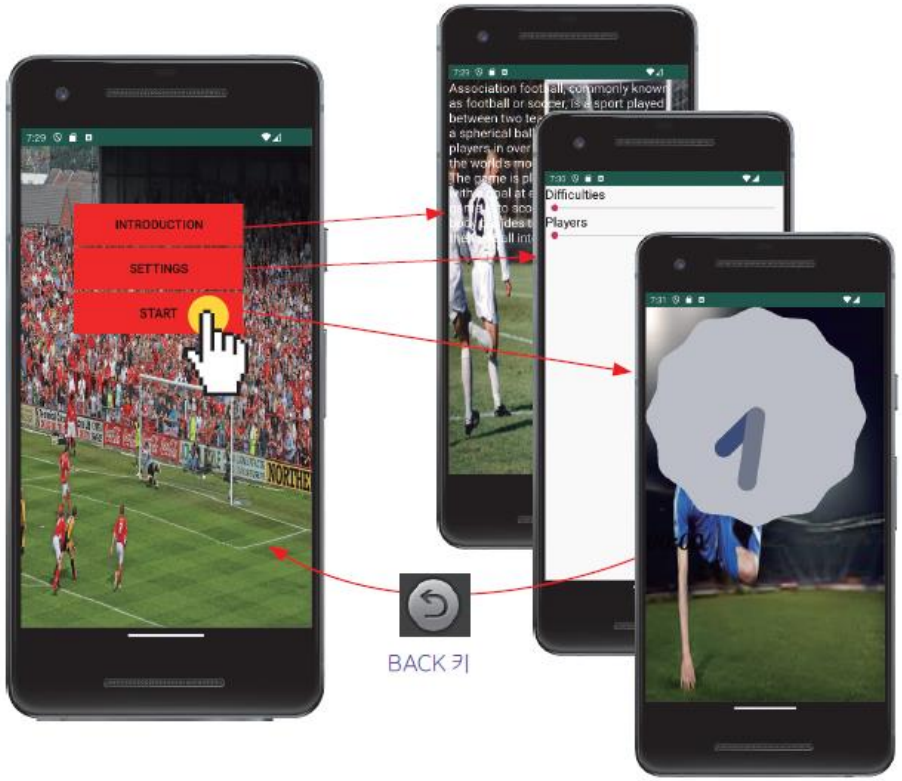
CHAP 6. 액티비티와 인텐트

액티비티와 인텐트


도대체 액티비티가 뭐예요?
개념이 머릿속에서 맴맴 돌 뿐
잡히지 않아서요?
그래서 이 늦은 시간에 전화? 하하하!
액티비티는 앱을 짓는 데 구성하는 빌딩
블록이라고 보면 되지. 따라서 애플리케
이션을 작성한다는 것은 결국 액티비티를
하나씩 작성한 후에 하나로 조립하는 것
이라고 할 수 있어.

6장의 목표

- 멀티페이지 앱 만들기

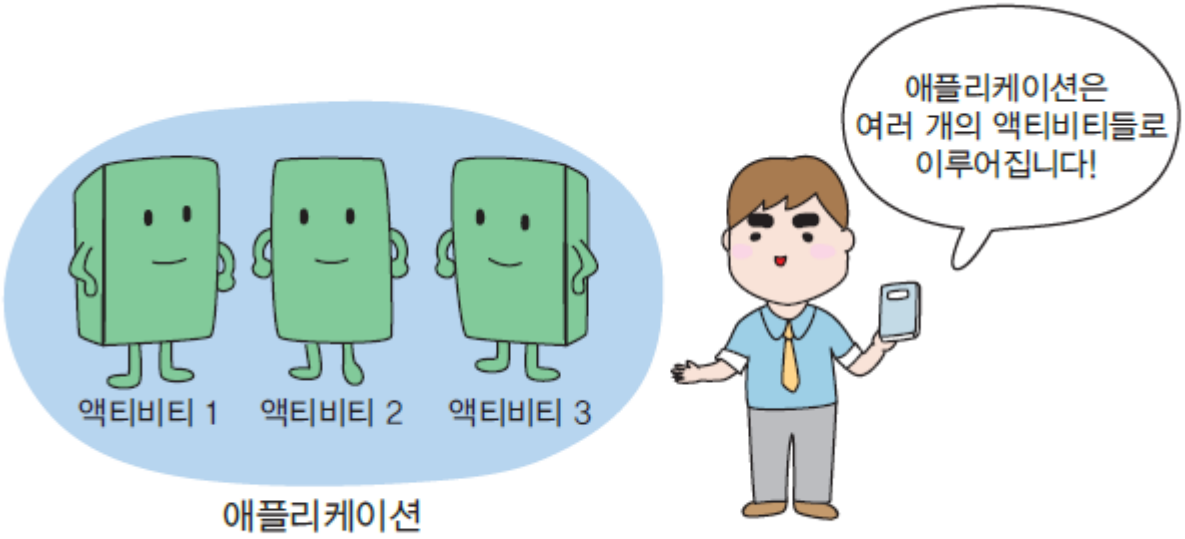


4가지의 중요한 개념

- 애플리케이션(application)
- 액티비티(activities)
- 액티비티 스택(activity stack)
- 태스크(task)

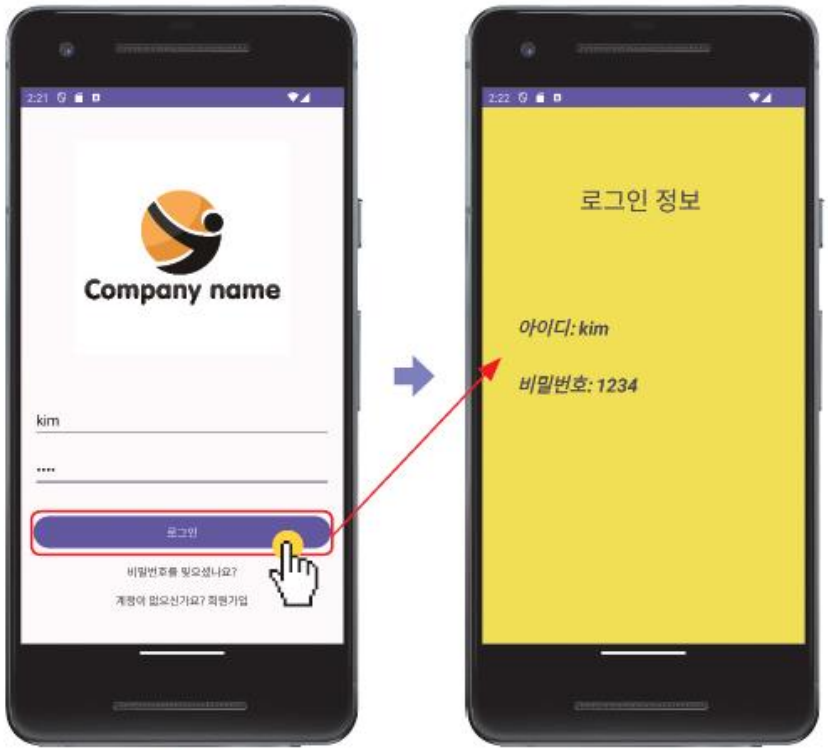
애플리케이션

- 한 개 이상의 액티비티들로 구성된다.
- 액티비티들은 애플리케이션 안에서 느슨하게 묶여 있다.



액티비티

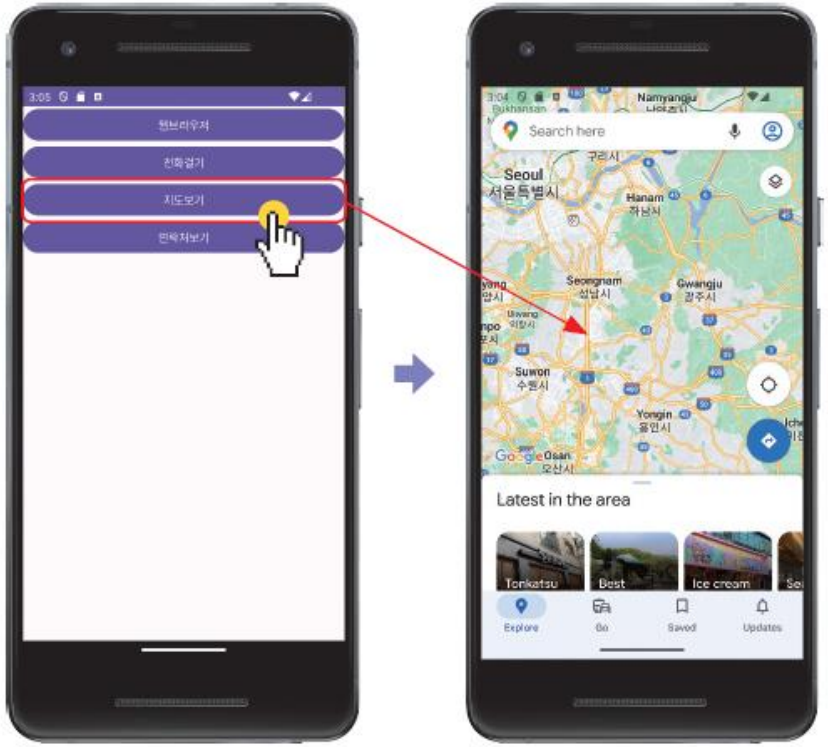
- 애플리케이션을 구성하는 빌딩 블록



첫 번째 액티비티 두 번째 액티비티

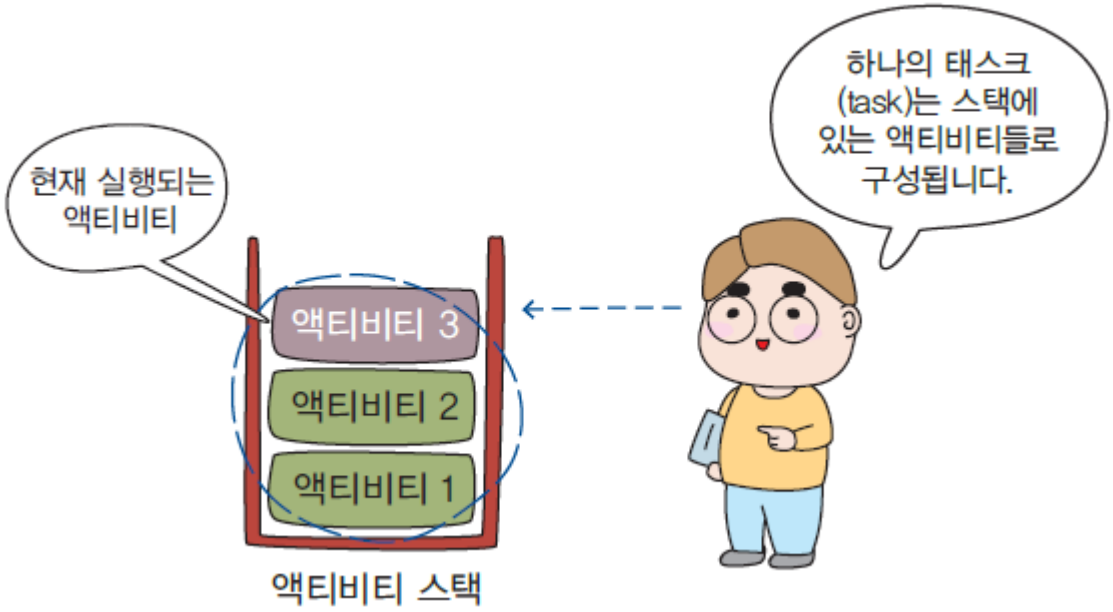
액티비티

- 안드로이드에서는 심지어 다른 애플리케이션의 액티비티도 시작할 수 있다.



태스크

- 태스크 = 스택에 있는 액티비티들

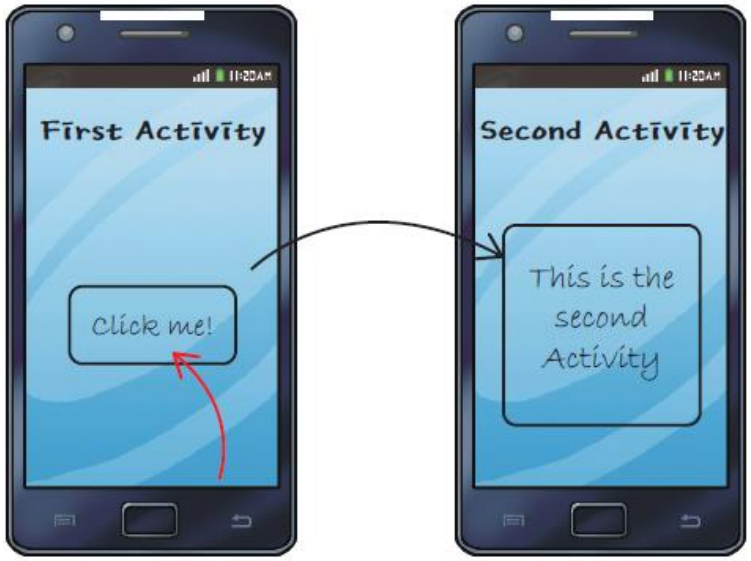


액티비티 스택

- Back 키를 누르면 현재 액티비티를 제거하고 이전 액티비티로 되돌아 간다.
- 사용자가 방문한 액티비티들은 어딘가에 기억

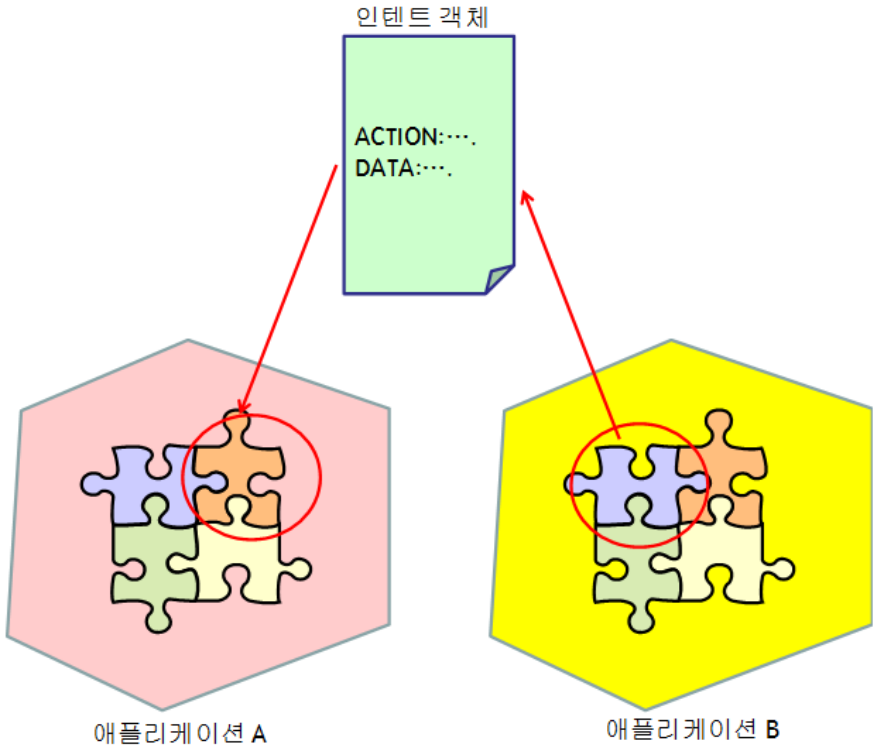
인텐트

- 각각의 화면은 별도의 액티비티로 구현된다.
- 하나의 액티비티(화면)에서 다른 액티비티(화면)로 전환하려면 어떻게 하여야 하는가?



인텐트

- 다른 액티비티를 시작하려면 액티비티의 실행에 필요한 여러 가지 정보들을 보내주어야 한다.
- 정보를 인텐트에 실어서 보낸다.



인텐트의 종류

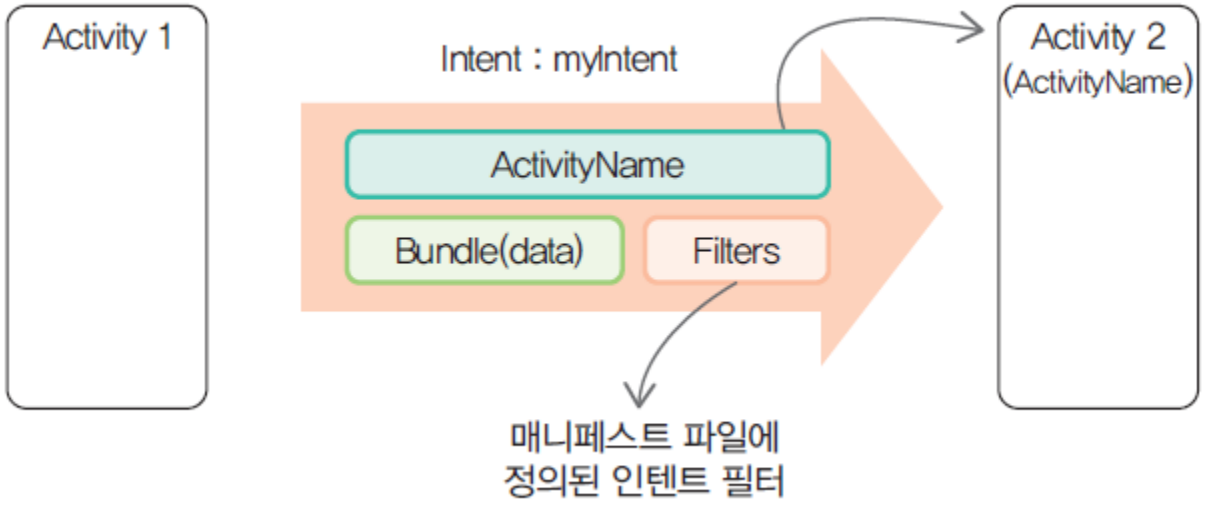
- **명시적 인텐트(explicit intent)**
  - “애플리케이션 A의 컴포넌트 B를 구동시켜라“와 같이 명확하게 지정

- **암시적 인텐트(implicit intent)**
  - “지도를 보여줄 수 있는 컴포넌트이면 어떤 것이라도 좋다”

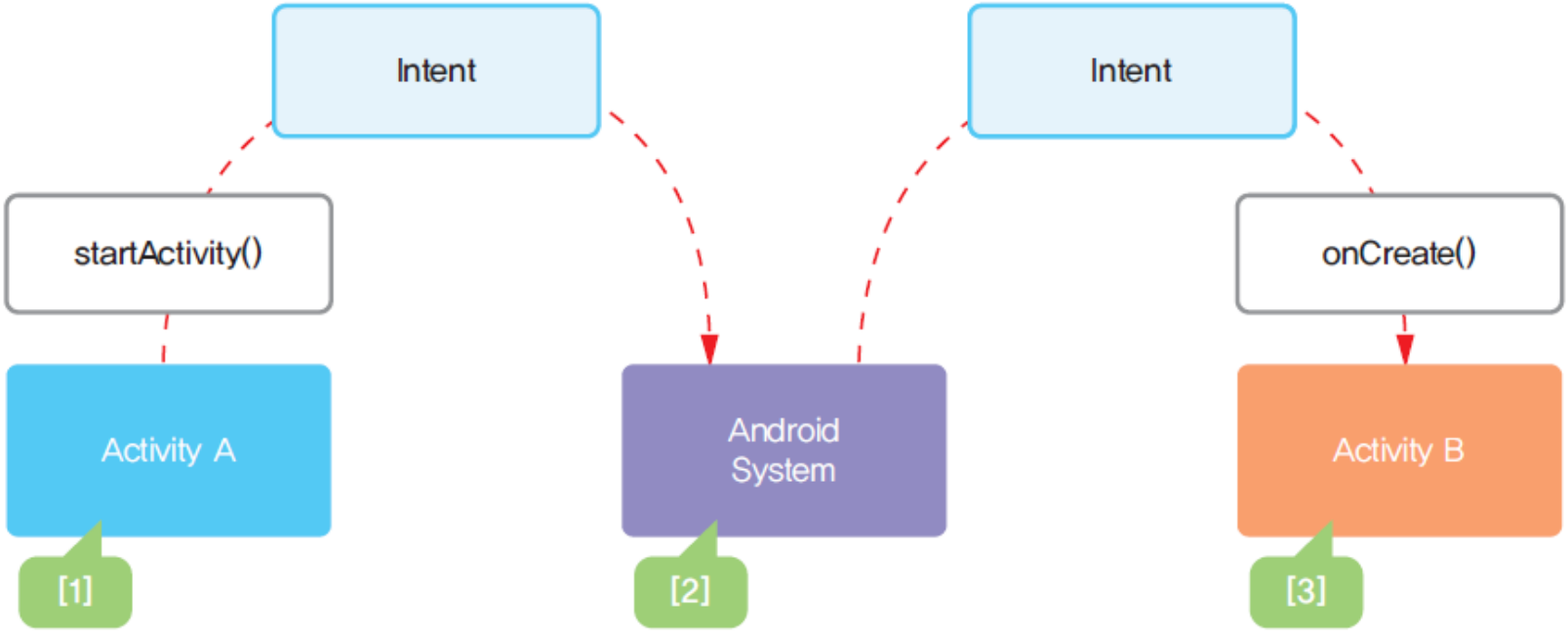
명시적인 인텐트

- 실행하고자 하는 액티비티의 이름을 적어 준다.

```
Intent intent = new Intent(this,  NextActivity.class);
startActivity(intent);
```



암시적인 인텐트



[1] Activity A가 원하는 액션을 기술한 인텐트를 생성하여서 startActivity()에 넘긴다.

[2] 안드로이드 시스템이 모든 애플리케이션을 대상으로 인텐트 필터를 검색한다. 일치하는 애플리케이션이 발견되면

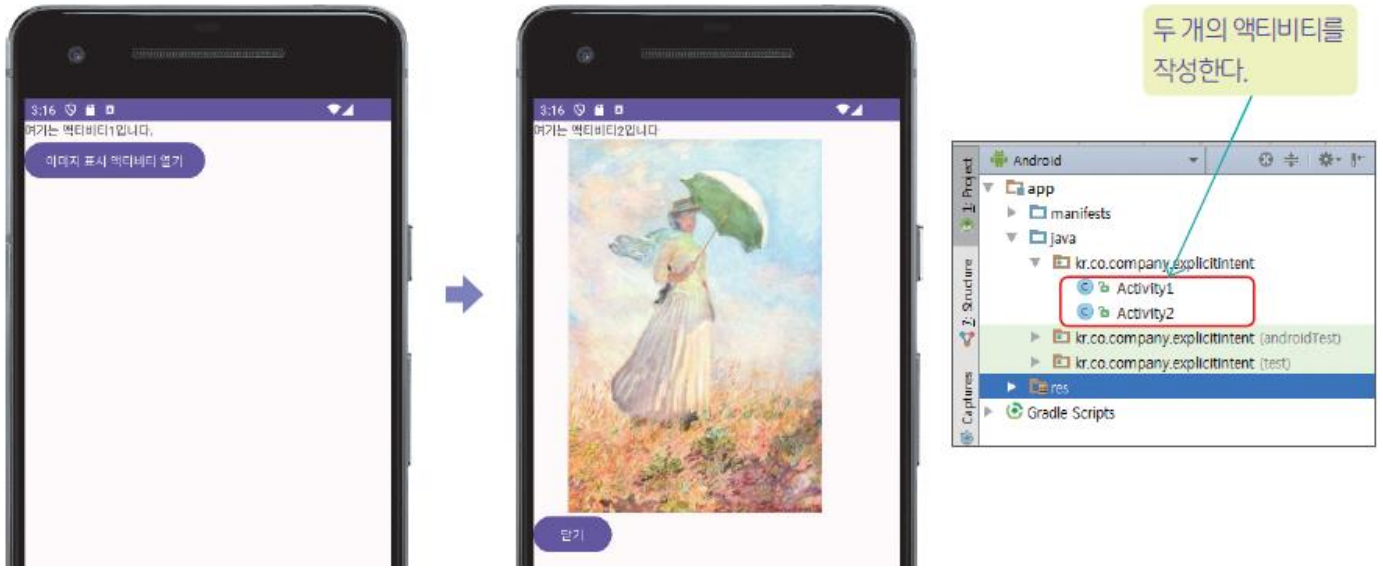
[3] 시스템은 일치하는 액티비티(액티비티 B)를 시작한다. onCreate() 메소드를 호출하고 인텐트를 인수로 넘긴다.

명시적인 인텐트 구조

- 우리는 명시적 인텐트를 사용하여서 하나의 화면에서 다른 화면으로 넘어갈 수 있다. 예를 들어서 현재 액티비티에서 NextActivity라는 이름의 새로운 액티비티를 시작하려면 다음과 같은 문장을 사용한다.

예제: 명시적인 인텐트

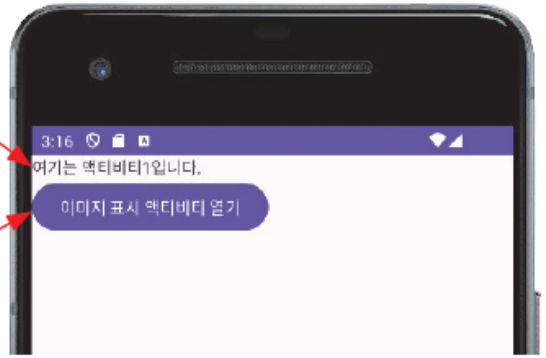
- 여기서 두 개의 액티비티로 이루어진 애플리케이션을 작성하여 보자. 첫 번째 액티비티는 Activity1, 두 번째 액티비티는 Activity2라고 하자.



레이아웃 파일 layout1.xml

layout1.xml

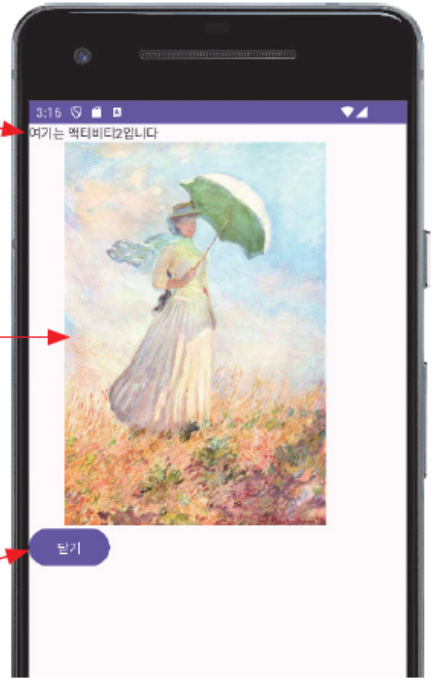
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="여기는 액티비티1입니다." />

    <Button
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="이미지 표시 액티비티 열기" />
</LinearLayout>
```



레이아웃 파일 layout2.xml

layout2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="여기는 액티비티2입니다" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="363dp"
        android:layout_height="418dp"
        android:src="@drawable/pic" />

    <Button
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="닫기" />
</LinearLayout>
```



Activity1.java

Activity1.java

```
package kr.co.company.explicitintent;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class Activity1 extends AppCompatActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout1);
        Button b = (Button)findViewById(R.id.Button01);
        b.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(Activity1.this,
                        Activity2.class);
            startActivity(intent);
        }
      });
  }
}
```

버튼이 클릭되면 Activity2를 시작한다.

두 번째 액티비티를 시작하려면 어떻게 하여야 하는가? 먼저 인텐트 객체를 생성한다. 우리는 두 번째 액티비티의 이름을 알고 있으므로 두 번째 액티비티의 클래스 이름을 인수로 주어서 인텐트 객체를 생성하면 된다. 즉 명시적인 인텐트를 사용하는 것이다.

Activity2.java

Activity2.java

```
package kr.co.company.explicitintent;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class Activity2 extends AppCompatActivity {

  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.layout2);
      Button b = (Button)findViewById(R.id.Button01);
      b.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            finish();
        }
      });
  }
}
```

이벤트 리스너에서는 버튼이 클릭되면 finish() 메소드를 호출하여서 현재의 액티비티를 종료한다.

메니페스트 파일

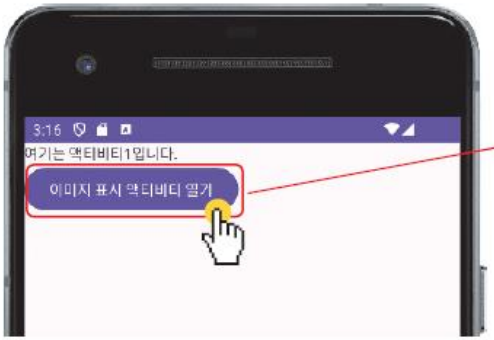
AndroidManifest.xml

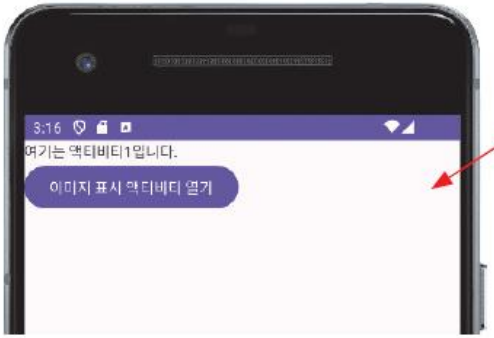
```
...
        <activity
            android:name="kr.co.company.explicitintent.Activity1"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Activity2" android:label="Activity2"></activity>

...
```

android:name에는 액티비티의 클래스 이름을 적어준다. 같은 패키지에 있는 경우에는 앞에 .을 찍거나 아니면 단순히 클래스 이름만 적어준다. 다른 패키지라면 패키지 이름을 포함한 완전한 경로 이름을 적어야 한다. android:label에는 타이틀바에 나타나는 텍스트를 적어준다.

실행 결과

여기서 "이미지 표시 액티비티 열기" 버튼을 누르면 액티비티2가 시작되어서 다음과 같은 화면이 나타난다. 액티비티1이 잠시 중단되고 액티비티2가 새로 시작된 것이다. 액티비티 스택에서 액티비티1 위에 액티비티2가 있을 것이다.

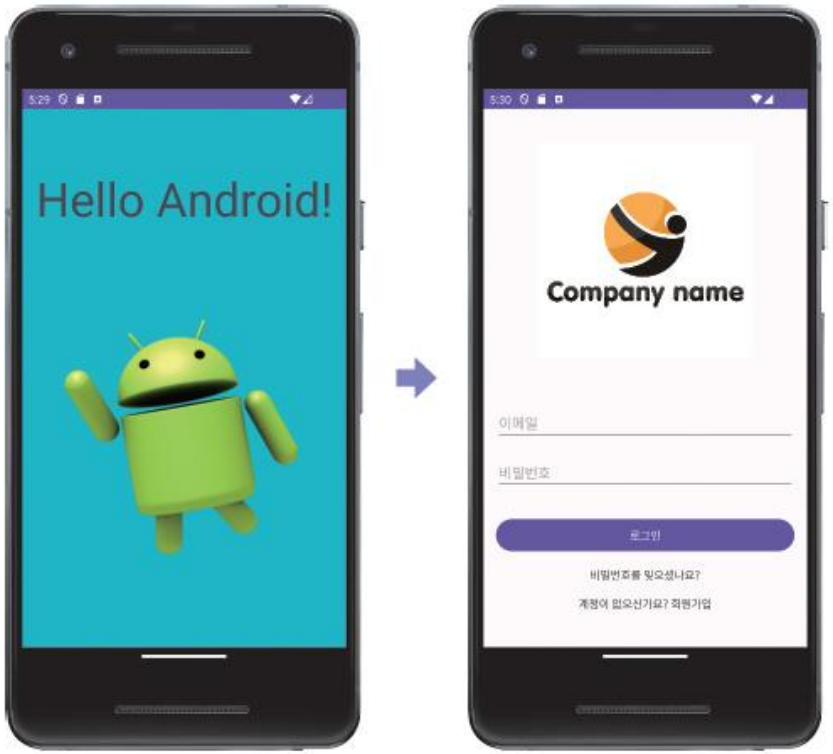




여기서 "닫기" 버튼을 누르면 액티비티2가 종료된다. 따라서 액티비티 스택에 있는 액티비티1이 다시 실행되어서 다음과 같은 화면이 다시 나타난다. 여기서 BACK 키를 눌러도 마찬가지의 결과를 얻는다.

예제: 스플래시 화면

- 앱이 시작되면 2초 동안 스플래시 화면을 보여주고, 2초가 지나면 로그인 화면을 표시하는 앱을 작성해보자.



예제: 스플래시 화면

activity_main.xml

```
<androidx.constraintlayout.widget.ConstraintLayout >
    <ImageView
        android:id="@+id/imageView "
        app:srcCompat="@drawable/img" />

    <TextView
        android:id="@+id/textView"
        android:text="Hello Android!"
        android:textSize="60sp"    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

예제: 스플래시 화면

MainActivity.java

```
public  class MainActivity extends AppCompatActivity {

    private static final int SPLASH_TIMEOUT = 2000; // 스플래시 화면을 보여줄 시간 (2초)

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // 일정 시간 후에 로그인 화면으로 이동
        new Handler().postDelayed(new Runnable() {        // ①
            @Override
            public void run() {
                Intent intent = new Intent(MainActivity.this, LoginActivity.class); // ②
                startActivity(intent);                    // ③
                finish(); // 스플래시 화면을 종료     ④
            }
        }, SPLASH_TIMEOUT);
    }
}
```

예제:

login.xml

```
<LinearLayout>

    <ImageView
        android:id="@+id/logoImageView"
        android:src="@drawable/companylogo" />

    <EditText
        android:id="@+id/emailEditText"
        android:hint="이메일"
        android:inputType="textEmailAddress"/>

    <EditText
        android:id="@+id/passwordEditText"
        android:hint="비밀번호"
        android:inputType="textPassword"/>

    <Button
        android:id="@+id/loginButton"
        android:text="로그인"/>

    <TextView
        android:id="@+id/forgotPasswordTextView"
        android:text="비밀번호를 잊으셨나요?" />

    <TextView
        android:id="@+id/registerTextView"
        android:text="계정이 없으신가요? 회원가입" />

</LinearLayout>
```

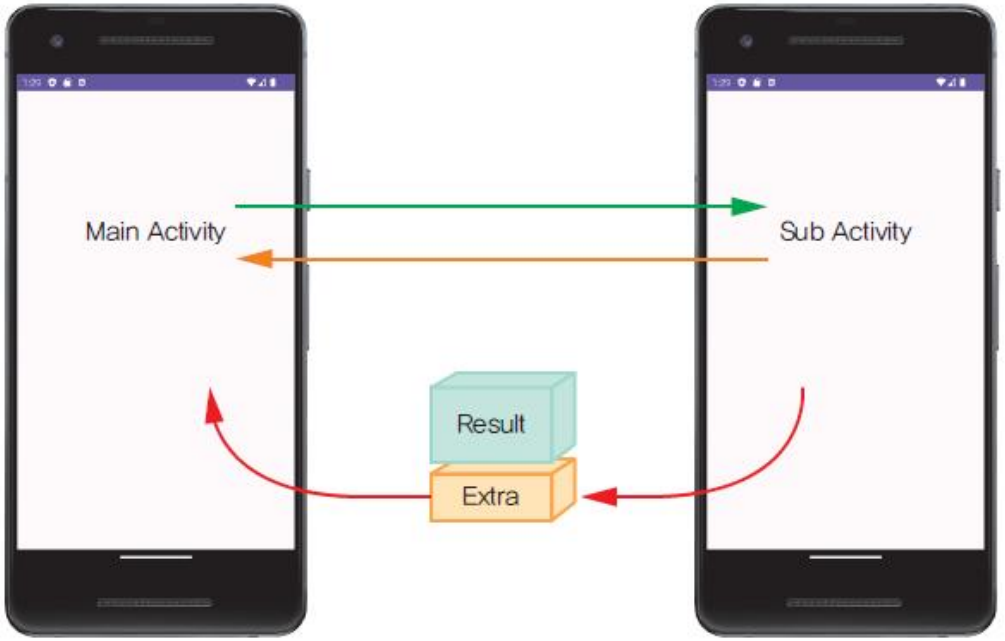
예제: 스플래시 화면

LoginActivity.java

```
public class LoginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);

        // 로그인 화면의 레이아웃을 정의하고 필요한 로직을 추가합니다.
    }
}
```

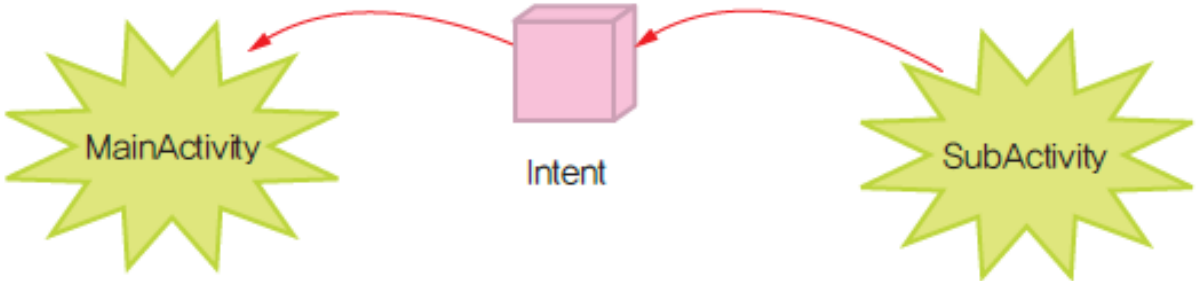
액티비티에서 결과받기

- 가끔은 서브 액티비티로부터 결과를 받아야 하는 경우가 있다. 이런 경우에는 startActivity( )가 아닌 startActivityForResult( )를 호출하여서 서브 액티비티를 시작하여야 한다.



값을 저장하고, 값을 읽는 메소드

- 서브 액티비티가 보내는 결과값은 무엇을 통하여 전달될까? 인텐트 객체 안에는 있는 엑스트라(extras) 필드가 이용된다.

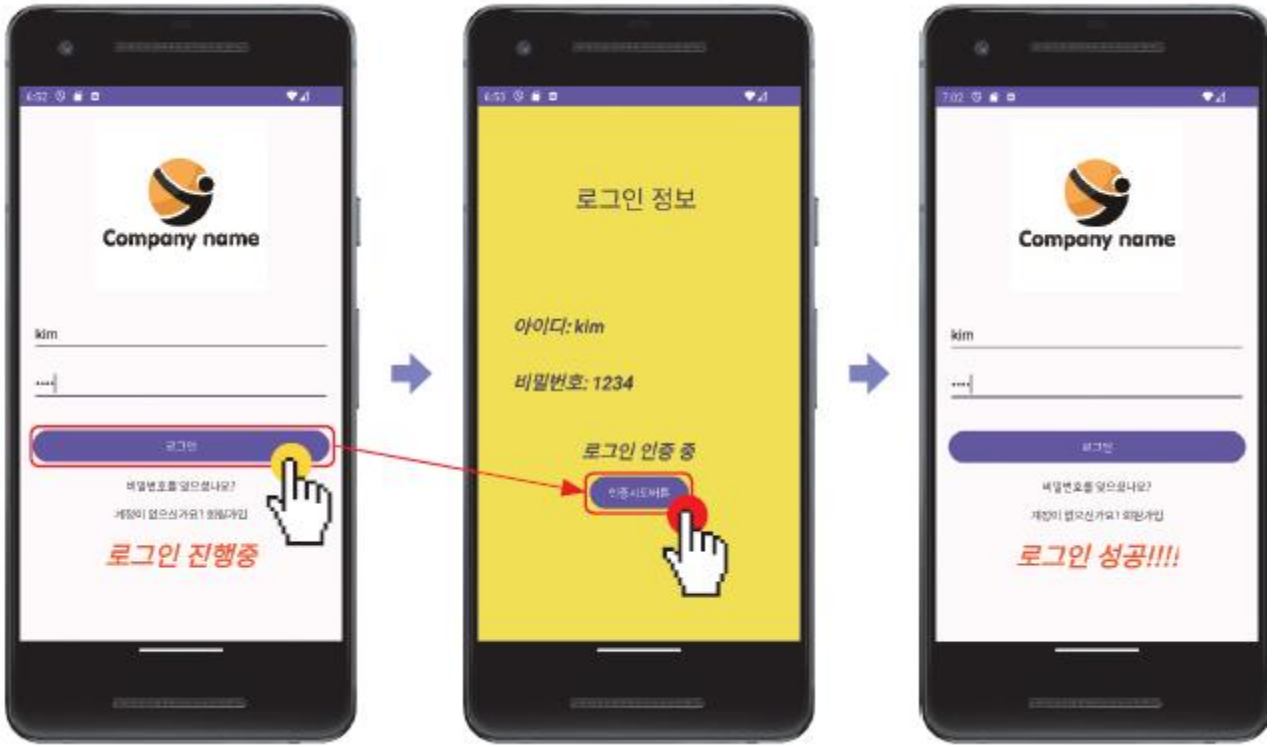


int getIntExtra(String name, int defaultValue)
float getFloatExtra(String name, float defaultValue)
int[] getIntArrayExtra(String name)
String getStringExtra(String name)

putExtra(String name, int value)
putExtra(String name, float value)
putExtra(String name, int[] value)
putExtra(String name, String value)

예제

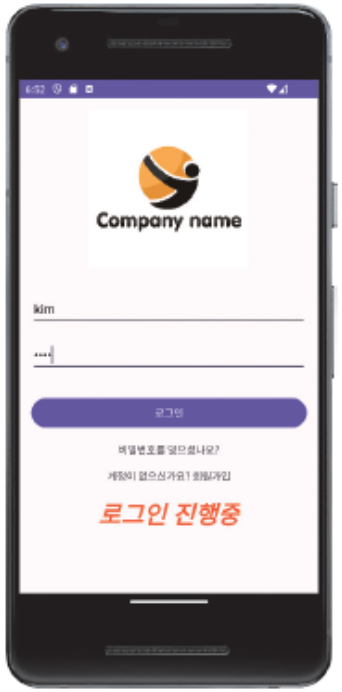
- 로그인 액티비티에서 아이디와 비밀번호를 입력하면 이것을 서브 액티비티로 넘기고, 서브 액티비티에서 아이디와 비밀번호를 인증한 후에, 인증 결과를 메인 액티비티로 넘기는 앱을 작성하여 보자.



예제

activity_main.xml

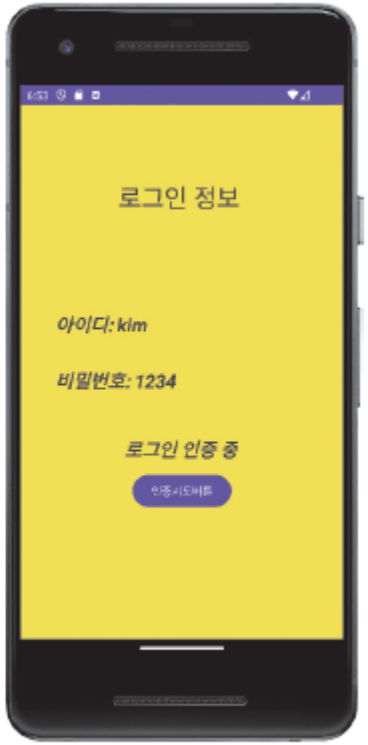
```
<LinearLayout>
    <ImageView
        android:id="@+id/logoImageView"
        android:src="@drawable/companylogo" />

    <EditText
        android:id="@+id/emailEditText"
        android:hint="이메일"
        android:inputType="textEmailAddress"/>

    <EditText
        android:id="@+id/passwordEditText"
        android:hint="비밀번호"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/loginButton"
        android:text="로그인"/>

    <TextView
        android:id="@+id/forgotPasswordTextView"
        android:text="비밀번호를 잊으셨나요?" />
```



예제

sub.xml

```
<androidx.constraintlayout.widget.ConstraintLayout >

    <TextView
        android:id="@+id/nnn"
        android:text="로그인 정보 " />


    <TextView
        android:id="@+id/displayIdTextView"
        android:text="아이디: " />

    <TextView
        android:id="@+id/displayPasswordTextView"
        android:text="비밀번호: " />

    <TextView
        android:id="@+id/loginSuccess"
        android:text="로그인 인증 중" />

    <Button
        android:id="@+id/button"
        android:text="인증시도버튼" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



예제

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
...
    <application
    ...
        <activity
            android:name="kr.co.company.activityforresult.MainActivity"
            android:label="@string/app_name" >
        ...
        </activity>
        <activity
            android:name=".SecondActivity"
        </activity>
    </application>

</manifest>
```

서브 액티비티 등록

예제

MainActivity.java

```
...
public class MainActivity extends AppCompatActivity {
    private EditText emailEditText, passwordEditText;
    private TextView statusText;
    private Button loginButton;
    ActivityResultLauncher<Intent> launcher;     // ①

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        emailEditText = findViewById(R.id.emailEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        loginButton = findViewById(R.id.loginButton);
        statusText = findViewById(R.id.loginStatus);
```

예제

```
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 아이디와 비밀번호 입력값을 가져오기
                String email = emailEditText.getText().toString();
                String password = passwordEditText.getText().toString();

                // 두 번째 화면으로 전환하면서 아이디와 비밀번호 데이터 전달
                Intent intent = new Intent(MainActivity.this, SecondActivity.class);
                intent.putExtra("ID", email);
                intent.putExtra("Password", password);
                launcher.launch(intent);
            }
        });
        launcher = registerForActivityResult(new ActivityResultContracts.
        StartActivityForResult(),       // ②
                result -> {             // ③
                    if (result.getResultCode() == Activity.RESULT_OK) {
                        Intent data = result.getData();
                        statusText.setText(data.getStringExtra("status"));
                    }
                });
    }
}
```

예제

SubActivity.java

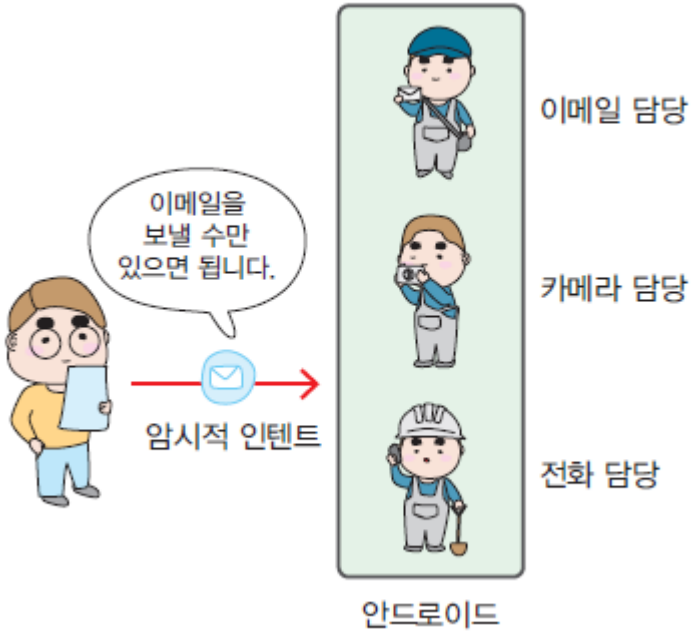
```
public class SecondActivity extends AppCompatActivity {

    private TextView displayIdTextView, displayPasswordTextView, statusTextView;
    String id, password;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        displayIdTextView = findViewById(R.id.displayIdTextView);
        displayPasswordTextView = findViewById(R.id.displayPasswordTextView);
        statusTextView = findViewById(R.id.loginSuccess);

        // 인텐트에서 아이디와 비밀번호 데이터 가져오기
        Intent intent = getIntent();
        if (intent != null) {
            id = intent.getStringExtra("ID");
            password = intent.getStringExtra("Password");

            // 화면에 아이디와 비밀번호 출력
            displayIdTextView.setText("아이디: " + id);
            displayPasswordTextView.setText("비밀번호: " + password);
        }
    }
```

예제

```
    public void check(View e){
        // 로그인 로직 (더미 데이터 사용)
        Intent intent= new Intent();
        if (isUserValid(id, password)) {
            // 인증 성공 시 메인 액티비티로 이동
            intent.putExtra("status", "로그인 성공!!!!");
        } else {

            // 인증 실패 시 메시지 표시
            // 실제로는 실패 처리 및 메시지를 표시하는 방식을 변경해야 한다.
            intent.putExtra("status", "로그인 실패!!!!");
        }
        setResult(RESULT_OK, intent);
        finish();
    }
    private boolean isUserValid(String username, String password) {
        // 실제로는 여기에서 서버 또는 로컬 데이터베이스를 통해 인증을 확인해야 한다.
        // 이 예제에서는 더미 데이터를 사용하여 단순하게 인증 성공 여부를 판단한다.
        return username.equals("kim") && password.equals("1234");
    }
}
```

암시적인 인텐트

- 어떤 작업을 하기를 원하지만 그 작업을 담당하는 컴포넌트의 이름을 명확하게 모르는 경우에 사용



암시적인 인텐트의 형식

전체 구조

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

이메일 전송을 의미하는 인텐트 생성

이메일의 송신자를 엑스트라 필드에 기술한다.

액션의 종류

상수	타깃 컴포넌트	액션
ACTION_VIEW	액티비티	데이터를 사용자에게 표시한다.
ACTION_EDIT	액티비티	사용자가 편집할 수 있는 데이터를 표시한다.
ACTION_MAIN	액티비티	태스크의 초기 액티비티로 설정한다.
ACTION_CALL	액티비티	전화 통화를 시작한다.
ACTION_SYNC	액티비티	모바일 장치의 데이터를 서버 상의 데이터와 일치시킨다.
ACTION_DIAL	액티비티	전화번호를 누르는 화면을 표시한다.

암시적인 인텐트 예

액션 / 데이터

ACTION_VIEW	*content://contacts/people/1* — 1번 연락처 정보를 표시한다.
ACTION_DIAL	*content://contacts/people/1* — 1번 연락처로 전화걸기 화면을 표시한다.
ACTION_VIEW	*tel:0101234567* — 0101234567번 전화번호 정보를 표시한다.
ACTION_DIAL	*tel:0101234567* — 0101234567번 전화번호로 전화걸기 화면을 표시한다.
ACTION_EDIT	*content://contacts/people/1* — 1번 연락처 정보를 편집한다.
ACTION_VIEW	*content://contacts/people/* — 연락처 리스트를 표시한다.

암시적인 인테트의 예

```
...
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:01012341234"));
startActivity(intent);
...
```

액션이 ACTION_CALL인 인텐트를 생성한다.

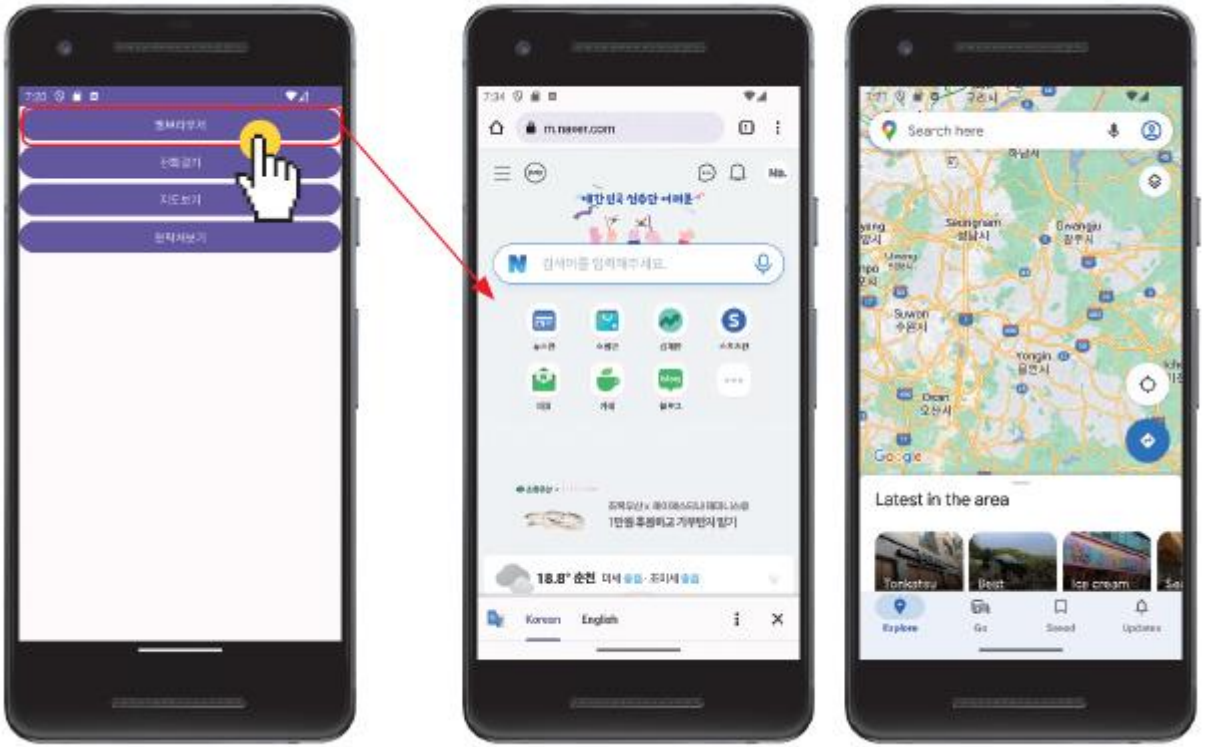
0101234567번 전화번호를 데이터로 설정한다.

인텐트를 시작한다.

암시적인 인텐트의 카테고리

상수	설명
CATEGORY_BROWSABLE	타깃 액티비티가 브라우저에 의해 시작되어서 이미지와 같은 데이터를 표시할 수 있다.
CATEGORY_GADGET	액티비티가 다른 액티비티 안에 개짓으로 내장된다.
CATEGORY_HOME	홈 화면을 표시하는 액티비티이다.
CATEGORY_LAUNCHER	액티비티가 최상위 애플리케이션으로 론처에 나열된다.
CATEGORY_PREFERENCE	타깃 액티비티가 환경 설정 패널이다.

예제: 암시적인 인텐트



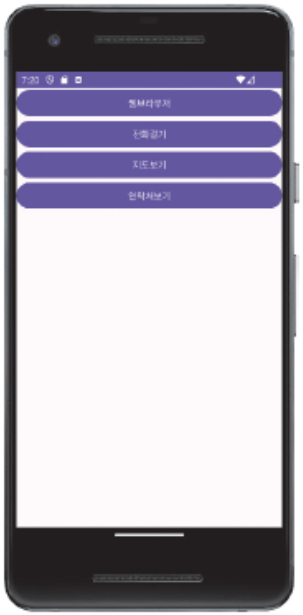
사용자 인터페이스

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/call"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

사용자 인터페이스

```
        android:onClick="onClick"
        android:text="전화걸기" >
    </Button>
    <Button
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="지도보기" >
    </Button>
    <Button
        android:id="@+id/web"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="웹브라우저" >
    </Button>
    <Button
        android:id="@+id/contact"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="연락처보기" >
    </Button>
</LinearLayout>
```



암시적 인텐트 예제

MainActivity.java

```
package kr.co.company.implicitintent;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class MainActivity extends AppCompatActivity {

  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
  }

  public void onClick(View view) {
      Intent intent = null;
      switch (view.getId()) {
      case R.id.web:
          intent = new Intent(Intent.ACTION_VIEW,
```

암시적 인텐트 예제

```
                Uri.parse("http://www.google.com"));          ← 암시적 인텐트로 웹페이지를 본다.
            break;
        case R.id.call:
            intent = new Intent(Intent.ACTION_DIAL,
                    Uri.parse("tel:(+82)12345789"));          ← 암시적 인텐트로 전화를 건다.
            break;
        case R.id.map:
            intent = new Intent(Intent.ACTION_VIEW,
                    Uri.parse("geo:37.30,127.2?z=10"));       ← 암시적 인텐트로 지도를 본다.
            break;                                              이것은 실제 장치에서만 가능
                                                                하다.
        case R.id.contact:
            intent = new Intent(Intent.ACTION_VIEW,
                    Uri.parse("content://contacts/people/"));  ← 암시적 인텐트로 연락처를 본다.
            break;
        }
        if (intent != null) {
            startActivity(intent);
        }
    }

}
```

암시적 인텐트 예제

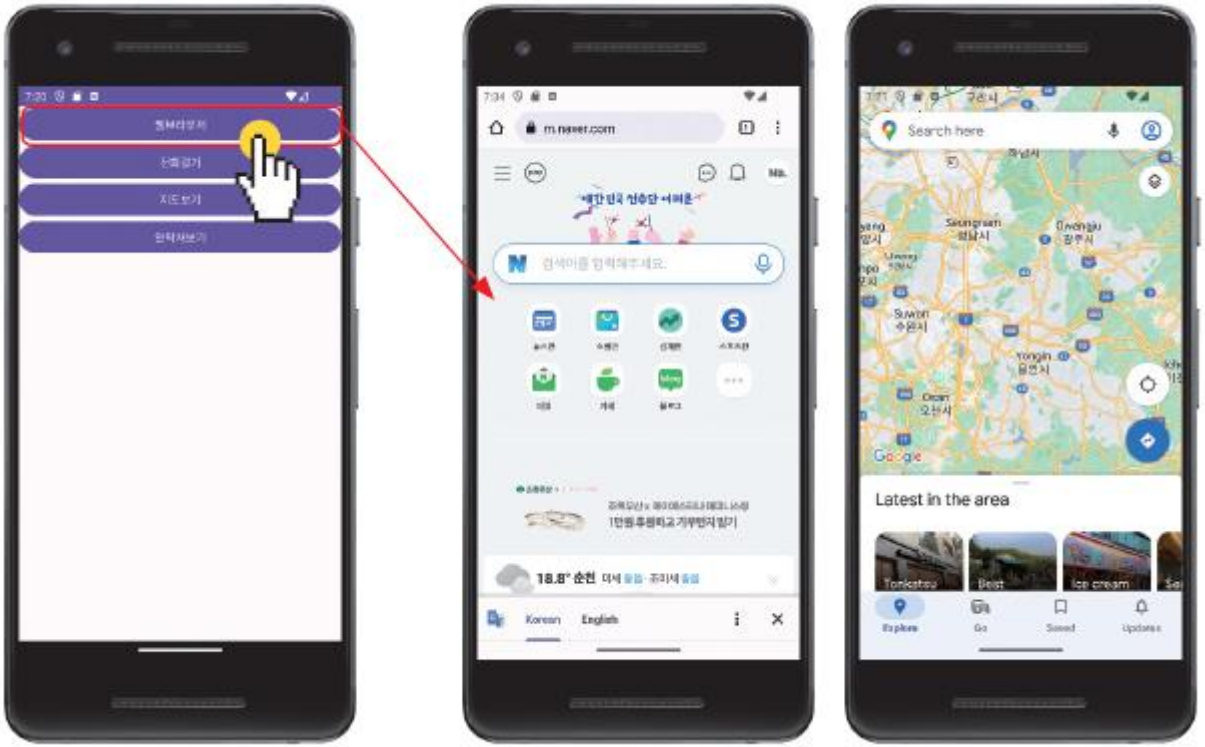
매니페스트 파일 수정

AndroidManifest.xml

```
...
    <uses-permission android:name="android.permission.CALL_PHONE" >
    </uses-permission>
    <uses-permission android:name="android.permission.CAMERA" >
    </uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS" >
    </uses-permission>
    <uses-permission android:name="android.permission.INTERNET" />

...
```

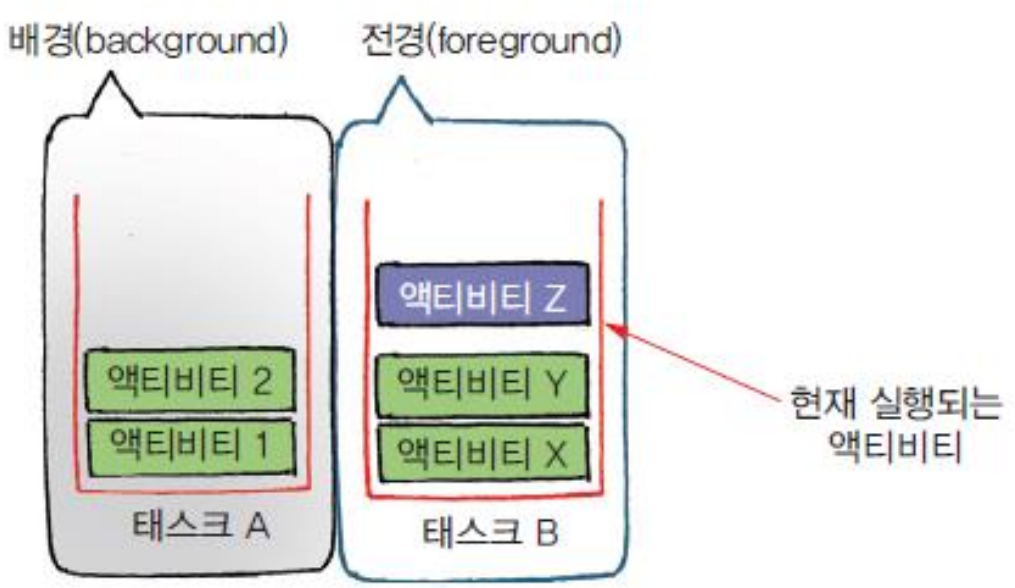
각종 권한을 요청한다.

실행결과



멀티태스킹

- 동시에 여러 태스크를 실행
- 현재의 태스크를 배경(background)으로 보내고 다른 태스크를 전경(foreground)에서 시작할 수 있다.



멀티태스킹을 시작하는 방법

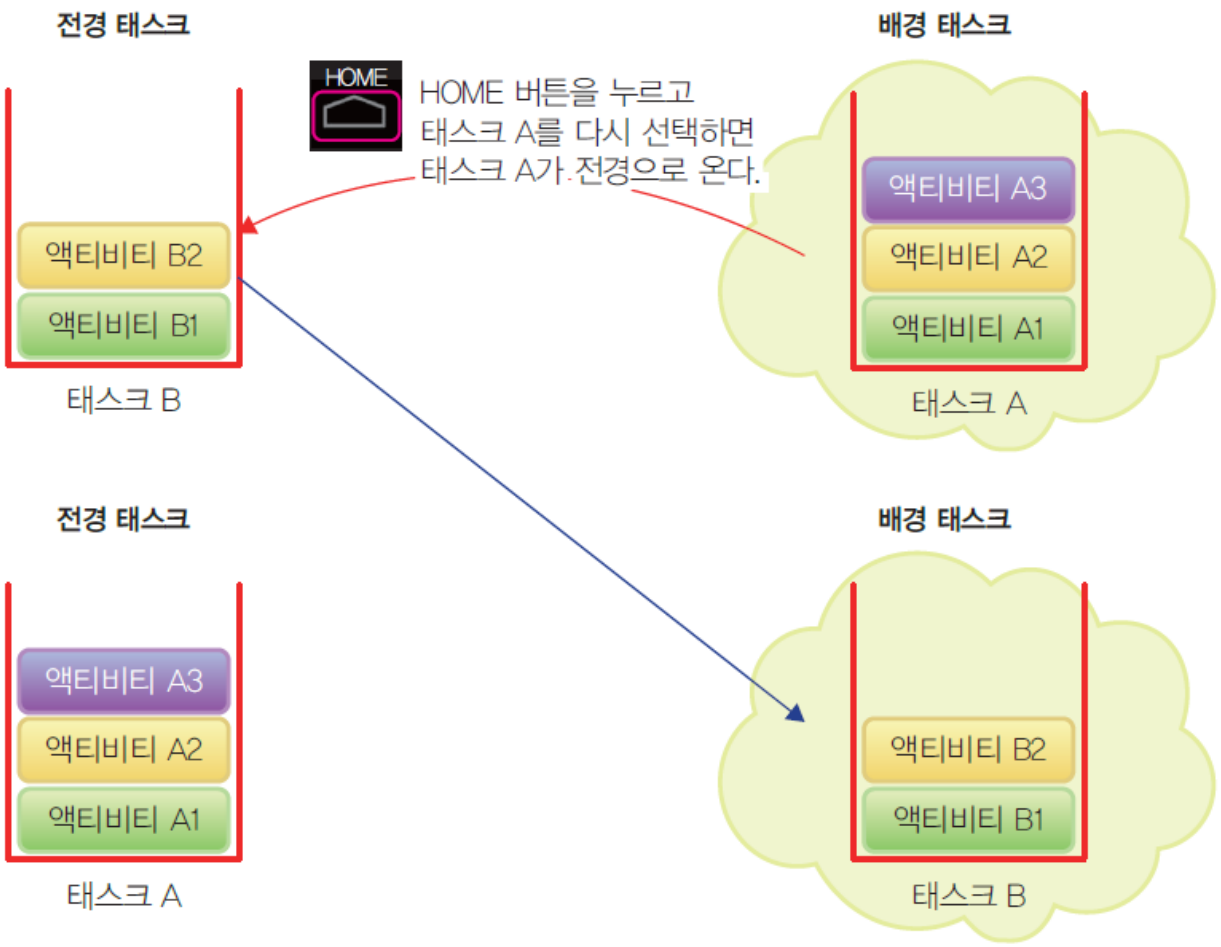
- 안드로이드에서 멀티태스킹을 시작하는 가장 일반적인 방법은 HOME 키( )를 누르는 것이다.

멀티 태스킹의 예


사용자 동작
전경 태스크
배경 태스크
HOME 버튼
사용자 동작
전경 태스크
배경 태스크
HOME 버튼
계산기 시작
123×4
492
사용자 동작
전경 태스크
배경 태스크
HOME 버튼
계산기 시작
HOME 버튼
123×4
사용자 동작
전경 태스크
배경 태스크
HOME 버튼
계산기 시작
HOME 버튼
칼렌다 시작
March 2015
123×4

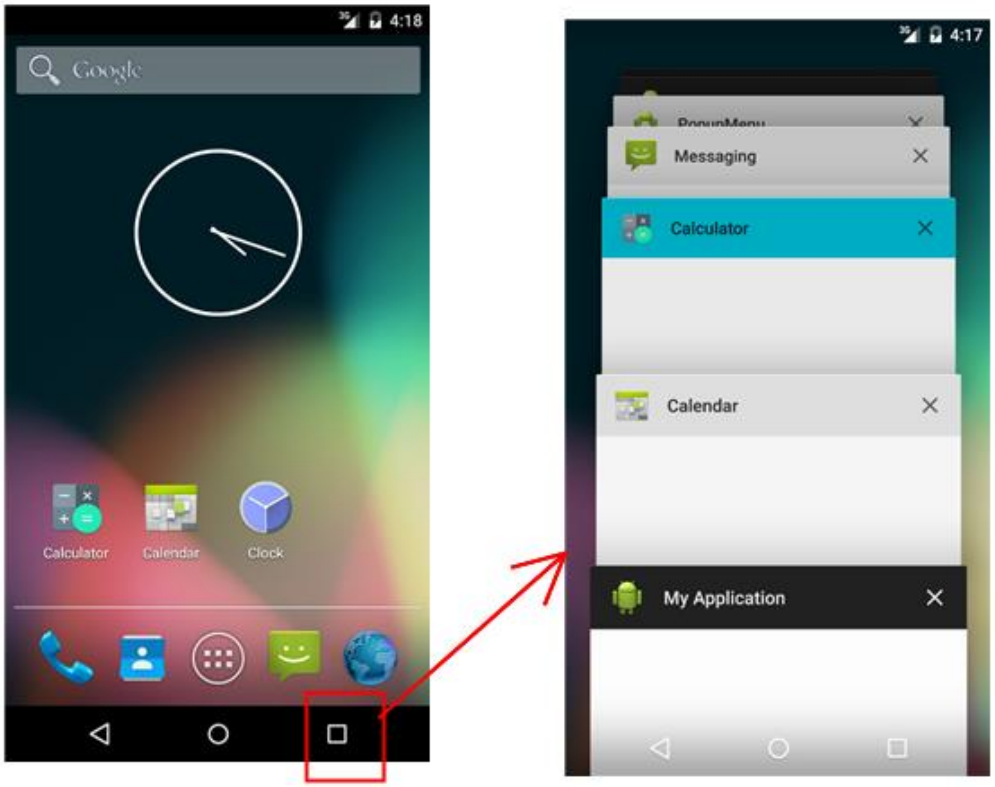
HOME 키를 누르면 멀티 태스킹이 시작된다.

태스크 A를 시작하였던 애플리케이션 아이콘을 선택한다고 하자



오버뷰 화면

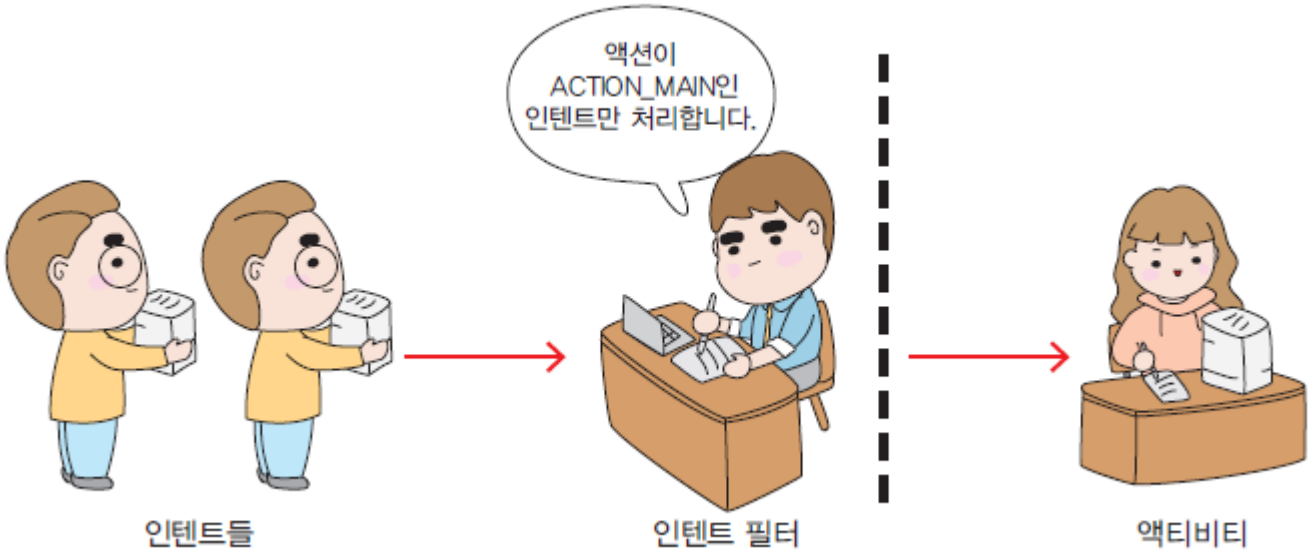
- 최근에 사용된 액티비티들과 태스크들을 보여주는 화면



인텐트 필터

- 컴포넌트는 자신들이 처리할 수 있는 인텐트의 종류를 인텐트 필터에 기록한다.

인텐트 필터



인텐트 필터

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kr.co.company.implicitintent"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.CAMERA" />
```

인텐트 필터

```
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="kr.co.company.implicitintent.ImplicitIntentActivity"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

액티비티 ImplicitIntent Activity인텐트 필터

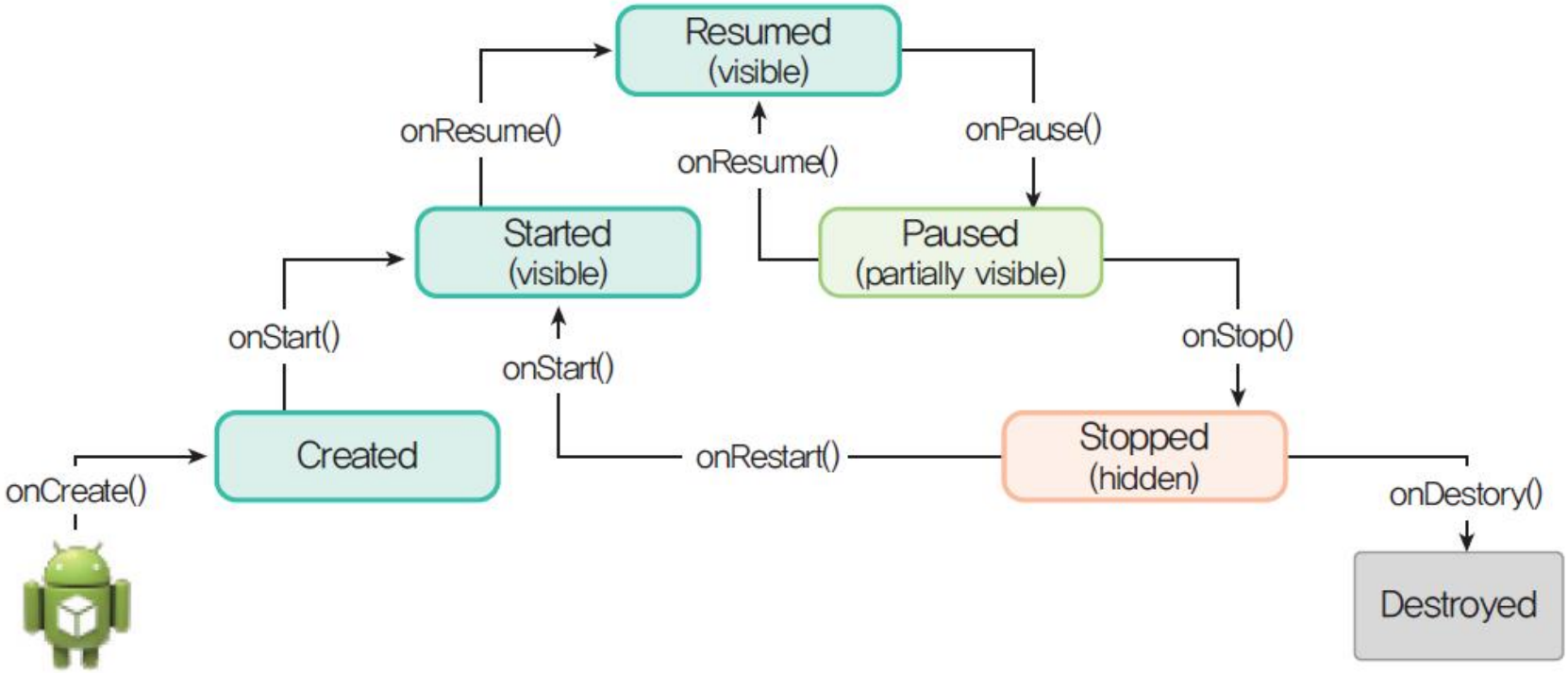
액티비티 생애주기

- **실행 상태(resumed, running):** 액티비티가 전경에 위치하고 있으며 사용자의 포커스를 가지고 있다.

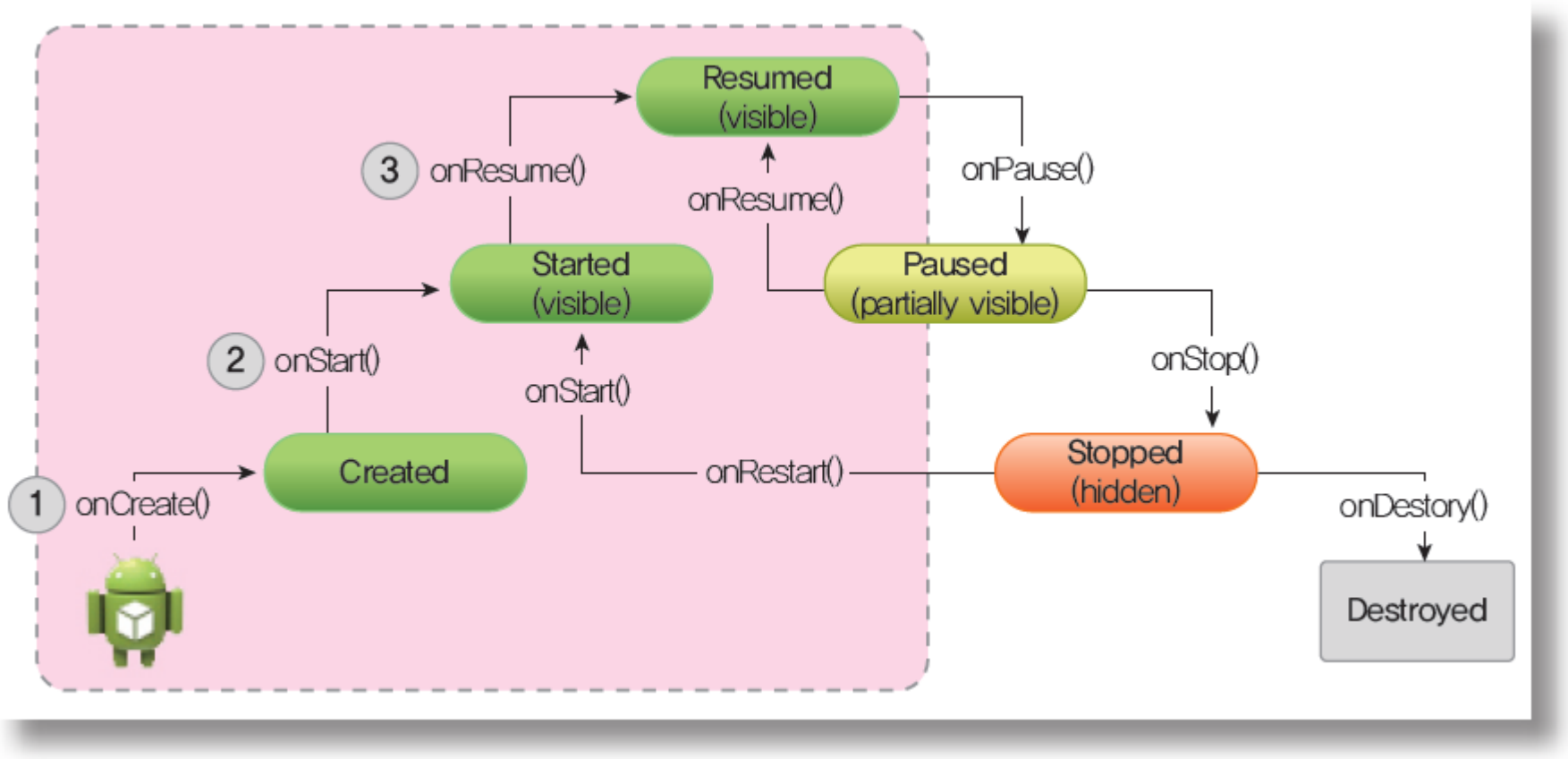
- **일시멈춤 상태(paused):** 다른 액티비티가 전경에 있으며 포커스를 가지고 있지만 현재 액티비티의 일부가 아직도 화면에서 보이고 있는 상태이다.

- **정지 상태(stopped):** 액티비티는 배경에 위치한다.

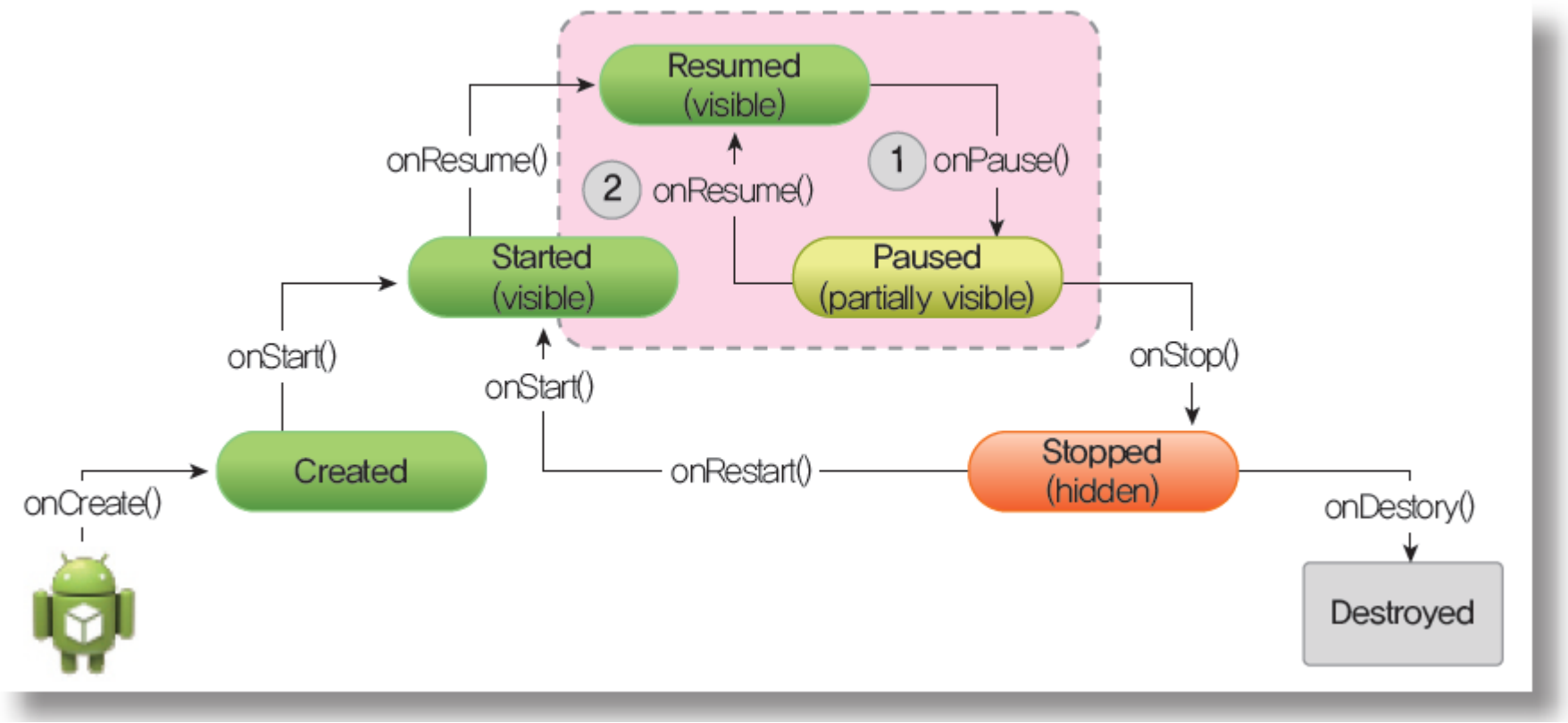
액티비티 상태


Resumed
(visible)
onResume()
onResume()
onPause()
Started
(visible)
Paused
(partially visible)
onStart()
onStart()
onStop()
Created
onRestart()
Stopped
(hidden)
onCreate()
onDestory()
Destroyed

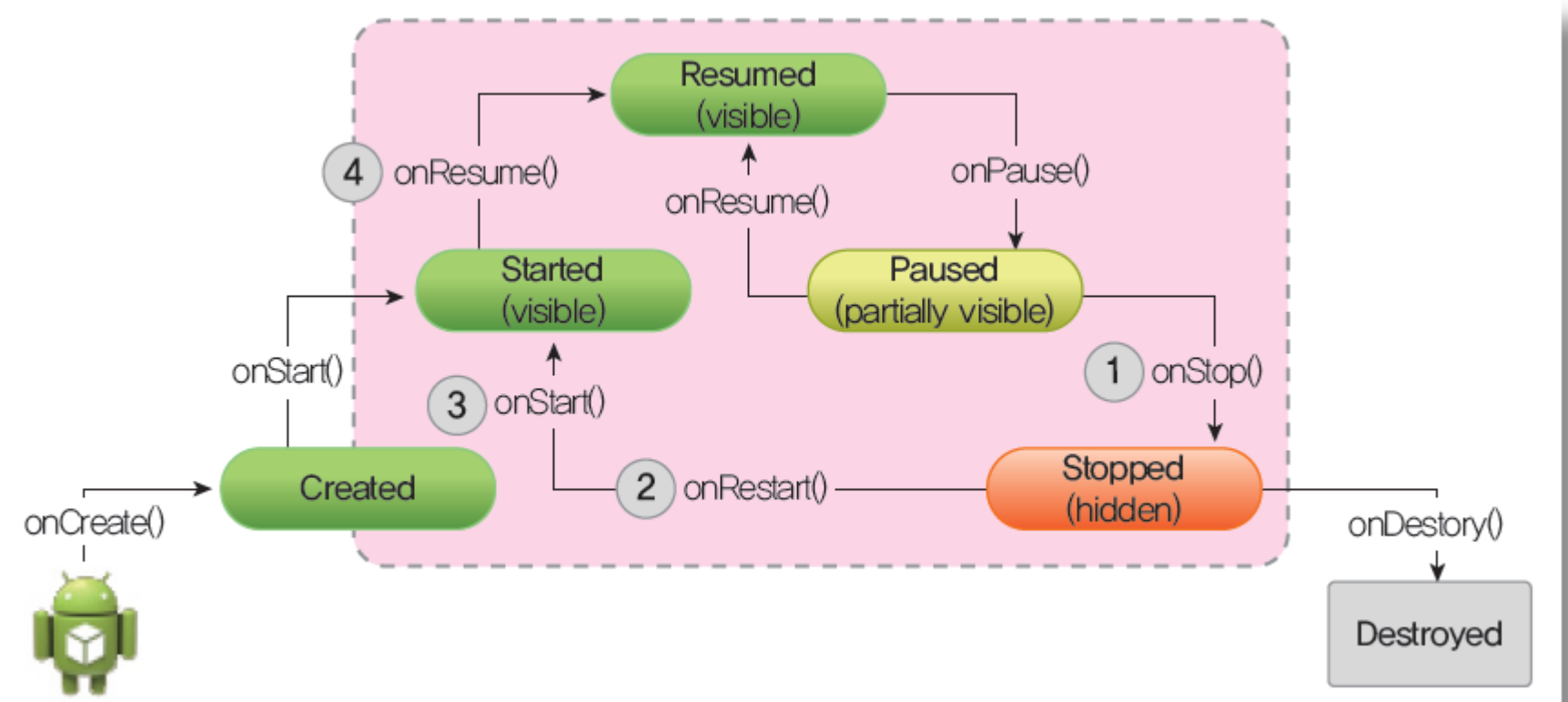
액티비티 객체 생성 단계


Resumed
(visible)
3 onResume()
onResume()
onPause()
Started
(visible)
Paused
(partially visible)
2 onStart()
onStop()
onStart()
Created
onRestart()
Stopped
(hidden)
1 onCreate()
onDestory()
Destroyed

일시 멈춤 상태


Resumed
(visible)
onResume()
2 onResume()
1 onPause()
Started
(visible)
Paused
(partially visible)
onStart()
onStart()
onStop()
Created
onRestart()
Stopped
(hidden)
onCreate()
onDestory()
Destroyed

정지되었다가 다시 실행하는 경우

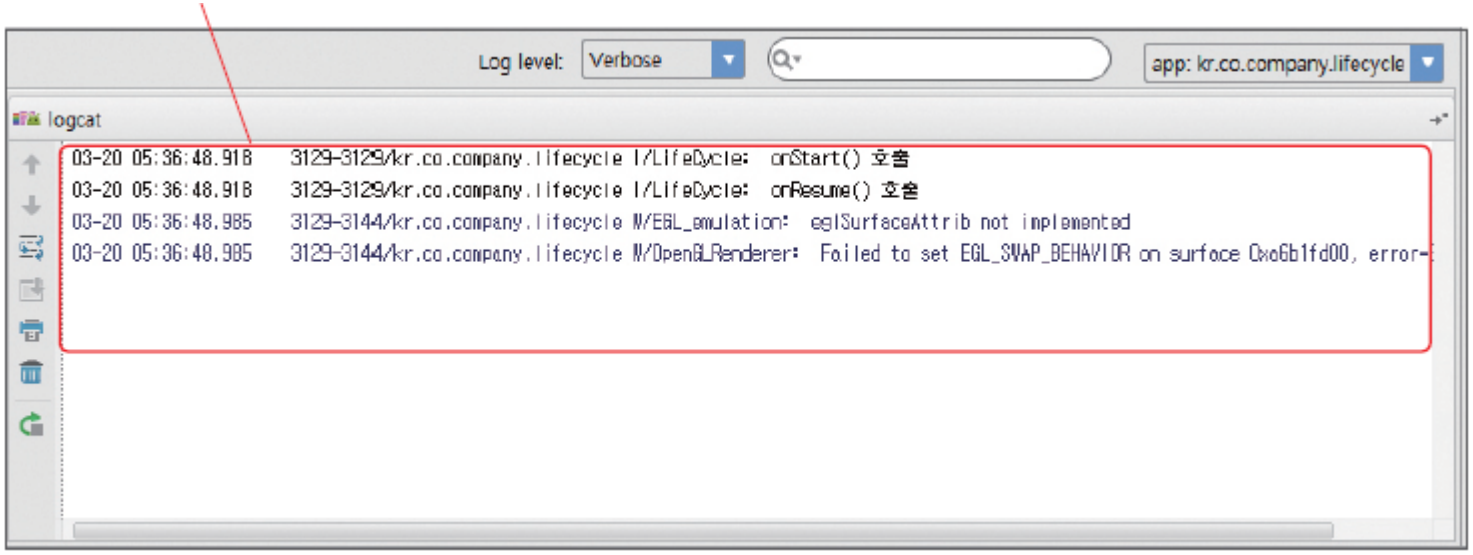


중요한 콜백 메소드

- **onCreate()**
  - 액티비티가 생성되면서 호출
  - 중요한 구성요소들을 초기화
- **onPause()**
  - 사용자가 액티비티를 떠나고 있을 때, 이 메소드가 호출
  - 그 동안 이루어졌던 변경사항을 저장

예제: 액티비티 생애주기

- 간단한 애플리케이션을 만들어서 생애주기의 메소드를 재정의하고 여기에서 간단하게 출력을 하여 보자.



예제: 액티비티 생애주기

MainActivity.java

```
package kr.co.company.lifecycle;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class MainActivity extends AppCompatActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
  }

  @Override
  public void onStart() {
      super.onStart();
      Log.i("LifeCycle", "onStart() 호출");
  }

  @Override
  public void onResume() {
      super.onResume();
      Log.i("LifeCycle", "onResume() 호출");
  }
```

onCreate()는 반드시 구현해야 하는 메소드이다. 안드로이드 시스템은 액티비티를 생성하면서 이 메소드를 호출한다. 이 메소드 안에서는 액티비티의 컴포넌트들을 초기화하여야 한다. 가장 중요한 작업은 setContentView()를 호출하여서 액티비티의 사용자 인터페이스 화면을 설정하여야 한다는 것이다.

액티비티가 화면에 보여질 예정이다.

액티비티가 화면에 보여진다.

예제: 액티비티 생애주기

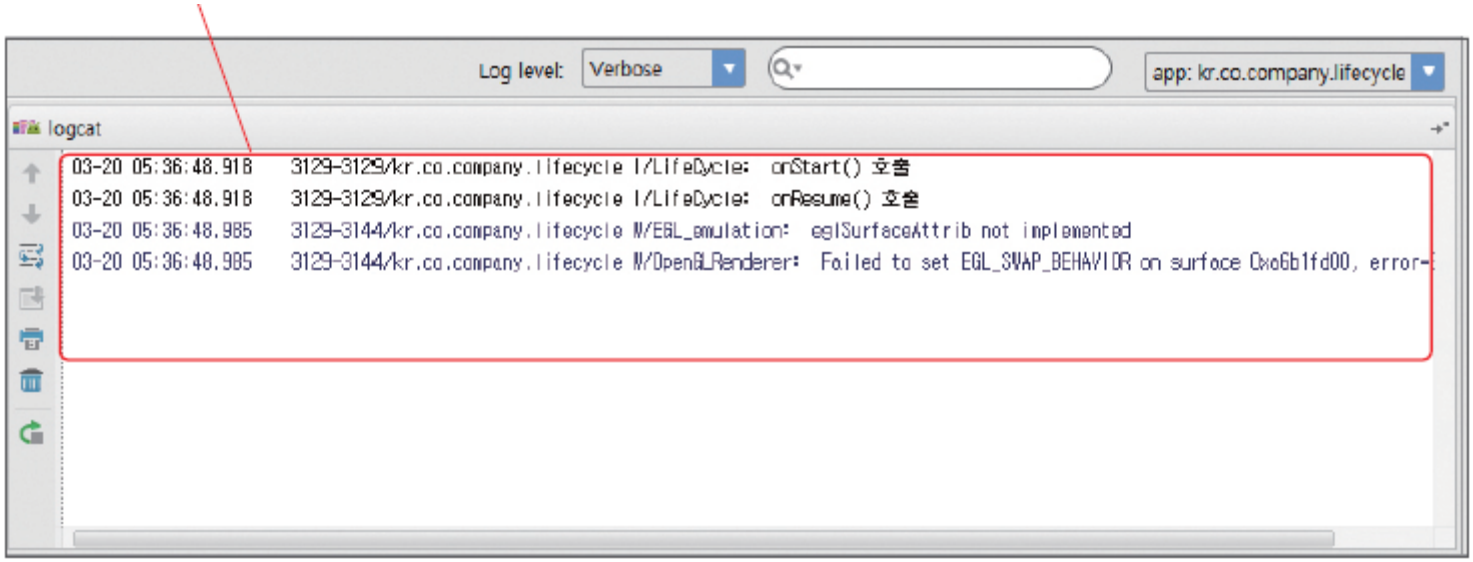
```
@Override
public void onPause() {
    super.onPause();
    Log.i("LifeCycle", "onPause() 호출");
}

@Override
public void onStop() {
    super.onStop();
    Log.i("LifeCycle", "onStop() 호출");
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.i("LifeCycle", "onDestroy() 호출");
}
}
```

액티비티가 포커스를 잃는다. onPause()는 사용자가 액티비티를 떠나려고 할 때 호출된다. 사용자가 액티비티를 떠난다고 해서 반드시 액티비티가 소멸되는 것은 아니다. 이 곳에서는 사용자가 돌아오지 않을 경우에 대비하여서 반드시 기록되어야 하는 변경 사항이 있으면 저장하여야 한다.

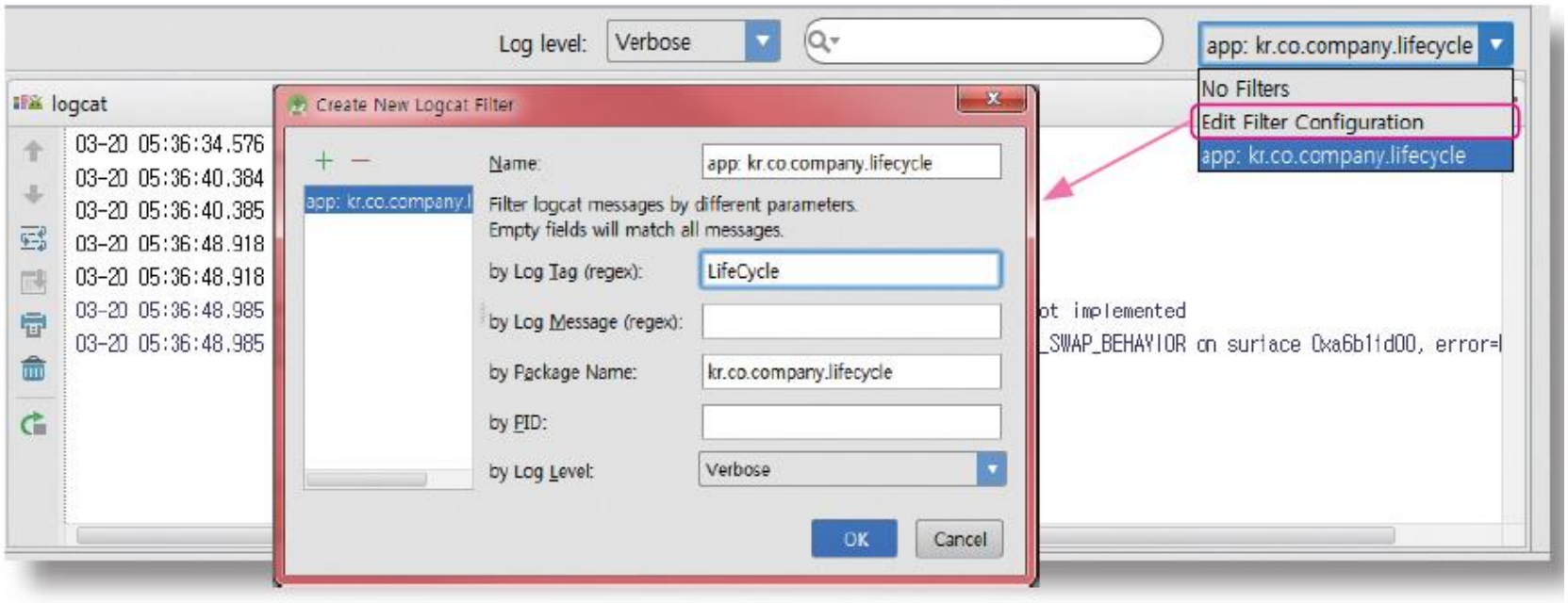
액티비티가 정지하기 직전에 호출된다.

액티비티가 제거되기 직전에 호출된다.

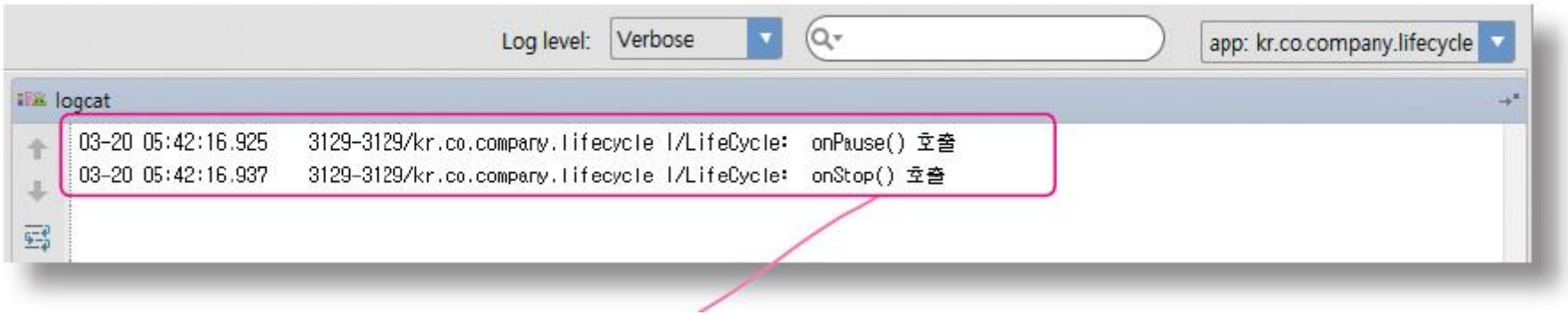
실행 결과


Log level: Verbose
app: kr.co.company.lifecycle
logcat
03-20 05:36:48.918 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStart() 호출
03-20 05:36:48.918 3129-3129/kr.co.company.lifecycle I/LifeCycle: onResume() 호출
03-20 05:36:48.985 3129-3144/kr.co.company.lifecycle W/EGL_emulation: eglSurfaceAttrib not implemented
03-20 05:36:48.985 3129-3144/kr.co.company.lifecycle W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0xa6b1fd00, error=

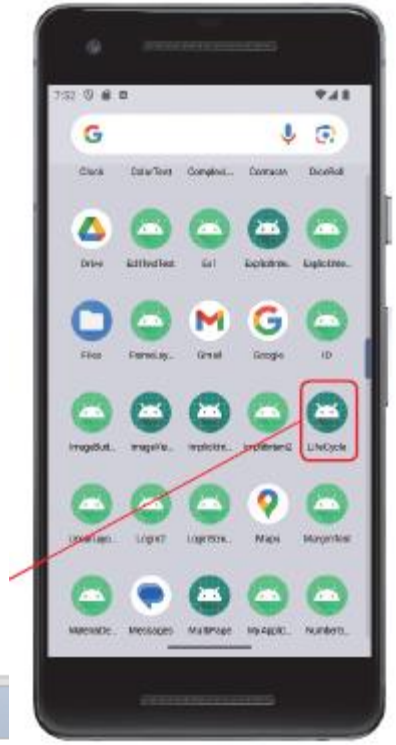
로그캣 필터 지정

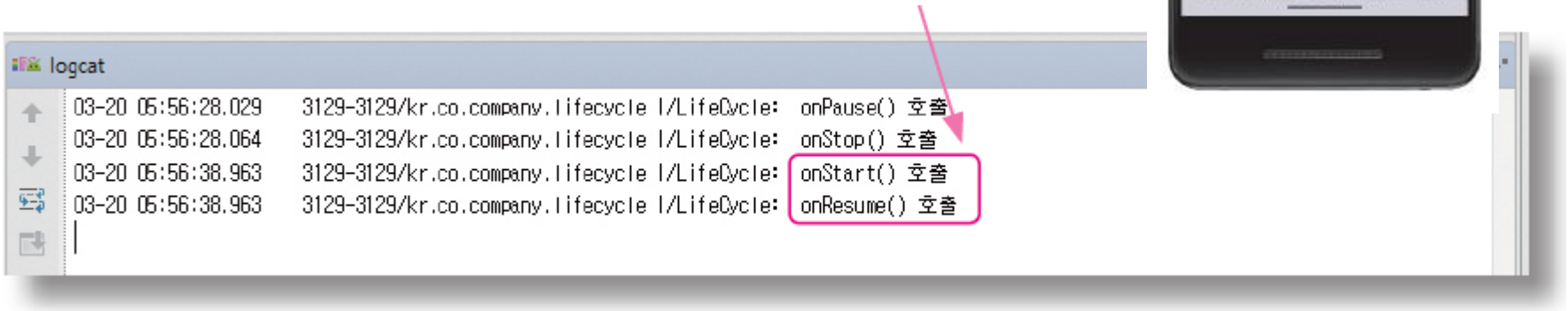

Log level:
Verbose
app: kr.co.company.lifecycle
No Filters
Edit Filter Configuration
app: kr.co.company.lifecycle
logcat
03-20 05:36:34.576
03-20 05:36:40.384
03-20 05:36:40.385
03-20 05:36:48.918
03-20 05:36:48.918
03-20 05:36:48.985
03-20 05:36:48.985
Create New Logcat Filter
Name:
app: kr.co.company.lifecycle
Filter logcat messages by different parameters.
Empty fields will match all messages.
by Log Tag (regex):
LifeCycle
by Log Message (regex):
by Package Name:
kr.co.company.lifecycle
by PID:
by Log Level:
Verbose
OK
Cancel

실행 결과(HOME 키)

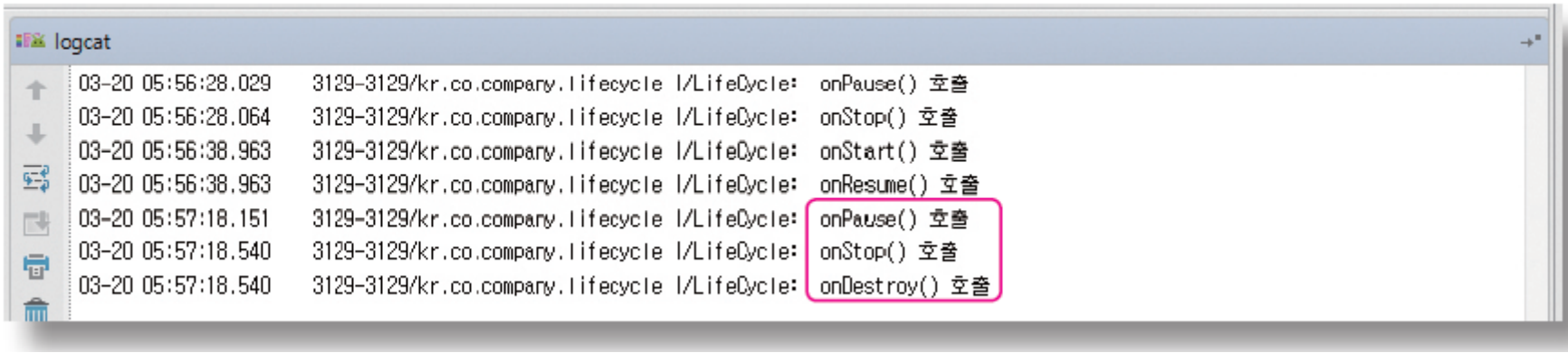

Log level:
Verbose
app: kr.co.company.lifecycle
logcat
03-20 05:42:16.925 3129-3129/kr.co.company.lifecycle I/LifeCycle: onPause() 호출
03-20 05:42:16.937 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStop() 호출

실행 결과(다시 앱 실행)



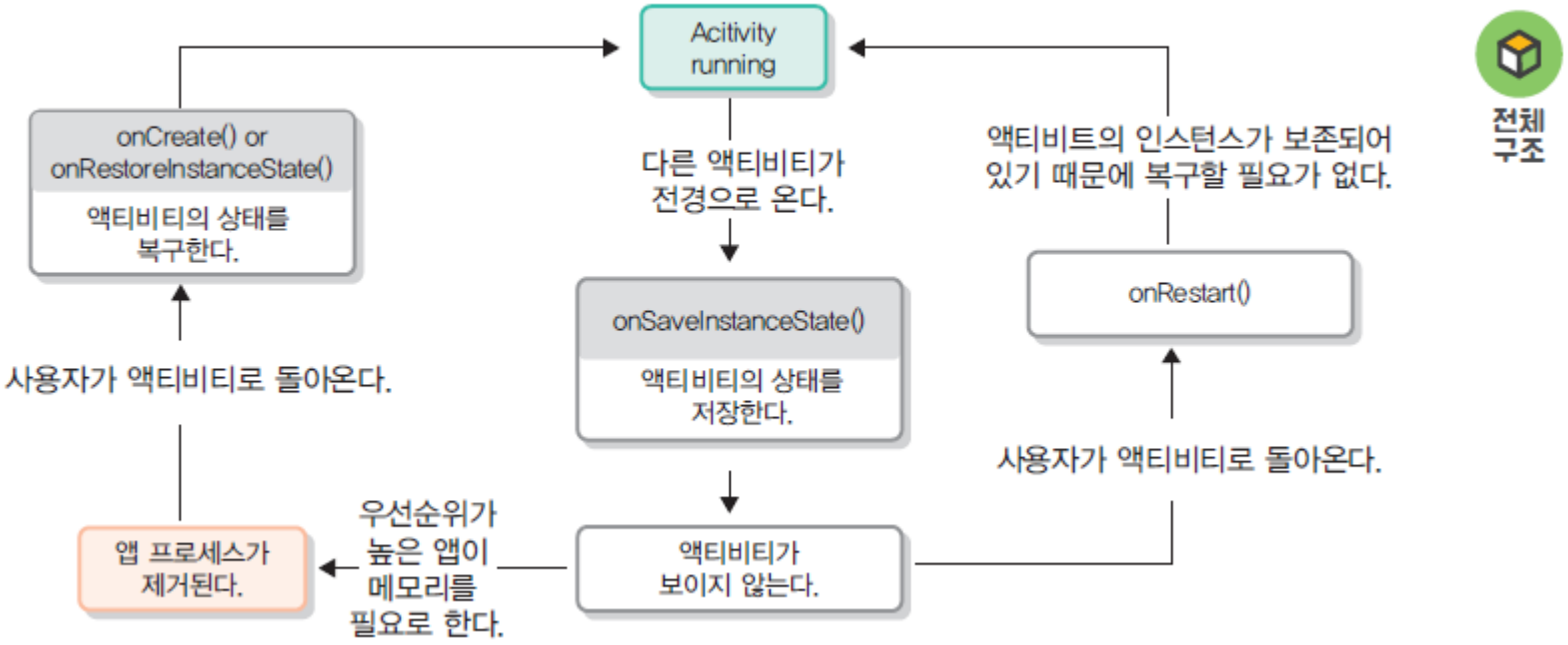

logcat
03-20 05:56:28.029 3129-3129/kr.co.company.lifecycle I/LifeCycle: onPause() 호출
03-20 05:56:28.064 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStop() 호출
03-20 05:56:38.963 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStart() 호출
03-20 05:56:38.963 3129-3129/kr.co.company.lifecycle I/LifeCycle: onResume() 호출

실행 결과(BACK 키)


logcat
03-20 05:56:28.029 3129-3129/kr.co.company.lifecycle I/LifeCycle: onPause() 호출
03-20 05:56:28.064 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStop() 호출
03-20 05:56:38.963 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStart() 호출
03-20 05:56:38.963 3129-3129/kr.co.company.lifecycle I/LifeCycle: onResume() 호출
03-20 05:57:18.151 3129-3129/kr.co.company.lifecycle I/LifeCycle: onPause() 호출
03-20 05:57:18.540 3129-3129/kr.co.company.lifecycle I/LifeCycle: onStop() 호출
03-20 05:57:18.540 3129-3129/kr.co.company.lifecycle I/LifeCycle: onDestroy() 호출

액티비티 상태 저장

- 시스템이 메모리를 확보하기 위하여 강제로 액티비티를 종료하는 경우에는 액티비티 객체가 파괴되므로 사용자가 변경한 부분은 없어진다. 이런 경우를 대비하여서 액티비티의 현재 상태를 저장하는 것이 필요하다.



Bundle 클래스

- Bundle 클래스는 일종의 Map 자료구조를 구현한 클래스이다. 키(key)와 값(value)을 받아서 객체 안에 저장한다. 키는 문자열로 되어 있다.

예제: 액티비티 상태 저장

- 선택한 상품을 저장하는 애플리케이션 작성

MainActivity.java

```
package kr.co.company.saverestore;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class MainActivity extends AppCompatActivity {
  Button button1, button2;
  TextView text;
  int count = 0;
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      text = (TextView) findViewById(R.id.text);
      button1 = (Button) findViewById(R.id.button1);
      button1.setOnClickListener(new OnClickListener() {
          public void onClick( View v) {
              count++;
              text.setText("현재 개수=" + count);
          }
      });
      button2 = (Button) findViewById(R.id.button2);
      button2.setOnClickListener(new OnClickListener() {
          public void onClick( View v) {
              count--;
              text.setText("현재 개수=" + count);
          }
      });
```

주문 개수를 변경하는 버튼에는 이벤트 처리기를 붙인다. 이벤트 처리기의 onClick() 안에서는 count 멤버 변수의 값을 증가시키거나 감소시킨다. 이러한 클래스 멤버의 값은 개발자가 저장하여야 한다.

상태 저장 및 복구

```
    if (savedInstanceState != null) {
        count = savedInstanceState.getInt("count");
        text.setText("현재 개수=" + count);
    }
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt("count", count);
}
}
```

savedInstanceState가 null이 아니면 getInt()를 사용하여서 저장된 값을 추출한 후에 텍스트 뷰의 내용을 이 값으로 변경한다.

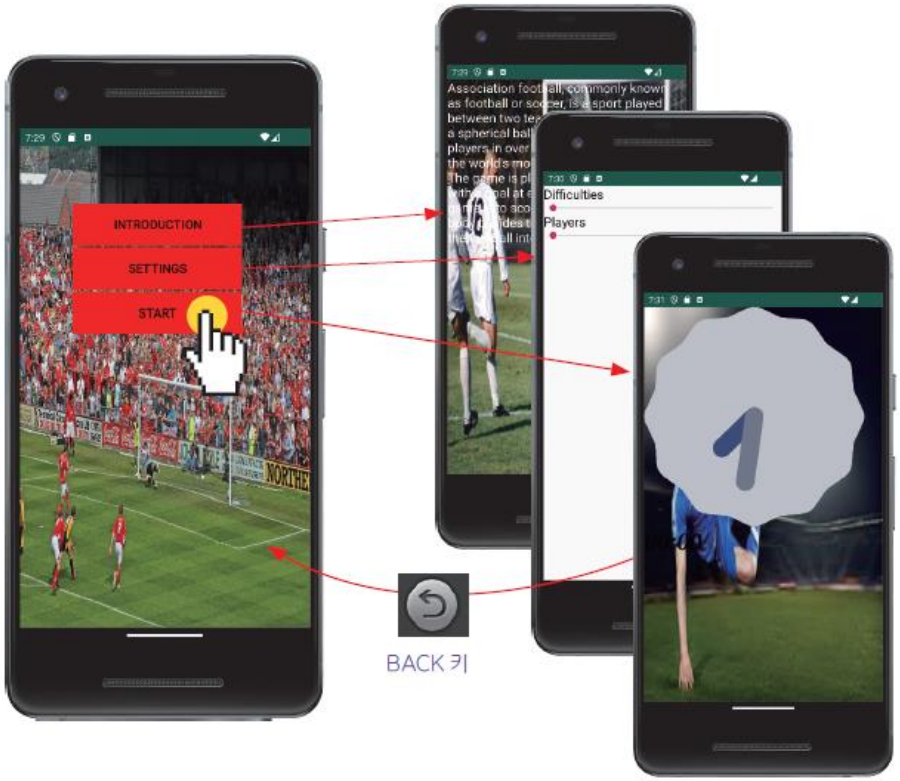
번들 객체인 outState 안에 putInt()를 사용하여서 정수 값인 count를 저장하였다.

실행 결과


만약 사용자가 BACK 버튼을 눌러서 종료시키면
이것은 강제 종료가 아니기 때문에 텍스트 뷰의 내
용이 저장되지 않는다.
에뮬레이터 설정에서
화면이 회전될 수 있도
록 허용하여야 한다.
SaveRestore
주문
피자
스파게티
현재 개수=3
증가
감소

Coding Challenge:
여러 페이지로 구성된 애플리케이션 작성

- 메인 페이지는 하나의 이미지와 3개의 버튼으로 구성되어 있다. 각각의 버튼을 누르면 해당되는 페이지로 이동한다. 이동한 페이지에서 BACK 키를 누르면 메인 페이지로 되돌아온다.



Q & A