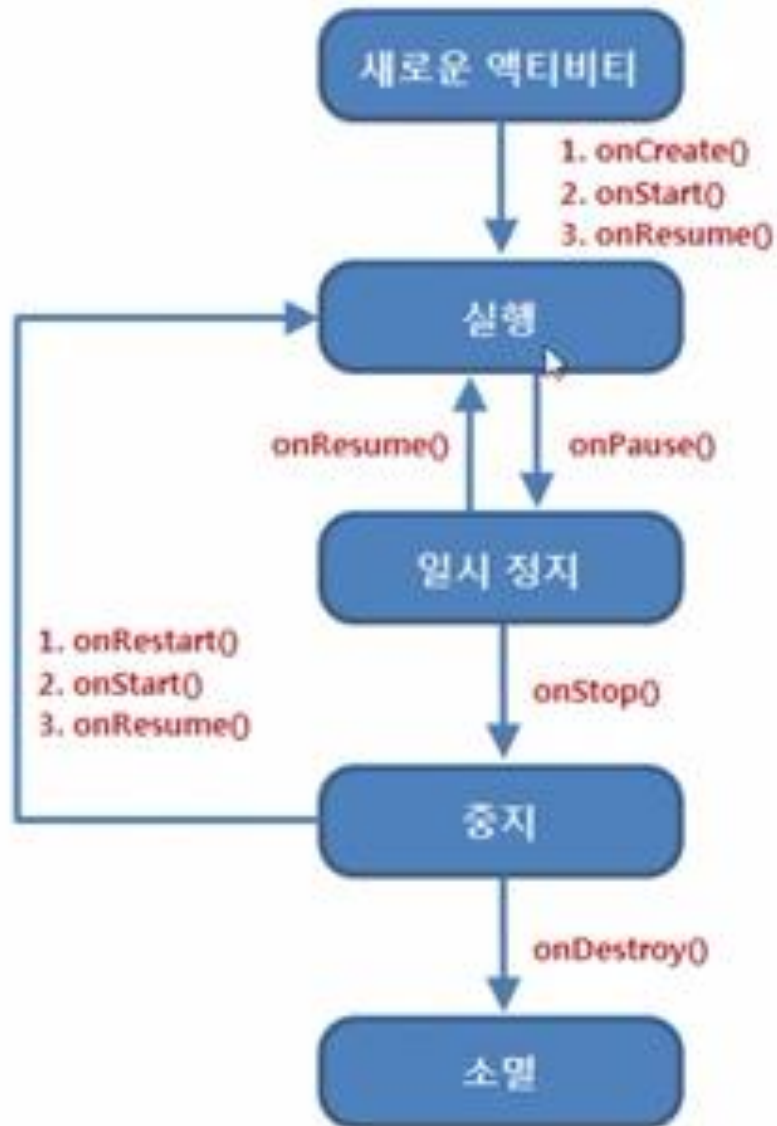
- ▣ 액티비티의 생성부터 소멸까지의 주기를 뜻함
- ▣ 안드로이드 응용프로그램은 화면이 작아 동시에 여러 개의 액티비티(화면)가 나올 수 없음
- ▣ 앞에 나오는 화면 하나만 활성화된 상태이고 나머지는 모두 비활성화된 상태로 남게 됨

4. 수명주기

□ 액티비티의 대표적인 상태 정보

상 태	설 명
실행(Running)	화면 상에 액티비티가 보이면서 실행되어 있는 상태. 액티비티 스택의 최상위에 있으며 포커스를 가지고 있음
일시 중지(Paused)	사용자에게 보이기는 하지만 다른 액티비티가 위에 있어 포커스를 받지 못하는 상태. 대화상자가 위에 있어 일부가 가려져 있는 경우에 해당함
중지(Stopped)	다른 액티비티에 의해 완전히 가려져 보이지 않는 상태

4. 수명 주기



함수 상세 설명
교재 : P232

로그캣 사용

저자
한마디

로그캣(LogCat)

작성 중인 프로그램에 예기치 못한 오류가 발생했을 때 원인을 파악하는 방법 중 하나가 로그(Log)를 남기는 것이다. 안드로이드는 `android.util.Log` 클래스를 제공해서 로그를 남기고, 로그캣(LogCat)이라는 화면을 제공해서 로그를 확인한다. 프로그래머가 로그를 남기기 위해 사용하는 메소드는 다음 표와 같다. 하지만 절대적인 기준은 아니며, 프로그래머가 적절한 메소드를 골라 사용해야 한다.

메소드	설명
<code>android.util.Log.d("태그", "메시지")</code>	Debugging : 디버깅 용도로 남기는 로그
<code>android.util.Log.e("태그", "메시지")</code>	Error : 가장 심각한 오류 발생시 남기는 로그
<code>android.util.Log.i("태그", "메시지")</code>	Information : 정보를 남기기 위한 로그
<code>android.util.Log.v("태그", "메시지")</code>	Verbose : 상세한 기록을 남기기 위한 로그
<code>android.util.Log.w("태그", "메시지")</code>	Warning : 경고 수준을 남기기 위한 로그

생명주기 예제

