

SHOUT · OUR · PASSION · TOGETHER

ODo IT SOPT O

# Android 1차 세미나

SHOUT OUR PASSION TOGETHER  
**SOPT**

**01** 프로젝트  
분석

**02** View  
컴포넌트

**03** Kotlin  
맞보기

**04** 과제

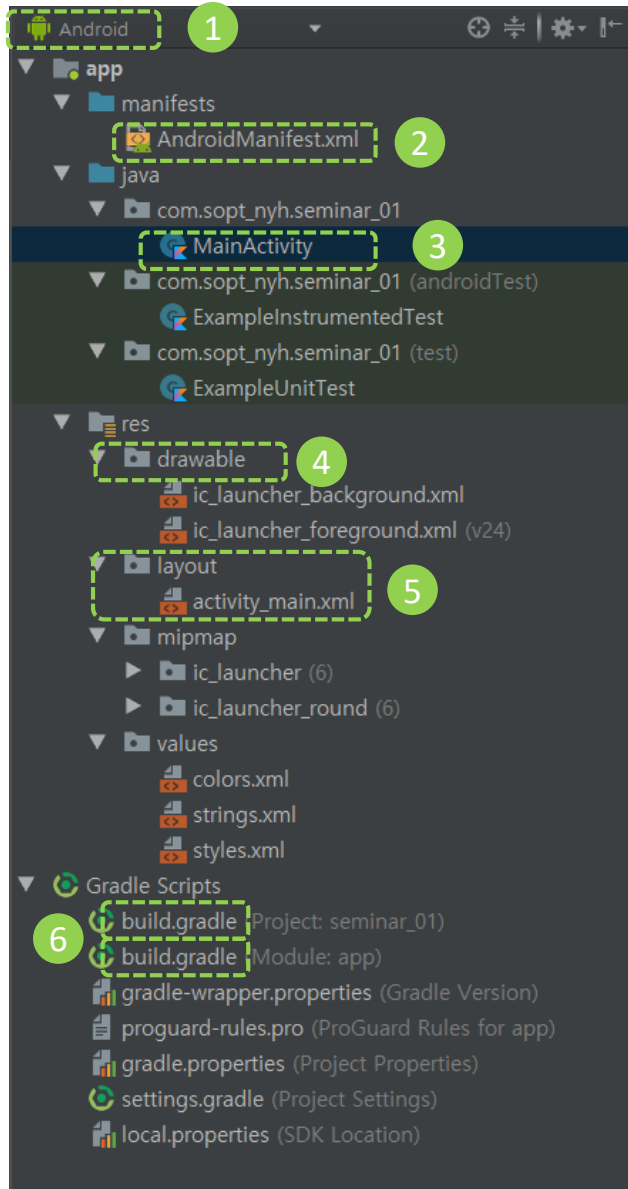
○

01

○

# 프로젝트 분석

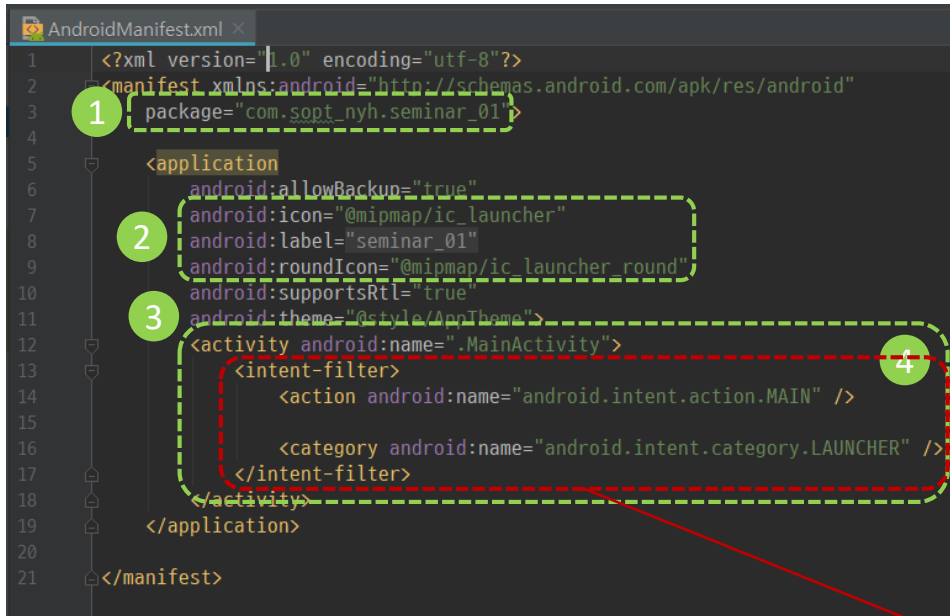
## 01 프로젝트 분석



### 프로젝트 디렉토리 분석하기!

- ① 프로젝트 디렉토리 구조를 어떤 기준으로 보여줄지 선택할 수 있습니다.  
(저는 Android로 설정!)
- ② **AndroidManifest.xml** – 앱에 대한 정보를 가진 파일  
(앱 패키지명, 이름, 아이콘, 각종 권한, Activity 등록 등)
- ③ **MainActivity** – **Programmatically Coding**
- ④ **drawable** – 화면에 그릴 수 있는 리소스들  
(상태 변경을 포함 다양한 xml, 비트맵 사진(.png / .jpg / .gif) 등)
- ⑤ **layout** – Activity, Fragment등 UI를 구성해야 할 layout을 정의  
(쉽게 말해서, 그냥 뷰 짜는 파일들 모아둔 디렉토리!)
- ⑥ **build.gradle**
  - Project 모든 모듈에 적용되는 빌드 구성
  - 특정 모듈(**android app module**)의 빌드 구성

# AndroidManifest.xml



① 패키지 명(추후 고유한 앱 id 생성에 영향을 미쳐요)

② icon - 앱 아이콘 이미지 설정

label - 앱 이름 설정

round Icon - 둥근 앱 아이콘 이미지 설정

③ theme - 앱 전역에서 공통적으로 적용할 테마

④ Activity 등록

빨간 박스의 intent-filter → action 과 category는 MainActivity가 App을 키면 가장 먼저 띄워지도록 하는 역할을 합니다!

## AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sopt_nyh.seminar_01">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="seminar_01"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
        </activity>

        <activity android:name=".SubActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

App실행 시 처음 시작하는 Activity를 바꾸려면?

1. SubActivity라는 액티비티를 하나 추가했습니다!
2. 그리고 이전 피피티 장에 있는 빨간 박스 안 옵션을 SubActivity의 옵션으로 옮겨봤어요!
3. 그러면 MainActivity가 아닌 SubActivity가 앱 실행 시 처음으로 띄워집니다!

권한 등록과 같은 기술은 세미나 하면서 경험 해봅시다!

### MainActivity.kt

Programmatically Coding !!!

```
1 package com.sopt.nyh.seminar_01
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
13
```

### activity\_main.xml

뷰 쉽게 짤 수 있도록 돕는 xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     tools:context=".MainActivity">
7
8     <TextView
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:text="Hello World!"
12        app:layout_constraintBottom_toBottomOf="parent"
13        app:layout_constraintLeft_toLeftOf="parent"
14        app:layout_constraintRight_toRightOf="parent"
15        app:layout_constraintTop_toTopOf="parent" />
16
17 </android.support.constraint.ConstraintLayout>
```

액티비티(화면) 하나는 하나의 layout을 가집니다!  
(액티비티 = 화면)

### 모듈 단위 build.gradle 파일

```
MainActivity.kt × activity_main.xml × app × seminar_01 × activity_sub.xml × SubActivity.kt × AndroidManifest.xml ×
1  apply plugin: 'com.android.application'
2
3  apply plugin: 'kotlin-android'
4
5  apply plugin: 'kotlin-android-extensions'
6
7  android {
8      ① compileSdkVersion 27
9      defaultConfig {
10         applicationId "com.sopt_nyh.seminar_01"
11         minSdkVersion 26
12         targetSdkVersion 27
13         versionCode 1
14         versionName "1.0"
15         testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
16     }
17     buildTypes {
18         release {
19             minifyEnabled false
20             proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
21         }
22     }
23 }
24
25 ② dependencies {
26     implementation fileTree(dir: 'libs', include: ['*.jar'])
27     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
28     implementation 'com.android.support:appcompat-v7:27.1.1'
29     implementation 'com.android.support.constraint:constraint-layout:1.1.3'
30     testImplementation 'junit:junit:4.12'
31     androidTestImplementation 'com.android.support.test:runner:1.0.2'
32     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
33 }
34
```

- ① - app 고유 id  
- 앱 실행에 필요한 최소한의 버전  
- 버전관리  
- 타겟 안드로이드 버전

- ② 외부 모듈 명시하는 곳  
(외부 라이브러리들 이곳에 명시)



### 실습2 외부 라이브러리 등록하기

#### ① 등록할 라이브러리

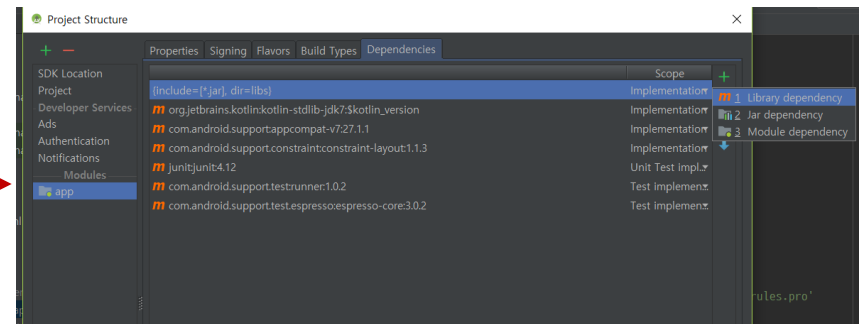
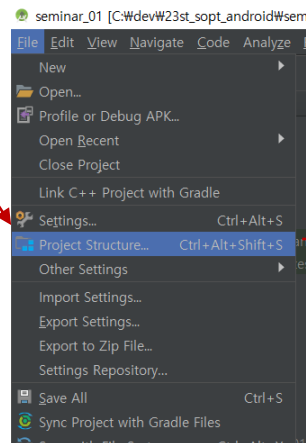
- kotlin을 위한 다재 다능 라이브러리 Anko
- material design이 적용된 view 라이브러리

#### Build.gradle 파일

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version'  
    implementation 'com.android.support:appcompat-v7:27.1.1'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
}
```

#### ② 라이브러리를 등록하는 2가지 방법

- 모듈 단위 Build.gradle에 직접 추가
- Android Studio 메뉴를 통해 추가



## 01 프로젝트 분석

### 라이브러리 등록 방법 1 - 복사

**Anko** : <https://github.com/Kotlin/anko>

#### Using Anko

##### Gradle-based project

Anko has a meta-dependency which plugs in all available features (including Commons, Layouts, SQLite) into your project at once:

```
dependencies {  
    implementation "org.jetbrains.anko:anko:$anko_version"  
}
```

복사!

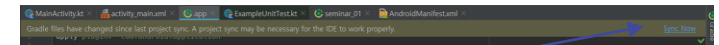
Make sure that you have the `$anko_version` settled in your gradle file at the project level:

```
ext.anko_version='0.10.6'
```

```
dependencies {  
    implementation fileTree(include: ['*.jar'], dir: 'libs')  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
    implementation 'com.android.support:appcompat-v7:27.1.1'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
    implementation 'com.android.support:design:27.1.1'  
    implementation "org.jetbrains.anko:anko:0.10.6"  
}
```

Build.gradle 파일

붙여넣기

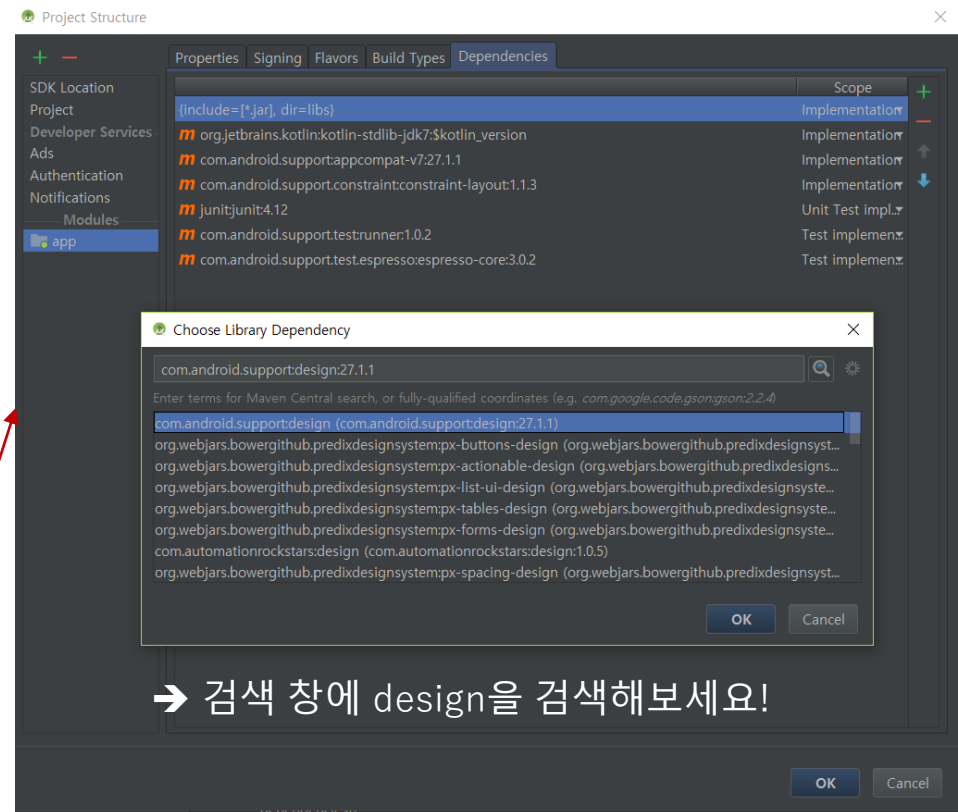
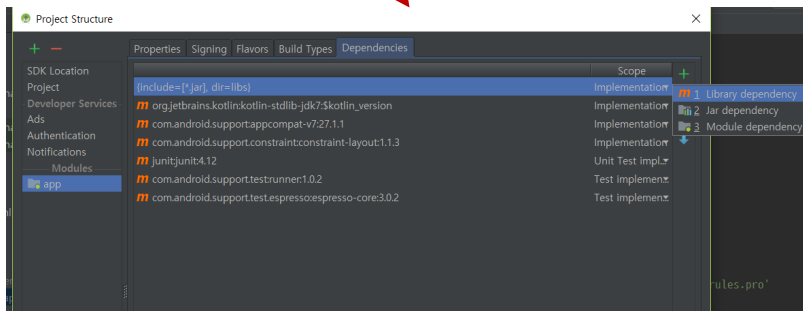
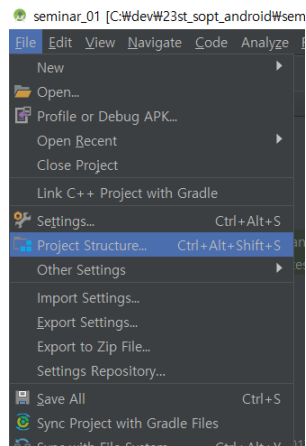


위에 뜨는 메시지에서  
동기화 버튼 클릭!

## 01 프로젝트 분석

### 라이브러리 등록 방법 2 - 안드로이드 스튜디오 메뉴 통해서!

material design이 적용된 view 라이브러리 찾기





02



# View 컴포넌트 맛보기

### View란? 눈에 보이는 모든 구성 요소!

- Button, TextView, ImageView, EditText 등  
→ **Widget**
- LinearLayout, RelativeLayout 등  
→ view(widget, view group)들을 둘러싼 **View Group**

**view들을 구성하여 화면에 보이는 UI를 만드는게 우리가 할 일!**

## 02 View 컴포넌트 맛보기

### View란? 눈에 보이는 모든 구성 요소!

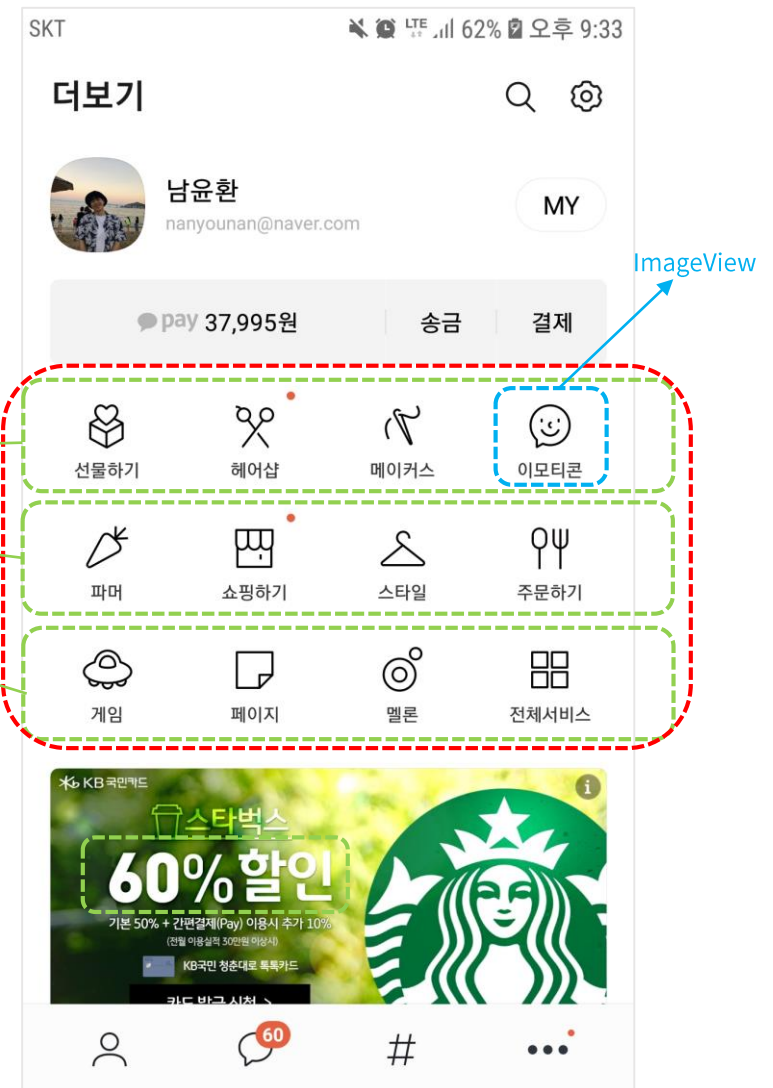
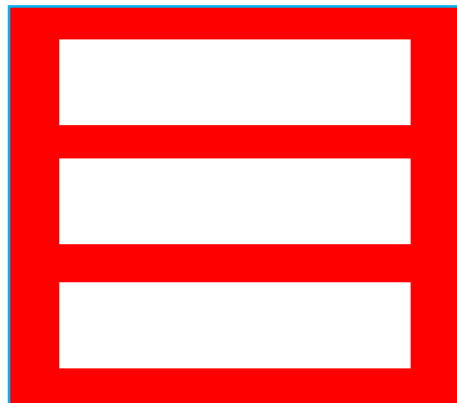
- Button, TextView, ImageView, EditText 등  
→ **Widget**
- LinearLayout, RelativeLayout 등  
→ view(widget, view group) 들을 둘러싼 **View Group**



# 자주 사용되는 ViewGroup

## 1. LinearLayout

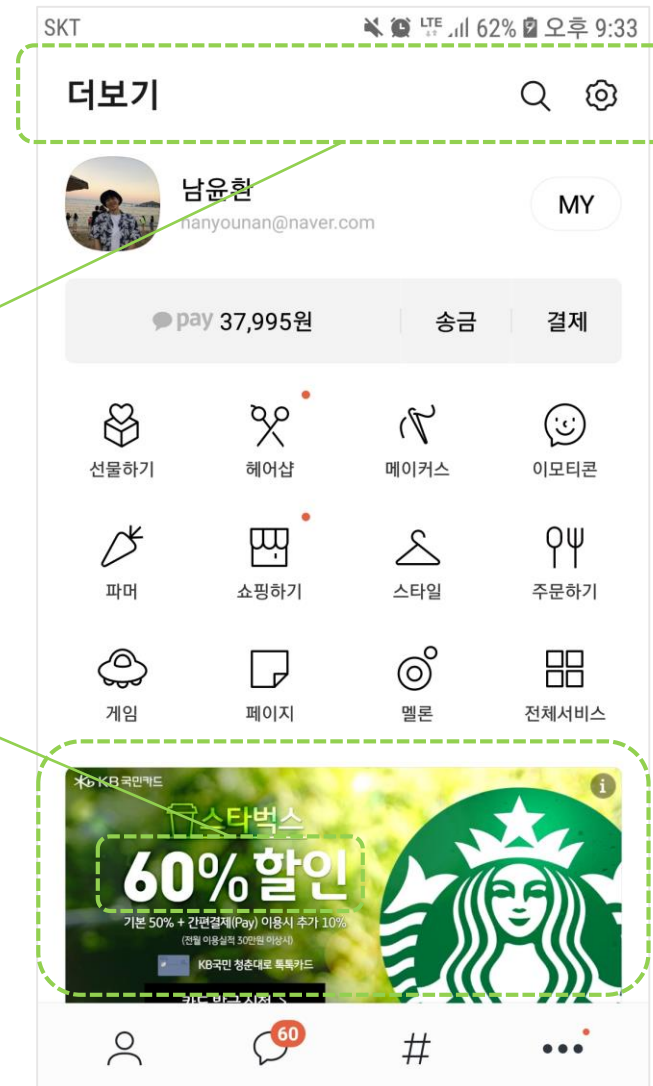
- 내부 View(위젯, viewGroup)을 **선형으로 쌓는** 배치
- orientation 옵션을 통해 세로인지 가로인지 명시
  - horizontal
  - vertical
- 일정하게 줄지어진 뷰를 짤 때 유용



# 자주 사용되는 ViewGroup

## 2. RelativeLayout

- 내부 View(위젯, viewGroup)을 **상대적**으로 배치
- 특정 뷰를 기준으로 다른 뷰를 배치
- 일정하게 정렬되지 않는 뷰를 짤 때 유용
- 겹쳐지는 뷰를 만들 때 유용



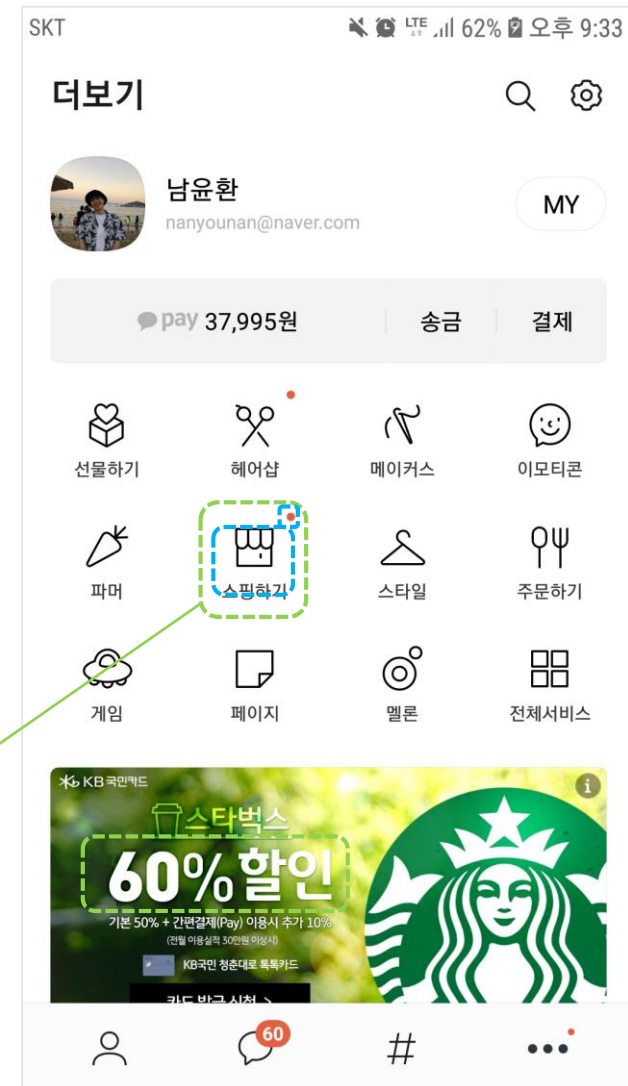


# 자주 사용되는 ViewGroup

## 2. RelativeLayout

- 내부 View(위젯, viewGroup)을 **상대적**으로 배치
- 특정 뷰를 기준으로 다른 뷰를 배치
- 일정하게 정렬되지 않는 뷰를 짤 때 유용
- 겹쳐지는 뷰를 만들 때 유용

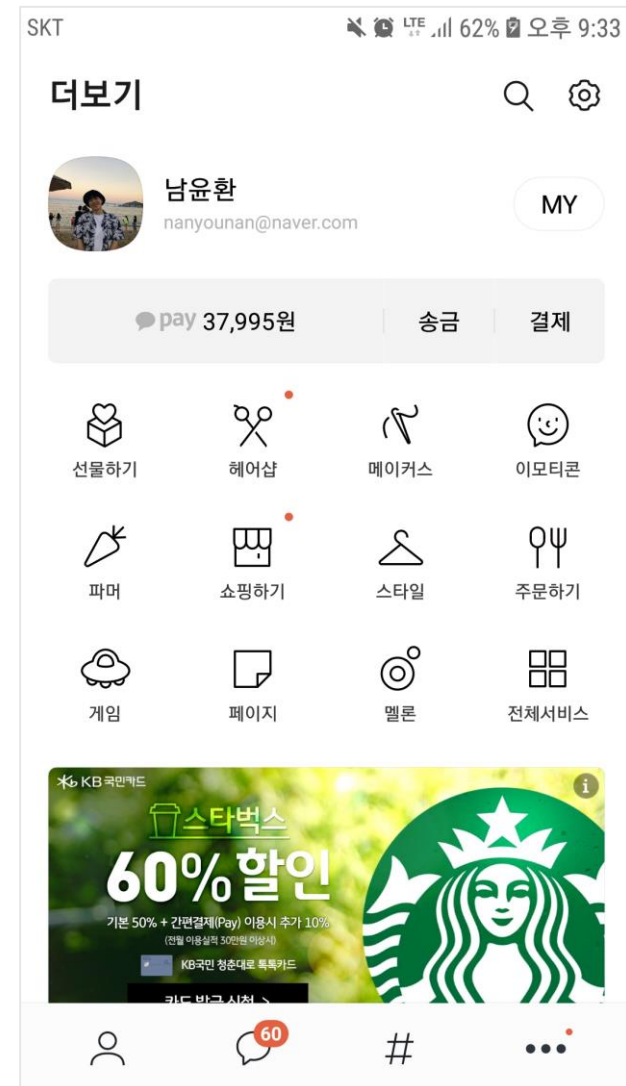
RelativeLayout



# 자주 사용되는 기본 widget view

- ① `<TextView/>` : 문자열을 보여줄 widget view
- ② `<ImageView/>` : 리소스 이미지를 보여줄 widget view
- ③ `<EditText/>` : 문자열 입력 가능한 widget view
- ④ `<Button/>` :

(모든 다른 뷰로 버튼을 대체 구현할 수 있지만 세미나 때 이용하기 위해서 알아봅시다!)



# 자주 쓰는 view 옵션 - 공통

### ① 필수 옵션

- layout\_width: 뷰의 가로 길이
- layout\_height: 뷰의 세로 길이

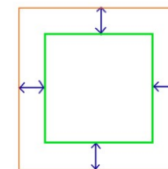
### ② 필수 옵션의 특수 할당 값

- wrap\_content: 내용물에 맞춰서 크기 할당
- match\_parent: 부모 뷰의 크기에 꽉 채워서 크기 할당

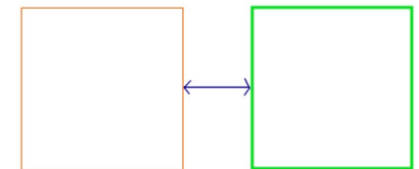
### ③ 많이 자주 쓰는 공통 옵션

- 1) id: 뷰를 참조하기 위한 값
- 2) margin: 다른 뷰와의 간격
- 3) padding: 본인 뷰의 바깥 틀과 내용물의 간격
- 4) visibility
  - gone - 자리 차지 없이 안보임
  - visible - 보임
  - invisible - gone과는 다르게 자리를 차지하지만 안보임

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/tv_main_act_title"
        android:text="안녕하세요!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```



**Padding**

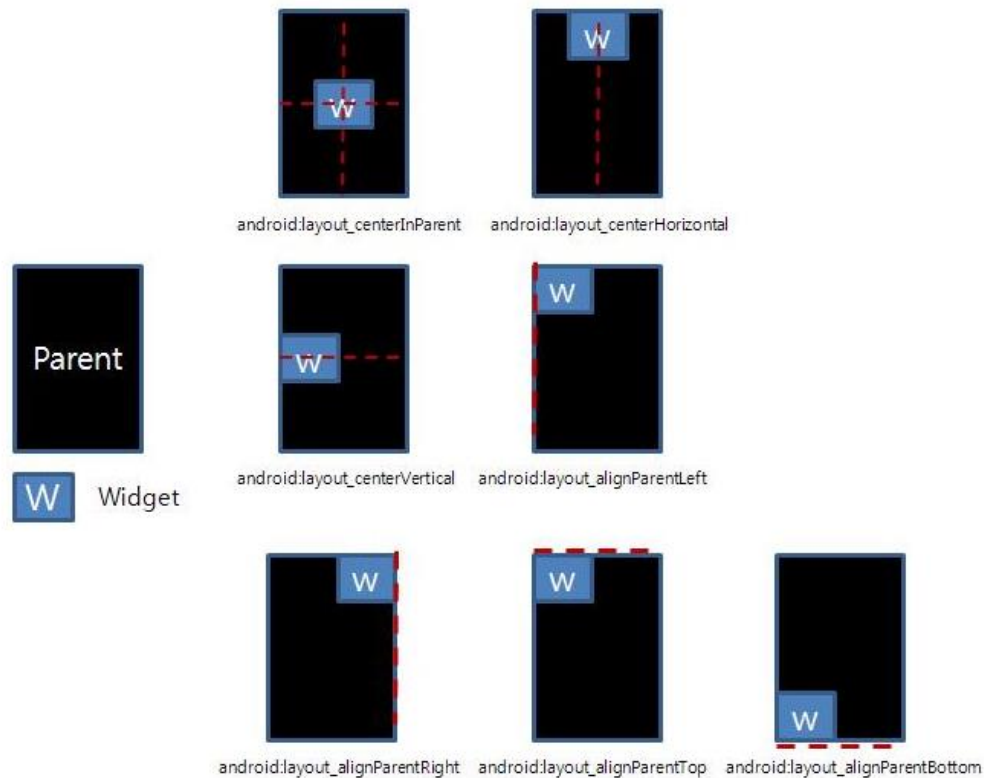


**Margin**

# 자주 쓰는 view 옵션 - RelativeLayout

## 1. RelativeLayout 내부 뷰가 가질 수 있는 옵션

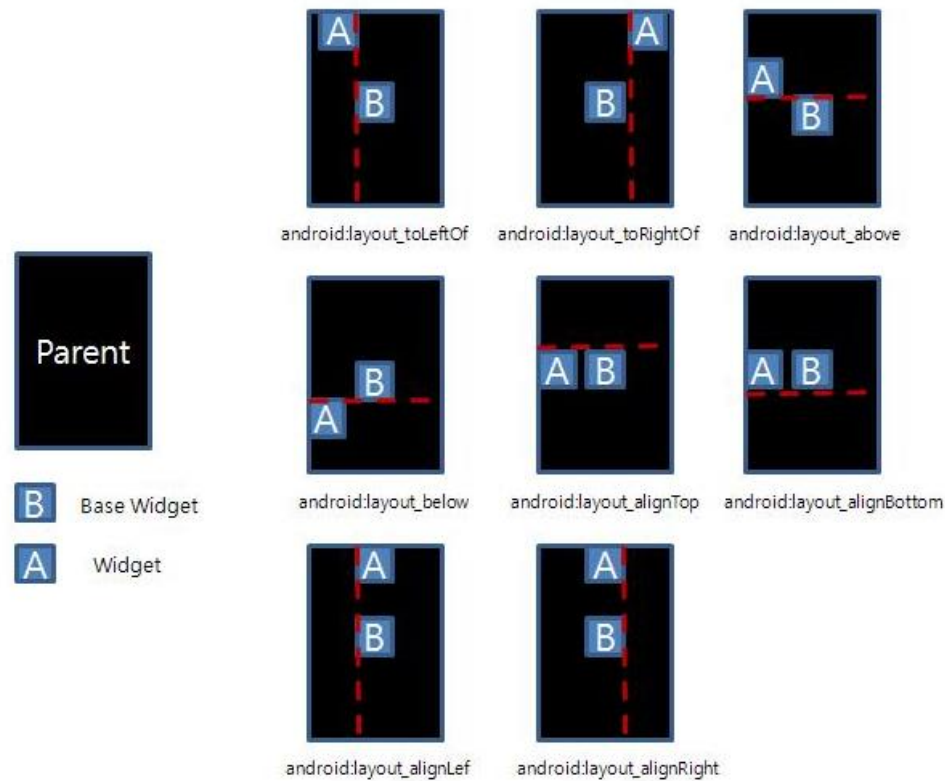
- RelativeLayout 안의 Content 배치



# 자주 쓰는 view 옵션 - RelativeLayout

## 2. RelativeLayout 내부 뷰끼리 가질 수 있는 옵션, view A의 id와 view B의 id를 통해 상대적 위치 결정

◦ Widget 들간의 배치



# 자주 쓰는 view 옵션 - LinearLayout

## 1. LinearLayout

- orientation : 가로/세로 뷰가 쌓이는 기준 설정
- layout\_weight : view가 차지하는 비중 설정
- gravity : 본인 뷰 내부의 뷰 위치 설정
- layout\_gravity : 부모 뷰의 기준으로 위치 설정



```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="100dp">
    <RelativeLayout
        android:background="#151395"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:text="0,0"
            android:layout_centerInParent="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </RelativeLayout>
    <RelativeLayout
        android:background="#151351"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:text="0,1"
            android:layout_centerInParent="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </RelativeLayout>
</LinearLayout>
```

### 실습3 - 구현하기!

- ① 유용 사이트 색상 선택 material design color :  
<https://material.io/tools/color/#!//?view.left=0&view.right=0>

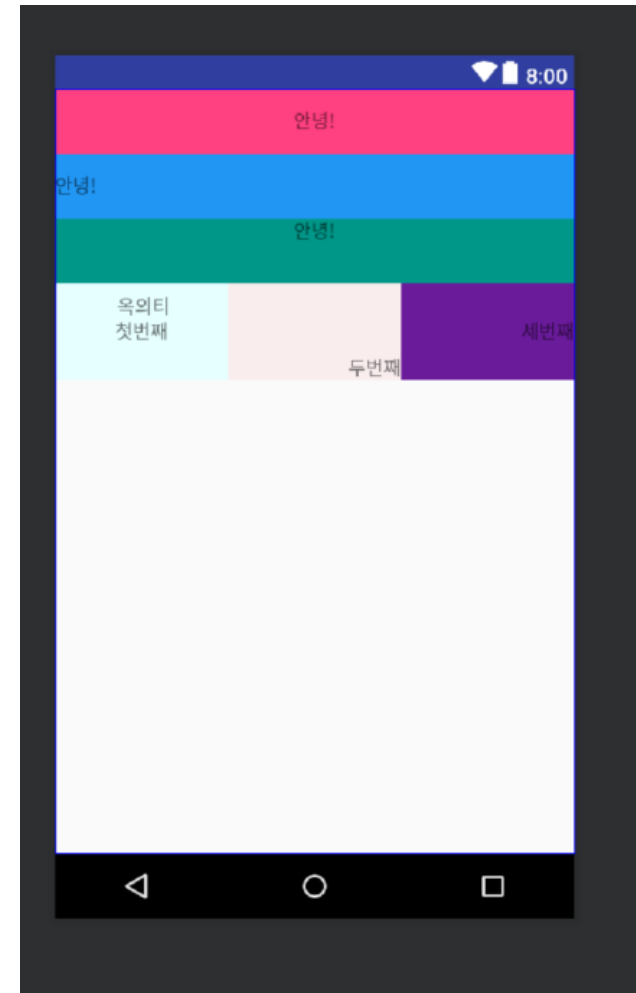
tip. 타이틀/액션 바 없애는 방법

res/values/styles.xml에 아래 두줄 코드 추가

```
<item name="windowActionBar">false</item>
<item name="windowNoTitle">true</item>
```

```
<resources>
    <!-- 테마로 지정된 스타일에!!! -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>

        <!-- 두줄 추가 -->
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>
</resources>
```





03

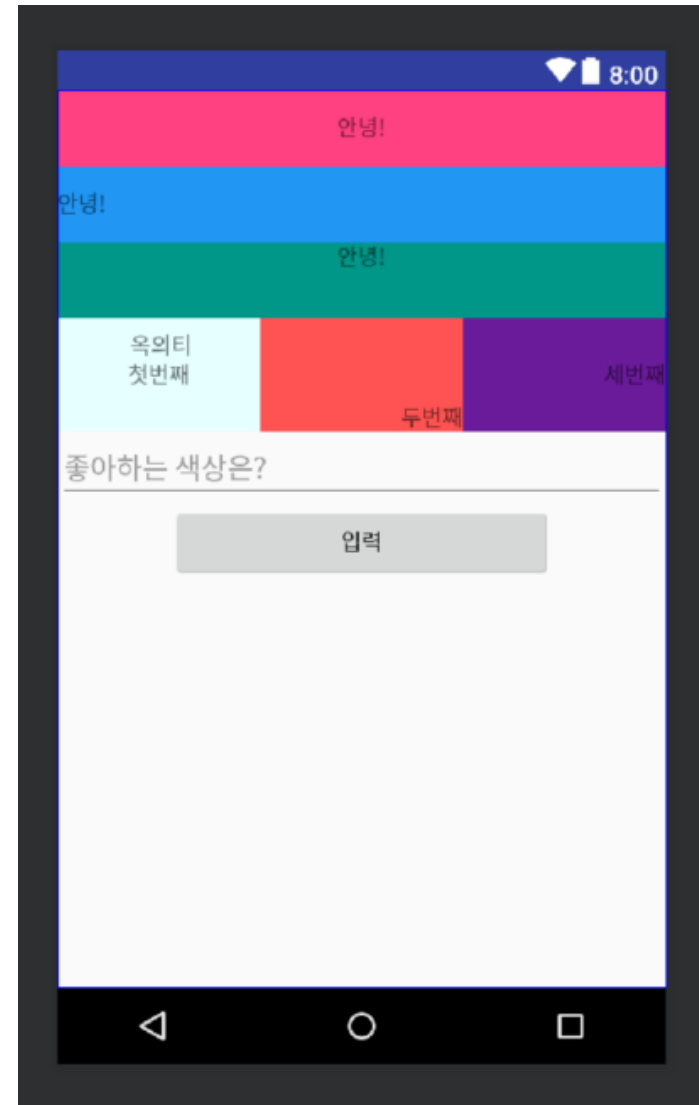


# kotlin 맛보기



### 실습 5

- ① view 클릭 리스너
- ② 변수 선언
- ③ editText의 문자열 받기
- ④ toast



### 03 kotlin 맛보기

## 이미지 만드는 방법

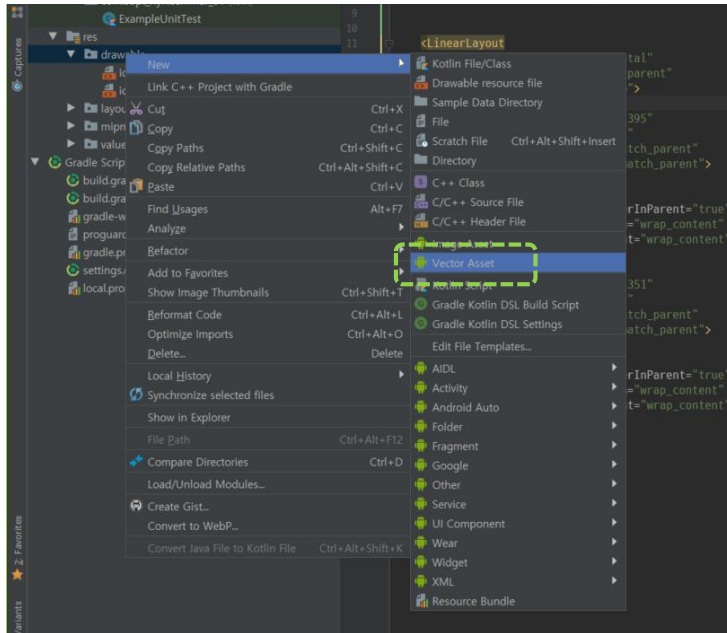
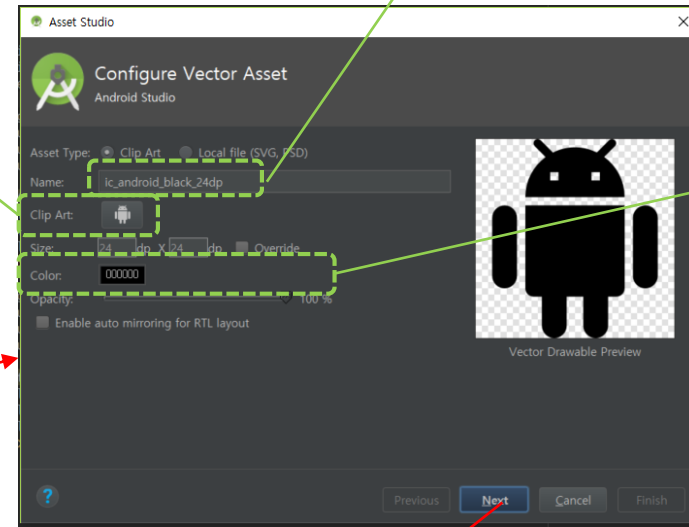


사진 선택



파일명 설정

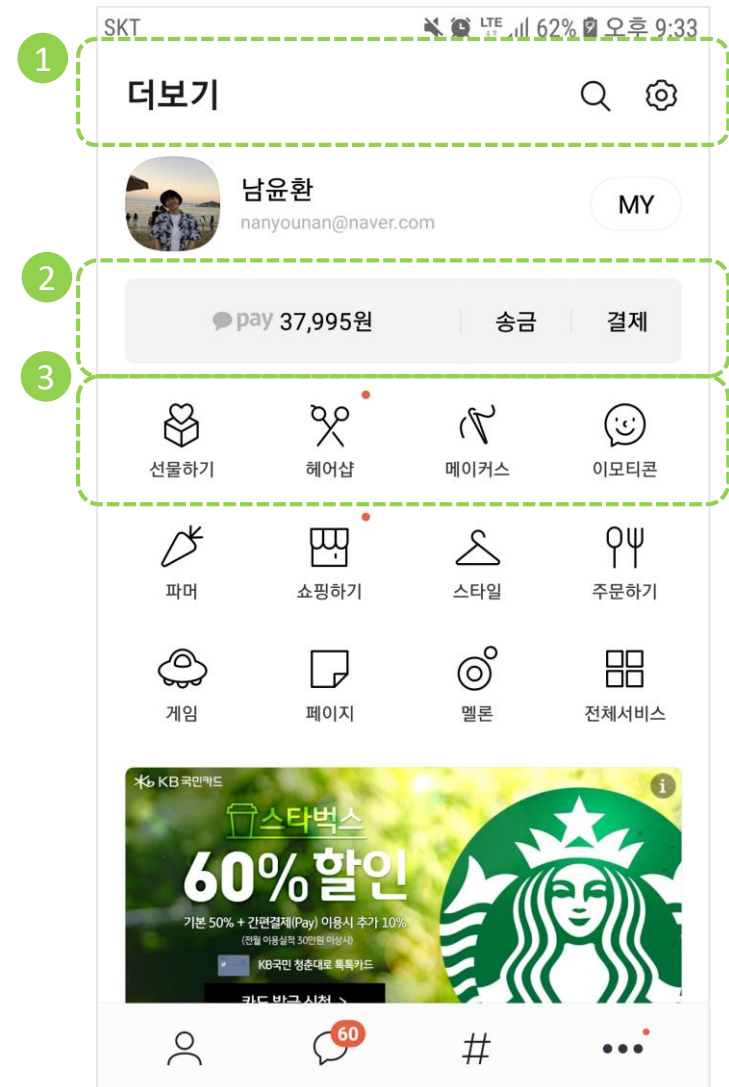
색상 선택

Next 클릭 후 다음 화면에서 Finish 클릭하면  
res/drawable에 해당 이미지 생성

### 실습 4

- ① 구현하기
- ② 구현하기
- ③ 구현하기

이미지는 아무거나 쓰기!!!





04

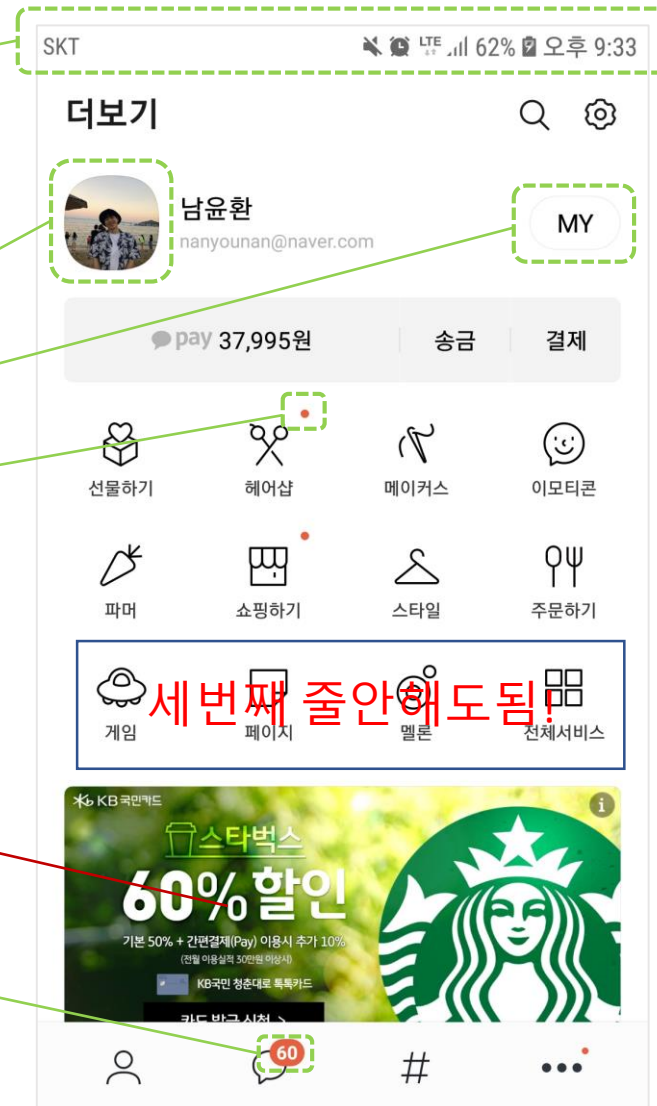


# 과 제 1

## 과제 1 - 모든 View 구현하기

- ① 상태 바 무시
- ② 모든 이미지는 본인이 넣고 싶은 이미지 넣기  
(background나 src를 통해 색상 넣어도 무방)
- ③ 모든 둥근 모서리 처리 무시, 그냥 사각형으로!  
(추후 세미나를 통해 학습)
- ④ 회색 outline 무시 (추후 세미나를 통해 학습)
- ⑤ 구현하기
- ⑥ ImageView로 구현, 사진은 아무거나 넣기!  
색상으로 채워도 무방
- ⑦ 무시하기
- ⑧ 텍스트 굵기 크기는 본인이 하고 싶은 사이즈로!

간격, 이미지 등 디자인 가이드 라인이 없으므로 완벽하게 구현하지 않아도 되며 **본인이 할 수 있는 만큼 구현해보기**



S H O U T · O U R · P A S S I O N · T O G E T H E R

ODo IT SOPT O

THANK U

PPT 디자인 한승미 세미나 자료 배다슬

SHOUT OUR PASSION TOGETHER  
**SOPT**