



[스파르타코딩클럽] 안드로이드 앱개발 기초반 - 5주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

1. 유형 검사 앱을 완성한다.
 - (1) 입력받은 값을 계산해 결과를 표시한다.
2. 유저 행동 데이터 수집을 위한 파이어베이스 애널리틱스를 사용해본다.
3. 애드몹을 이용해 앱에 광고를 추가한다.
4. 유형 검사 앱을 배포한다.
 - (1) 앱 설치파일인 apk, aab를 생성한다.
 - (2) 구글 플레이 개발자 계정을 등록한다.

[목차]

01. 5주차 오늘 배울 것

- 02. 검사 앱 결과 화면 완성하기 (1)
- 03. 검사 앱 결과 화면 완성하기 (2)
- 04. 검사 앱 결과 화면 완성하기 (3)
- 05. 유저 행동 데이터 수집하기 (1)
- 06. 유저 행동 데이터 수집하기 (2)
- 07. 유형 검사 앱에 광고 붙이기 (1)
- 08. 유형 검사 앱에 광고 붙이기 (2)
- 09. 유형 검사 앱 배포하기 (1)
- 10. 유형 검사 앱 배포하기 (2)
- 11. 5주차 끝 & 숙제 설명



모든 토글을 열고 닫는 단축키

Windows : **Ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

01. 5주차 오늘 배울 것

- ▼ 1) 5주차: 구글 애드몹, 파이어베이스 애널리틱스, apk 생성 및 배포



힘들게 만든 앱 수익까지 낼 수 있다면 정말 좋겠죠? 구글애드몹(Google AdMob)을 이용하여 광고를 붙이는 방법을 다뤄봅니다.

만든 앱을 자랑하기위해 매번 친구를 불러 USB에 연결해 매번 설치해 줄 수는 없잖아요!

누구나 내가 만든 앱을 쉽게 설치할 수 있도록, APK 파일을 만들고 구글플레이에 등록을 해보겠습니다.

+

설치한 사람들이 앱을 어떻게 쓰는지 알 수 없어서 너무 답답하지 않나요?

내가 만든 앱을 몇 명이 어떻게 사용하는지 분석하기 위한 파이어베이스 애널리틱스(Firebase Analytics)도 다뤄볼게요!

02. 검사 앱 결과 화면 완성하기 (1)

▼ 2) 검사 결과 화면 추가하기

(1) 검사 결과 액티비티를 추가해볼까요?



1. Project 창에서 app 폴더 마우스 오른쪽 클릭
2. New > Activity > Empty Activity
3. 액티비티 이름 'ResultActivity' 설정
4. Finish!

(2) MainActivity의 clickSubmit 함수안에 기존 Toast 코드는 지우고 새로운 화면을 시작하는 코드로 변경해볼게요. ResultActivity를 실행할 때 SeekBar의 값을 함께 전달할게요.



토스트를 띄우는 코드는 제거해주세요.

▼ [코드스니펫] - 유형 검사 결과 화면 띄우기

```
var intent = Intent(this, ResultActivity::class.java)
intent.putExtra("TypeKey", answer1)
startActivity(intent)
```

```
//MainActivity.kt
...
fun clickSubmit(view: View) {
    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var intent = Intent(this, ResultActivity::class.java)
    intent.putExtra("TypeKey", answer1)
    startActivity(intent)
}
...
}
```



Intent가 빨간색으로 Error가 나타나면 뭘 해야할까요?
네, 맞아요! Alt+Enter 를 눌러 Import 해주세요.



한번 프로젝트를 실행해서 새로운 액티비티가 나오는지 확인해볼까요?

(3) 액티비티에 전달받은 값이 잘 넘어오는지 확인해볼까요? ResultActivity에서 전달받은 값을 토스트 메시지로 띄워 볼게요.



이번에는 getIntentExtra를 이용해서 값을 받아볼게요. getStringExtra 대신 getIntentExtra를 사용하는 이유는 MainActivity에서 putExtra에 전달넣어준 값이 'Int'형태라서 getIntentExtra를 사용해요. getIntentExtra는 getStringExtra와 다르게 만약 key에 해당하는 값이 없는 경우 저장할 값을 명시해줘야 해요.

▼ [코드스니펫] - 유형 검사 결과 화면 토스트 확인하기

```
var typeKey = intent.getIntExtra("TypeKey", -1)
Toast.makeText(this, "TypeKey: ${typeKey}", Toast.LENGTH_SHORT).show()
```

```
class ResultActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_result)

        var typeKey = intent.getIntExtra("TypeKey", -1)
        Toast.makeText(this, "TypeKey: ${typeKey}", Toast.LENGTH_SHORT).show()
    }
}
```

▼ 3) 선택된 값을 이용해 검사 결과 계산하기



사용자들이 앱에서 질문에 대한 답을 계산하여 I / E, S / N 과 같이 나만의 알파벳으로 만들어볼게요.



해당 내용은 개발적인 지식이 아닌 유형 검사 앱 예시를 위한 기획적인 내용이에요. 그냥 이런 것을 만들 계획 이다 정도로만 이해하시면 됩니다.

강의에서는 간단하게 질문 4개를 이용해 S(Simple) / F(Full) 와 P(Perfect) / E(Efficient) 로 구분할거예요.

질문 2개는 S와 F를 구분하고 다른 질문 두개는 P와 E로 분류하고 PS(완벽추구 심플리스트), ES(효율적인 미니멀리스트), PF(준비된 완벽주의자), EF(일석이조 유니콘)으로 결과를 보여주는 작업을 해볼게요.



S/F, P/E 타입 구분 방식에는 답이 없어요. 나만의 계산방식을 가져가면 됩니다. 강의에서는 1번과 4번 질문 이 S/F 구분을 위한 질문이고, 2번과 3번은 P/E 구분을 위한 질문이에요.

1번과 4번 답의 평균이 5점에 가까우면 S 타입이고, 2번과 3번 답의 평균이 5점에 가까우면 P 타입으로 계산을 할거예요.

(1) 선택한 답을 계산하기 전에 질문을 추가할게요. 레이아웃 템플릿을 이용해 activity_main.xml 구성을 변경해주세요.

▼ [코드스니펫] - 유형 검사 화면 질문지 템플릿

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="32dp">

            <TextView
                android:id="@+id/title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
                android:gravity="center"
                android:padding="8dp"
                android:text="나는 어떤 유형일까?"
                android:textSize="20sp" />

            <EditText
                android:id="@+id/nameEditText"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
                android:hint="이름을 입력해주세요." />

            <androidx.constraintlayout.widget.ConstraintLayout
                android:id="@+id/layoutQ1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:paddingTop="16dp"
                android:paddingBottom="16dp">

                <TextView
                    android:id="@+id/titleQ1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="내 책상엔 물건이 많은 편이다."
                    android:textSize="16sp"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="parent" />

                <TextView
                    android:id="@+id/disagreeQ1"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="16dp"
                    android:text="그렇지 않다"
                    app:layout_constraintBottom_toBottomOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@id/titleQ1" />

                <SeekBar
                    android:id="@+id/barQ1"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_margin="4dp"
                    android:max="5"
                    android:min="1"
                    android:progress="3"
                    app:layout_constraintBottom_toBottomOf="@id/disagreeQ1"
                    app:layout_constraintEnd_toStartOf="@id/agreeQ1"
                    app:layout_constraintStart_toEndOf="@id/disagreeQ1"
                    app:layout_constraintTop_toTopOf="@id/disagreeQ1" />

            <TextView

```

```

        android:id="@+id/agreeQ1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="그렇다"
        app:layout_constraintBottom_toBottomOf="@id/disagreeQ1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@id/disagreeQ1" />

</androidx.constraintlayout.widget.ConstraintLayout>

<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/layoutQ2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp">

    <TextView
        android:id="@+id/titleQ2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ctrl+c ctrl+v보다 직접 쓰는게 마음이 편하다."
        android:textSize="16sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/disagreeQ2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="그렇지 않다"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/titleQ2" />

    <SeekBar
        android:id="@+id/barQ2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="4dp"
        android:max="5"
        android:min="1"
        android:progress="3"
        app:layout_constraintBottom_toBottomOf="@id/disagreeQ2"
        app:layout_constraintEnd_toStartOf="@id/agreeQ2"
        app:layout_constraintStart_toEndOf="@id/disagreeQ2"
        app:layout_constraintTop_toTopOf="@id/disagreeQ2" />

    <TextView
        android:id="@+id/agreeQ2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="그렇다"
        app:layout_constraintBottom_toBottomOf="@id/disagreeQ2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@id/disagreeQ2" />

</androidx.constraintlayout.widget.ConstraintLayout>

<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/layoutQ3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp">

    <TextView
        android:id="@+id/titleQ3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="단축키가 마우스보다 편하다."
        android:textSize="16sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

```

```

<TextView
    android:id="@+id/disagreeQ3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="그렇지 않다"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/titleQ3" />

<SeekBar
    android:id="@+id/barQ3"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    android:max="5"
    android:min="1"
    android:progress="3"
    app:layout_constraintBottom_toBottomOf="@id/disagreeQ3"
    app:layout_constraintEnd_toStartOf="@id/agreeQ3"
    app:layout_constraintStart_toEndOf="@id/disagreeQ3"
    app:layout_constraintTop_toTopOf="@id/disagreeQ3" />

<TextView
    android:id="@+id/agreeQ3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="그렇다"
    app:layout_constraintBottom_toBottomOf="@id/disagreeQ3"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="@id/disagreeQ3" />

</androidx.constraintlayout.widget.ConstraintLayout>
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/layoutQ4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp">

    <TextView
        android:id="@+id/titleQ4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="나중에 쓰일 것 같으면 일단 갖고 있다."
        android:textSize="16sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/disagreeQ4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="그렇지 않다"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/titleQ4" />

    <SeekBar
        android:id="@+id/barQ4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="4dp"
        android:max="5"
        android:min="1"
        android:progress="3"
        app:layout_constraintBottom_toBottomOf="@id/disagreeQ4"
        app:layout_constraintEnd_toStartOf="@id/agreeQ4"
        app:layout_constraintStart_toEndOf="@id/disagreeQ4"
        app:layout_constraintTop_toTopOf="@id/disagreeQ4" />

    <TextView
        android:id="@+id/agreeQ4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```

```

        android:text="그렇다"
        app:layout_constraintBottom_toBottomOf="@id/disagreeQ4"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="@id/disagreeQ4" />

</androidx.constraintlayout.widget.ConstraintLayout>

<Button
    android:id="@+id/submitType"
    android:layout_width="match_parent"
    android:layout_height="64dp"
    android:layout_marginTop="16dp"
    android:onClick="clickSubmit"
    android:text="나의 유형 확인하기" />
</LinearLayout>
</androidx.core.widget.NestedScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>

```

(2) barQ1~barQ4에 들어온 값을 변수에 저장해주세요.

☞ 기존 clickSubmit 에서 startActivity 코드는 잠시 주석처리하고 시작할게요. 주석처리하고 싶은 범위를 선택하고 `ctrl + /` (Mac은 `Cmd + /`) 를 누르면 주석처리를 쉽게할 수 있어요.

☞ 변수 barQ1과 answer1 을 생성하는 코드를 복사하여 barQ1~barQ4, answer1~answer4까지 만들어 볼게요.

```

//MainActivity.kt
...
fun clickSubmit(view: View) {
    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress
    var barQ4: SeekBar = findViewById(R.id.barQ4)
    var answer4 = barQ4.progress

    // var intent = Intent(this, ResultActivity::class.java)
    // intent.putExtra("TypeKey", answer1)
    // startActivity(intent)
}
...
}

```

(3) answer1~answer4까지 저장되었는지 토스트 메시지를 이용해 확인해볼게요.

☞ 문자열을 표현하는 " " 안에 문자열 템플릿(String Templates)인 \${ } 를 이용하여 변수를 이용해 편하게 문자열을 만들 수 있어요.

```

//MainActivity.kt
...
fun clickSubmit(view: View) {
    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)

```

```

        var answer3 = barQ3.progress
        var barQ4: SeekBar = findViewById(R.id.barQ4)
        var answer4 = barQ4.progress
        Toast.makeText(this, "${answer1}, ${answer2}, ${answer3}, ${answer4}", Toast.LENGTH_SHORT).show()

//        var intent = Intent(this, ResultActivity::class.java)
//        intent.putExtra("TypeKey", answer1)
//        startActivity(intent)
    }
    ...
}

```

👉 프로젝트를 실행해서 SeekBar를 조절하며 값이 제대로 토스트 메시지에 순서대로 나오는지 확인해주세요.

(4) answer1~answer4 변수들을 계산해서 S/F, P/E 타입을 구분하여 토스트 메시지로 확인해볼게요.

```

//MainActivity.kt
...
fun clickSubmit(view: View) {
    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress
    var barQ4: SeekBar = findViewById(R.id.barQ4)
    var answer4 = barQ4.progress
    Toast.makeText(this, "${answer1}, ${answer2}, ${answer3}, ${answer4}", Toast.LENGTH_SHORT).show()

    var scoreSF = (answer1 + answer4) / 2
    var scorePE = (answer2 + answer3) / 2
    Toast.makeText(this, "${scoreSF}, ${scorePE}", Toast.LENGTH_SHORT).show()
    //    var intent = Intent(this, ResultActivity::class.java)
    //    intent.putExtra("TypeKey", answer1)
    //    startActivity(intent)
}
...
}

```

👉 코틀린에서 Int를 Int로 나누게 되는 경우 결과 값은 Int로 저장되며 반올림이 아니라 버림이 됩니다. 예를 들어 3 / 2 를 하면 결과값은 1이 돼요. 강의에서는 버림값을 이용하기에 별다른 처리를 하지 않았어요.

👉 만약, 소숫점을 관리하고 싶으신 경우 3.0 / 2.0 의 형태로 Float 타입을 활용해야해요. 저장된 변수가 위의 코드처럼 Int 인 경우에는 toFloat() 함수를 이용하면 됩니다.
ex) num.toFloat() / 2.0

Int와 Float의 차이는 초보자일 때 자주하는 실수이니 조심하세요!

▼ [코드스니펫] - 유형 검사 결과 계산하기 clickSubmit (완성)

```

fun clickSubmit(view: View) {
    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress

```



```

var barQ4: SeekBar = findViewById(R.id.barQ4)
var answer4 = barQ4.progress

var scoreSF = (answer1 + answer4) / 2
var scorePE = (answer2 + answer3) / 2
Toast.makeText(this, "${scoreSF}, ${scorePE}", Toast.LENGTH_SHORT).show()
// var intent = Intent(this, ResultActivity::class.java)
// intent.putExtra("TypeKey", answer1)
// startActivity(intent)
}

```

03. 검사 앱 결과 화면 완성하기 (2)

▼ 4) 계산된 값을 활용해 검사결과 보여주기

(1) 계산된 값인 scoreSF 값과 scorePE 값을 putExtra()를 이용하여 ResultActivity에 데이터를 전달해볼게요.

☞ 주석처리해놓은 ResultActivity를 시작하는데 필요한 코드를 되돌릴게요.
주석처리한 코드 라인 혹은 코드 블럭에서 ctrl + / (Mac: Cmd + /)를 누르면 주석이 다시 풀려요.

☞ putExtra()로 TypeSF에 scoreSF값을 TypePE에 scorePE값을 담아볼게요.

```

//MainActivity.kt
...
fun clickSubmit(view: View) {
    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress
    var barQ4: SeekBar = findViewById(R.id.barQ4)
    var answer4 = barQ4.progress

    var scoreSF = (answer1 + answer4) / 2
    var scorePE = (answer2 + answer3) / 2
    Toast.makeText(this, "${scoreSF}, ${scorePE}", Toast.LENGTH_SHORT).show()
    var intent = Intent(this, ResultActivity::class.java)
    intent.putExtra("TypeKey", answer1)
    intent.putExtra("TypeSF", scoreSF)
    intent.putExtra("TypePE", scorePE)
    startActivity(intent)
}
...
}

```

(2) 사용자의 이름도 받아서 ResultActivity에 데이터를 전달해볼게요.

☞ EditText의 text 값은 String 타입이 아니다보니 toString을 이용해서 String 타입으로 변환해줘야해요.
만약 text 값을 넘기게 되면 null이 나오게 됩니다.
나중에 비슷한 문제를 겪으시면 'EditText getStringExtra null' 과 같은 검색어로 검색하시면 돼요.

```

//MainActivity.kt
...
fun clickSubmit(view: View) {

```

```

var editText: EditText = findViewById(R.id.nameEditText)
var userName = editText.text.toString()

var barQ1: SeekBar = findViewById(R.id.barQ1)
var answer1 = barQ1.progress
var barQ2: SeekBar = findViewById(R.id.barQ2)
var answer2 = barQ2.progress
var barQ3: SeekBar = findViewById(R.id.barQ3)
var answer3 = barQ3.progress
var barQ4: SeekBar = findViewById(R.id.barQ4)
var answer4 = barQ4.progress

var scoreSF = (answer1 + answer4) / 2
var scorePE = (answer2 + answer3) / 2
var intent = Intent(this, ResultActivity::class.java)
intent.putExtra("UserName", userName)
intent.putExtra("TypeSF", scoreSF)
intent.putExtra("TypePE", scorePE)
startActivity(intent)
}
...
}

```

▼ [코드스니펫] - 유형 검사 결과 보여주기 clickSubmit (완성)

```

fun clickSubmit(view: View) {
    var editText: EditText = findViewById(R.id.nameEditText)
    var userName = editText.text.toString()

    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress
    var barQ4: SeekBar = findViewById(R.id.barQ4)
    var answer4 = barQ4.progress

    var scoreSF = (answer1 + answer4) / 2
    var scorePE = (answer2 + answer3) / 2
    var intent = Intent(this, ResultActivity::class.java)
    intent.putExtra("UserName", userName)
    intent.putExtra("TypeSF", scoreSF)
    intent.putExtra("TypePE", scorePE)
    startActivity(intent)
}

```

(3) ResultActivity에서 계산된 값을 활용해서 토스트 메시지를 띄워볼게요.



ResultActivity.kt를 수정하지 않고 앱을 실행하게 되면 ResultActivity에서 나오는 토스트 메시지는 -1이 나오게 돼요.
MainActivity에서 인텐트에 "TypeKey"값을 보내주지 않아서 없으면 기본 값으로 설정한 -1이 나오게 된답니다.

```

class ResultActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_result)

        var typeKey = intent.getIntExtra("TypeKey", -1)
        Toast.makeText(this, "TypeKey: ${typeKey}", Toast.LENGTH_SHORT).show()
        var userName = intent.getStringExtra("UserName")
        var scoreSF = intent.getIntExtra("TypeSF", -1)
        var scorePE = intent.getIntExtra("TypePE", -1)
        if (scoreSF < 3) {
            if (scorePE < 3) {

```

```

        Toast.makeText(this, "${userName} FE", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "${userName} FP", Toast.LENGTH_SHORT).show()
    }
} else {
    if (scorePE < 3) {
        Toast.makeText(this, "${userName} SE", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "${userName} SP", Toast.LENGTH_SHORT).show()
    }
}
}
}
}

```

▼ [코드스니펫] - 유형 검사 결과 보여주기 ResultActivity 토스트메세지 (완성)

```

var userName = intent.getStringExtra("UserName")
var scoreSF = intent.getIntExtra("TypeSF", -1)
var scorePE = intent.getIntExtra("TypePE", -1)
if (scoreSF < 3) {
    if (scorePE < 3) {
        Toast.makeText(this, "${userName} FE", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "${userName} FP", Toast.LENGTH_SHORT).show()
    }
} else {
    if (scorePE < 3) {
        Toast.makeText(this, "${userName} SE", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "${userName} SP", Toast.LENGTH_SHORT).show()
    }
}
}
}

```

(4) 이제 거의 다왔어요! 토스트메시지 말고 TextView를 이용해 값이 표시되도록 변경할게요.



activity_result.xml 레이아웃에 id 값이 `resultText` 인 TextView를 하나 추가해주세요.

▼ [코드스니펫] - 유형 검사 화면 질문지 템플릿

```

<TextView
    android:id="@+id/resultText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```



주석을 이용해 알파벳 타입을 기록해두면 나중에 코드만보고도 이게 어떤 유형을 표현하려고 했던것인지 파악하기 쉬울 수 있어요.

주석은 이런식으로 미사용 코드를 관리하는 것 뿐만아니라 내용을 기록하는 용도로도 사용한답니다.

```

class ResultActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_result)

        var resultTextView: TextView = findViewById(R.id.resultText)
        var userName = intent.getStringExtra("UserName")
        var scoreSF = intent.getIntExtra("TypeSF", -1)
        var scorePE = intent.getIntExtra("TypePE", -1)
    }
}

```

```

        if (scoreSF < 3) {
            if (scorePE < 3) {
                //FE 타입
                resultTextView.text = "${userName}님의 유형은 일석이조 유니콘"
            } else {
                //FP 타입
                resultTextView.text = "${userName}님의 유형은 준비된 완벽주의자"
            }
        } else {
            if (scorePE < 3) {
                //SE 타입
                resultTextView.text = "${userName}님의 유형은 효율적인 미니멀리스트"
            } else {
                //SP 타입
                resultTextView.text = "${userName}님의 유형은 완벽추구 심플리스트"
            }
        }
    }
}
}

```

▼ [코드스니펫] - 유형 검사 결과 보여주기 ResultActivity (완성)

```

class ResultActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_result)

        var resultTextView: TextView = findViewById(R.id.resultText)
        var userName = intent.getStringExtra("UserName")
        var scoreSF = intent.getIntExtra("TypeSF", -1)
        var scorePE = intent.getIntExtra("TypePE", -1)
        if (scoreSF < 3) {
            if (scorePE < 3) {
                //FE 타입
                resultTextView.text = "${userName}님의 유형은 일석이조 유니콘"
            } else {
                //FP 타입
                resultTextView.text = "${userName}님의 유형은 준비된 완벽주의자"
            }
        } else {
            if (scorePE < 3) {
                //SE 타입
                resultTextView.text = "${userName}님의 유형은 효율적인 미니멀리스트"
            } else {
                //SP 타입
                resultTextView.text = "${userName}님의 유형은 완벽추구 심플리스트"
            }
        }
    }
}
}

```

04. 검사 앱 결과 화면 완성하기 (3)

▼ 5) 기존 선택한 값 초기화하기

- 조금 더 완벽한 동작을 위해 마지막 하나 더! '나의 유형 확인하기' 버튼을 눌러 결과화면으로 이동하게 되면 기존의 선택된 값을 초기화 해볼게요.



SeekBar의 값은 progress에 값을 설정해주면 됩니다. EditText는 TextView와는 값을 설정하는 방식이 조금 달라요. editText.text = "" 형태가 아니라 editText.setText("")를 해줘야해요.

```

//MainActivity.kt
...

```

```

fun clickSubmit(view: View) {
    var editText: EditText = findViewById(R.id.nameEditText)
    var userName = editText.text.toString()

    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress
    var barQ4: SeekBar = findViewById(R.id.barQ4)
    var answer4 = barQ4.progress

    var scoreSF = (answer1 + answer4) / 2
    var scorePE = (answer2 + answer3) / 2
    var intent = Intent(this, ResultActivity::class.java)
    intent.putExtra("UserName", userName)
    intent.putExtra("TypeSF", scoreSF)
    intent.putExtra("TypePE", scorePE)
    startActivity(intent)

    //값 초기화
    barQ1.progress = 3
    barQ2.progress = 3
    barQ3.progress = 3
    barQ4.progress = 3
    editText.setText("")
}
...
}

```

▼ [코드스니펫] - 유형 검사 결과 보여주기 clickSubmit (완성)

```

fun clickSubmit(view: View) {
    var editText: EditText = findViewById(R.id.nameEditText)
    var userName = editText.text.toString()

    var barQ1: SeekBar = findViewById(R.id.barQ1)
    var answer1 = barQ1.progress
    var barQ2: SeekBar = findViewById(R.id.barQ2)
    var answer2 = barQ2.progress
    var barQ3: SeekBar = findViewById(R.id.barQ3)
    var answer3 = barQ3.progress
    var barQ4: SeekBar = findViewById(R.id.barQ4)
    var answer4 = barQ4.progress

    var scoreSF = (answer1 + answer4) / 2
    var scorePE = (answer2 + answer3) / 2
    var intent = Intent(this, ResultActivity::class.java)
    intent.putExtra("UserName", userName)
    intent.putExtra("TypeSF", scoreSF)
    intent.putExtra("TypePE", scorePE)
    startActivity(intent)

    //값 초기화
    barQ1.progress = 3
    barQ2.progress = 3
    barQ3.progress = 3
    barQ4.progress = 3
    editText.setText("")
}

```



유형 검사 앱을 완성했어요!

질문을 추가하거나 계산방식을 고민해서 나만의 유형 검사 계산 방식을 만들고, 레이아웃을 다듬어 앱의 디자인을 변경해서 나만의 앱으로 탄생시켜주세요.

05. 유저 행동 데이터 수집하기 (1)

▼ 6) 프로젝트에 파이어베이스 추가하기

👉 파이어베이스 애널리틱스를 사용하게 되면 사용자들이 앱을 언제, 어떻게 사용하고 있는지 파악하게 도와줘요. 파이어베이스에 앱을 연결만하면 기본적인 데이터들은 자동으로 데이터 수집이 시작되어 파이어베이스 콘솔을 활용해 데이터를 분석, 조회할 수 있어요.
파이어베이스는 일정 사용량까지는 무료로 사용할 수 있으며, 애널리틱스 외에도 크래시리틱스 (Crashlytics), 클라우드 메시징, 동적링크 같은 편리한 기능들이 있습니다.
자세한 소개는 파이어베이스 소개페이지([링크](#))를 참고해주세요.

(1) Firebase Console([링크](#))로 이동하여 프로젝트 추가를 클릭한 후 프로젝트 이름을 입력해주세요.

(2) '이 프로젝트에서 Google 애널리틱스 사용 설정'을 활성화 해둔 상태로 '계속' 버튼을 눌러주세요.

(3) Google 애널리틱스 계정을 사용하거나 새 계정을 만들어서 선택해주세요.

(4) 앱에 Firebase를 추가하여 시작하기 화면에서 Android 아이콘을 클릭해주세요.

(5) 등록하려는 앱의 패키지 이름을 적고 앱 이름을 등록해주세요.

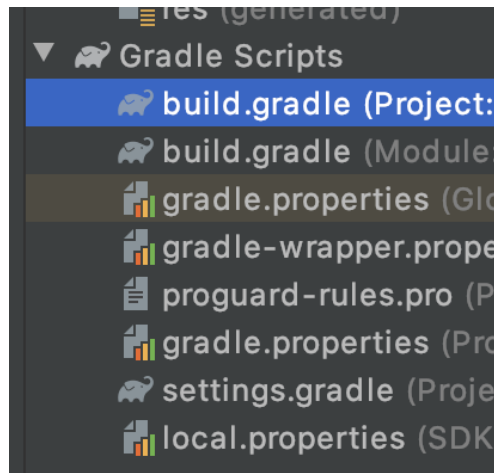
👉 앱 패키지 이름이 기억나지 않으신다면 앱 패키지 이름은 매니페스트파일에 있어요.
AndroidManifest.xml 파일을 열어서 manifest element의 attribute로 있는 package 값을 확인해주세요.

(6) 'google-services.json 다운로드' 버튼을 눌러 파일을 다운받고 app 폴더로 파일을 이동시켜주세요.

👉 파이어베이스의 가이드를 따라하셔도 되지만 프로젝트 보기 전환하지 않고도 충분히 가능합니다.

(7) Gradle Scripts를 열어 build.gradle (Project: ~~) 파일을 열어주세요.

👉 Module이 적혀있는 build.gradle 파일이 아닌 Project가 적혀있는 build.gradle 파일을 열어주셔야 해요.



☞ buildscript 아래의 repositories에 google()이 있는지 확인하고 없다면 추가해주세요.
동일하게 allprojects 아래의 repositories에 google()이 있는지 확인하고 없다면 추가해주세요.

☞ [코드스니펫] - build.gradle classpath 추가에 있는 코드를 복사해서 dependencies에 추가해주세요.

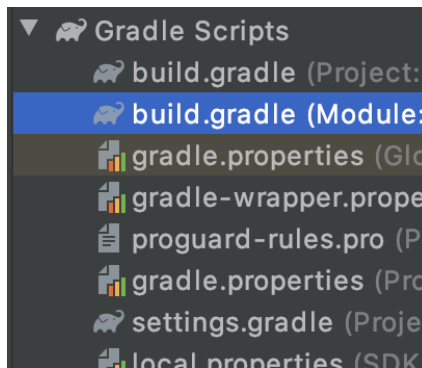
▼ [코드스니펫] - build.gradle classpath 추가

```
'com.google.gms:google-services:4.3.5'
```

```
buildscript {
    ...
    repositories {
        google()
        ...
    }
    dependencies {
        ...
        classpath 'com.google.gms:google-services:4.3.5'
    }
}
...
```

(8) 이번에는 Gradle Scripts안의 Module 수준의 build.gradle (Module: ~~) 파일을 열어주세요.

☞ 기존 Project 레벨 build.gradle이 아닌 Module이 적혀있는 build.gradle 파일에서 설정을 진행해요.



👉 build.gradle 상태에 따라서 설정하는 방식이 달라지니 주의해서 따라와주세요. plugins { 형태로 시작하는 경우는 [8-1]처럼 코드를 추가하시고, apply plugin: 형태로 시작하는 경우는 [8-2]처럼 코드를 추가해주세요.

▼ [코드스니펫] - google-services plugin 추가

```
'com.google.gms.google-services'
```

```
[8-1]
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'
    ...
}

android {
    ...
}
```

```
[8-2]
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'

android {
    ...
}
```

👉 [코드스니펫] - dependencies Firebase BoM 추가 코드를 dependencies에 추가해주세요.

▼ [코드스니펫] - dependencies Firebase BoM 추가

```
implementation platform('com.google.firebase:firebase-bom:26.8.0')
implementation 'com.google.firebase:firebase-analytics-ktx'
```

```
...
android {
    ...
}
```



```
dependencies {
    ...
    implementation platform('com.google.firebase:firebase-bom:26.8.0')
    implementation 'com.google.firebase:firebase-analytics-ktx'
}
```

(9) 추가를 완료하셨다면 네트워크 상태를 확인하고 안드로이드 스튜디오 표시줄에 있는 'Sync Now' 버튼을 눌러주세요.

☞ Sync Now를 누르게 되면 build.gradle에 정의된 대로 필요한 외부 파일을 가져와 다운로드 받는 작업이 진행돼요. 파이어베이스 관련된 설정을 추가해서 파이어베이스를 가져와 동기화하는 작업이 진행된답니다.

!! Sync가 진행되는 동안에는 안드로이드 스튜디오를 종료하거나 네트워크가 원활하지 않은 경우 추가로 설정이 필요해질 수 있으니 주의해주세요.

(10) 로딩이 완료되고 'BUILD SUCCESSFUL' 단어가 보인다면 완료! 프로젝트를 실행해서 앱을 사용해 보세요.

(11) 데이터가 수집되기를 기다렸다가 파이어베이스 콘솔의 '애널리틱스 > Dashborad'로 이동하면 데이터 수집정보를 확인할 수 있어요.

☞ 설정하는데 그동안 배운 내용이 하나도 나오지 않아서 어렵고 두려우신가요?
괜찮습니다! build.gradle은 한번 추가하거나 설정하면 수정할 일이 자주있는 파일은 아니에요. 그리고 build.gradle 설정할 일이 생기는 경우는 파이어베이스 문서처럼 대부분 가이드문서가 잘 되어 있습니다. 설정할 일이 생긴다면 가이드문서를 믿고 잘 따라가주세요.

06. 유저 행동 데이터 수집하기 (2)

▼ 7) 이벤트 기록 및 사용자 속성 설정하기

(1) MainActivity에서 '나의 유형 확인하기' 버튼을 누르면 어떤 답변들을 선택했는지 정보를 담아서 'click_submit'으로 로깅해볼게요. 공식 문서에서 가이드하는 방식을 확인하고 싶으시면 [링크](#)를 확인해주세요.

▼ [코드스니펫] - 메인 액티비티의 이벤트 기록하기

```
FirebaseAnalytics.getInstance(this).logEvent("click_submit") {
    param("answer1", answer1.toLong())
    param("answer2", answer2.toLong())
    param("answer3", answer3.toLong())
    param("answer4", answer4.toLong())
}
```

```
//MainActivity.kt
...
fun clickSubmit(view: View) {
    ...
    var intent = Intent(this, ResultActivity::class.java)
    intent.putExtra("UserName", userName)
```

```

intent.putExtra("TypeSF", scoreSF)
intent.putExtra("TypePE", scorePE)
startActivity(intent)


FirebaseAnalytics.getInstance(this).logEvent("click_submit") {
    param("answer1", answer1.toLong())
    param("answer2", answer2.toLong())
    param("answer3", answer3.toLong())
    param("answer4", answer4.toLong())
}
...
}
...
}

```

!! 코드스니펫을 이용해 붙여넣은 경우 import가 제대로 되지 않아 param 에 빨간색 글자가 나타나고 alt+Enter로도 해결이 되지않을 수 있어요.
 그런 경우 FirebaseAnalytics.getInstance(this) 코드에서 .logEvent 를 작성하실 때 추천에 나오는 logEvent(name: String, crossline block: ~~)을 선택해주세요.
 만약, 위의 방법으로도 해결이 안된다면 **[코드스니펫] - 메인 액티비티의 이벤트 기록하기 import 에러해결** 을 package 와 import 위치 사이에 추가해주세요.

▼ [코드스니펫] - 메인 액티비티의 이벤트 기록하기 import 에러 해결

```
import com.google.firebase.analytics.ktx.logEvent
```

 파이어베이스 애널리틱스는 앱을 실행할 때와 같이 몇 가지 이벤트들을 자동으로 로깅해주고 있어요. 자동으로 기록되는 이벤트는 자동으로 수집되는 이벤트([링크](#))에서 확인할 수 있어요.

(2) FirebaseAnalytics 를 사용하기 위해서는 매니페스트에 권한을 추가해야해요. AndroidManifest.xml에 권한을 추가할게요.

▼ [코드스니펫] - 파이어베이스 애널리틱스 사용을 위한 권한 추가

```

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />

```

```

//AndroidManifest.xml
<manifest ...>

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <application ...>
        ...
    </application>
</manifest>

```

(3) 검사 결과 조회화면에서 정해진 타입을 사용자 속성으로 추가해볼게요.

▼ [코드스니펫] - 유형 검사 앱 사용자 속성 추가하기

```
FirebaseAnalytics.getInstance(this).setUserProperty("user_test_type", "FE")
```

```

class ResultActivity : AppCompatActivity() {
    ...
    if (scoreSF < 3) {
        if (scorePE < 3) {
            //FE 타입
            FirebaseAnalytics.getInstance(this).setUserProperty("user_test_type", "FE")
            resultTextView.text = "${userName}님의 유형은 일석이조 유니콘"
        } else {
            //FP 타입
            FirebaseAnalytics.getInstance(this).setUserProperty("user_test_type", "FP")
            resultTextView.text = "${userName}님의 유형은 준비된 완벽주의자"
        }
    } else {
        if (scorePE < 3) {
            //SE 타입
            FirebaseAnalytics.getInstance(this).setUserProperty("user_test_type", "SE")
            resultTextView.text = "${userName}님의 유형은 효율적인 미니멀리스트"
        } else {
            //SP 타입
            FirebaseAnalytics.getInstance(this).setUserProperty("user_test_type", "SP")
            resultTextView.text = "${userName}님의 유형은 완벽추구 심플리스트"
        }
    }
}
}
}

```

👉 파이어베이스 애널리틱스는 사용자 속성도 자동으로 로깅해주고 있어요. 자동으로 기록되는 속성들은 사전 정의된 사용자 측정기준([링크](#))에서 확인할 수 있어요.

👉 현업에서는 이벤트 로깅에 필요한 이벤트 이름과 속성 값들은 개발자가 정의해서 필요한 데이터를 확인하기도 하지만 주로 기획팀 혹은 데이터 분석팀과 언제 어떤 이름으로 기록할지 논의해서 추가가 돼요.

07. 유형 검사 앱에 광고 붙이기 (1)

▼ 8) 애드몹에 앱 등록하기

👉 애드몹(AdMob)은 단순한 방식으로 앱에 광고를 추가하여 수익을 올릴 수 있어요. 유저 행동 데이터를 수집하기 위해 추가한 파이어베이스를 사용중이라면 더욱 간단한 설정으로 광고를 추가할 수 있어서 애드몹을 사용할거예요.

(1) 애드몹([링크](#))에서 '시작하기' 버튼을 눌러 가입을 진행해 AdMob 계정을 생성해주세요.

(2) 애드몹 홈화면에서 '시작하기' 버튼 혹은 메뉴 '앱 > 앱 추가'를 클릭하여 앱을 추가해주세요. 플랫폼은 'Android'이고 아직 스토어에 등록하지 않았으니 '아니요'를 선택해주세요. 앱 이름을 설정하고 앱을 추가해주세요.

👉 스토어와 연동은 앱 배포하고 구글플레이에 등록하기 항목에서 진행할게요. 구글플레이에 등록되어 있지 않아도 괜찮으니 걱정하지마세요.

(3) 애드몹과 파이어베이스를 연동해볼게요. 방금 등록한 앱을 선택한 뒤 메뉴창의 '앱 > 앱 설정' 으로 이동해주세요. 앱 설정 메뉴 리스트에 '연결된 서비스 > Firebase' 항목을 보시면 '연결 안 됨'으로 표시가 되어있네요. 'FIREBASE에 연결' 버튼을 눌러 연결해주세요.

(4) 패키지 이름을 등록하고 '계속'을 눌러주세요. '연결 옵션'에서 '기존 Firebase 프로젝트 및 기존 Firebase 앱에 연결'을 눌러 파이어베이스에 등록된 앱 프로젝트를 선택해주세요. 프로젝트를 선택하면 앱의 패키지명칭이 나오는데 일치하는지 확인해주시면 좋아요. '연결 옵션'을 설정하였다면 '계속'을 눌러 파이어베이스와 연동을 진행해주세요.

(5) '완료'를 누른 뒤 '앱 설정 > 연결된 서비스 > Firebase' 항목에 프로젝트가 연결된 것을 확인해주시면 연동 완료!

👉 애드몹 초기 설정은 완료되었어요. 이제 앱에 광고를 추가하러 가볼까요!

▼ 9) 프로젝트에 모바일 광고 SDK(Mobile Ads SDK) 추가하기

👉 SDK? 낯선단어인가요? SDK 는 소프트웨어 개발 키트(Software Development Kit)의 약자로 편의를 위해 만들어놓은 툴이라고 생각하시면 돼요. 광고를 온라인에서 받아서 보여주는 것까지 직접 구현할 수도 있겠지만 애드몹은 코드 한두줄만 넣으면 알아서 다 처리해준답니다.

SDK는 마치 요리를 할 때 재료를 따로 구해서 손질할 필요 없이 바로 조리를 시작할 수 있게 미리 만들어 놓은 말키트 같은 존재로 생각하시면 돼요.

유저 행동 로깅을 위해 추가한 파이어베이스도 SDK랍니다. 안드로이드 프로젝트에서는 SDK를 추가하기 위해서 build.gradle을 활용해요.

👉 해당 내용은 파이어베이스 구글 애드몹 추가 가이드([링크](#))를 참고하여 작성되었어요.

- 파이어베이스 애널리틱스 추가했을 때 처럼 프로젝트에 애드몹을 추가해볼게요. 파이어베이스 설정에서 이미 고생했기에 애드몹은 파이어베이스보다 훨씬 수월하니 너무 걱정하지마세요.

(1) AndroidManifest.xml 매니페스트 파일에 애드몹에 대한 정보를 추가해주세요.

👉 meta-data 값을 복사해서 [ADMOB_APP_ID]값을 변경하여 application 항목 안에 추가해주세요.

👉 [ADMOB_APP_ID]는 '애드몹 > 앱 > 앱 설정' 화면에서 볼 수 있는 '앱 ID'값을 복사해서 변경해주세요. 애드몹 앱 ID는 'ca-app-pub-'로 시작하는 값이에요.

ex) 애드몹의 앱 ID `ca-app-pub-xxxxx` 인 경우

```
<meta-data
    android:name="com.google.android.gms.ads.APPLICATION_ID"
    android:value="ca-app-pub-xxxxx"/>
```

▼ [코드스니펫] - 매니페스트 파일 애드몹 메타데이터 추가

```
<meta-data
    android:name="com.google.android.gms.ads.APPLICATION_ID"
```

```
android:value="[ADMOB_APP_ID]"/>
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application ...>
    <meta-data
      android:name="com.google.android.gms.ads.APPLICATION_ID"
      android:value="[ADMOB_APP_ID]"/>
    <activity ...>
    </activity>
  </application>
</manifest>
```

(2) AndroidManifest.xml 매니페스트 파일에 인터넷 사용권한을 추가할게요. 광고를 가져오려면 네트워크가 필요해요. 앱에서 네트워크를 사용하기 위해서는 매니페스트 파일에 권한을 추가해야해요.

▼ [코드스니펫] - 매니페스트 파일 인터넷 사용권한 추가

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

```
<manifest ... >

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

  <application ...>
    ...
  </application>
</manifest>
```

(3) 모바일 광고 SDK를 프로젝트에 추가해볼까요? build.gradle (Module: ~) 파일에 코드 한줄만 추가해주세요.

▼ [코드스니펫] - 모바일 광고 SDK 추가

```
implementation 'com.google.android.gms:play-services-ads:19.8.0'
```

```
...
dependencies {
  ...
  implementation platform('com.google.firebase:firebase-bom:26.8.0')
  implementation 'com.google.firebase:firebase-analytics-ktx'
  implementation 'com.google.android.gms:play-services-ads:19.8.0'
}
```

(4) 'Sync Now'를 눌러 SDK 파일을 다운받아 프로젝트에 동기화해주세요.

(5) 앱을 처음 실행 시 나타나는 액티비티에서 모바일 광고 사용을 위해 초기화 코드를 추가할게요.



모바일 광고 초기화 코드는 앱 시작 후 1회만 진행하면 돼요. 유형 검사 앱에서 앱 시작 시 처음 코드가 동작하는 액티비티는 MainActivity라서 MainActivity에 추가할게요.

▼ [코드스니펫] - 모바일 광고 초기화 하기

```
MobileAds.initialize(this)
```

```
//MainActivity.kt
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        MobileAds.initialize(this)
    }
    ...
}
```

- 앱에 광고를 붙일 준비는 모두 완료되었어요. 추가한 코드에 문제가 없는지 프로젝트를 한번 실행해볼까요?

!! 앱이 비정상종료되면서 'The Google Mobile Ads SDK was initialized incorrectly'와 같은 메시지가 나타나 시나요?

<meta-data> 값을 잘못 추가하면 발생하는 에러메시지예요. <meta-data>가 매니페스트에 제대로 등록됐는지 확인해주시고, 등록한 ADMOB_APP_ID를 다시한번 확인해주세요.

👉 공식 문서를 보실 때, 영어도 괜찮으시면 공식 문서를 영어로 확인하시는 것을 권장드려요. 번역이 늦게 되어 한국어와 영어간의 가이드 방식에 차이가 있을 수 있답니다.

08. 유형 검사 앱에 광고 붙이기 (2)

▼ 10) 배너광고 추가하기

👉 배너 광고는 기기 화면의 상단이나 하단에 나타나는 직사각형의 광고를 의미해요. 무료 앱이나 게임에서 많이 보셨을 거예요. 애드몹에서도 앱 광고를 처음 운영할때는 배너 광고를 사용하는 것을 권장합니다.

(1) 애드몹에서 배너 광고 영역을 추가할게요. '앱 > 광고 단위' 메뉴의 '시작하기' 버튼을 눌러 광고형식은 '배너'를 선택해주세요. 광고 단위 이름은 '메인화면 배너'로 설정하고 '광고 단위 만들기'를 눌러주세요.

👉 사용하려는 광고 영역을 추가했어요. 나오는 가이드는 이미 설정을 다 해둔 상태니 '완료' 버튼을 눌러 광고 단위 리스트로 돌아와주세요.

(2) activity_main.xml에 광고를 그려주는 AdView를 추가해주세요. NestedScrollView 끝나는 위치에 추가하여 스크롤 여부와 상관없이 항상 노출될 수 있도록 할게요.

👉 **[코드스니펫] - 레이아웃에 AdView 추가하기**를 복사해서 코드를 넣어주세요. 우선 광고가 잘 나오는지 확인하기 위해서 테스트 광고를 이용해볼게요. 테스트 광고는 애드몹 '테스트 광고 > 샘플 광고 단위' 페이지([링크](#))에서 제공하는 배너 광고 테스트용 AdUnitId 값입니다.

▼ [코드스니펫] - 레이아웃에 AdView 추가하기

```
<com.google.android.gms.ads.AdView xmlns:ads="http://schemas.android.com/apk/res-auto"
    android:id="@+id/adView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    ads:adSize="SMART_BANNER"
    ads:adUnitId="ca-app-pub-3940256099942544/6300978111"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

```
<!-- activity_main.xml -->
...
</LinearLayout>
</androidx.core.widget.NestedScrollView>

<com.google.android.gms.ads.AdView xmlns:ads="http://schemas.android.com/apk/res-auto"
    android:id="@+id/adView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    ads:adSize="SMART_BANNER"
    ads:adUnitId="ca-app-pub-3940256099942544/6300978111"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

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



배너광고의 위치는 일반 레이아웃 뷰와 동일하게 다룰 수 있어요. 앱 하단이 아닌 곳으로 위치를 옮기고 싶으시다면 뷰의 위치 변경하듯이 이동할 수 있습니다.

(3) NestedScrollView의 위치를 조절해볼게요. NestedScrollView의 높이 값이ダイナミック하게 설정되도록 0dp로 변경하고 범위를 adView위로 설정할게요.

```
<!-- activity_main.xml -->
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@id/adView"
        app:layout_constraintTop_toTopOf="parent">
```

(4) 광고 영역을 위한 레이아웃 준비는 완료되었어요! 이제 광고를 불러와볼까요? 메인화면 배너 로드하기 코드를 추가하고 프로젝트를 실행시켜주세요.

▼ [코드스니펫] - 메인화면 배너 로드하기

```
var mainAdView: AdView = findViewById(R.id.adView)
var adRequest = AdRequest.Builder().build()
mainAdView.loadAd(adRequest)
```

```
//MainActivity.kt
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        MobileAds.initialize(this)
        var mainAdView: AdView = findViewById(R.id.adView)
        var adRequest = AdRequest.Builder().build()
        mainAdView.loadAd(adRequest)
    }
    ...
}
```



혹시 광고영역이 표시되지 않나요? 애드레이터의 네트워크 상태를 확인해주세요.

(5) 테스트 광고가 정상적으로 나오는 것을 확인했으니 테스트 광고 id가 아니라 실제 사용할 adUnitId로 변경할게요.



adUnitId는 애드몹 관리자 페이지의 '앱 > 등록된 앱 > 광고 단위' 메뉴에서 추가한 '메인화면 배너'의 id 값으로 'ca-app-pub-xxx/xxx' 형태로 되어있어요.

```
<!-- activity_main.xml -->
...
</LinearLayout>
</androidx.core.widget.NestedScrollView>

<com.google.android.gms.ads.AdView xmlns:ads="http://schemas.android.com/apk/res-auto"
    android:id="@+id/adView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    ads:adSize="SMART_BANNER"
    ads:adUnitId="ca-app-pub-~~~~~"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

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



adUnitId를 변경하면 로그메세지에 'Ad failed to load : 3'가 기록되며 광고가 노출되지 않을거예요. 정상적이니 너무 걱정하지마세요.

이전에는 앱 배포전에 Debug 상태에서도 광고가 잘 출력되었지만, 최근 애드몹이 업데이트되어 앱이 구글플레이에 배포가 된 이후에 광고가 채워진답니다.


(6) 부정 광고로 인해 애드몹 정책 위반하는 것을 방지하기 위해 애드몹 사이트에서 테스트 기기를 추가해둘게요. 애드몹 관리 페이지에서 메뉴의 '설정 > 기기 테스트' 탭으로 이동하여 '테스트 기기 추가' 버튼을 눌러주세요.


!! 주의!! 테스트 광고가 아닌 상태에서 짧은 순간 배너를 열심히 클릭하게 되면 무효 활동으로 계정이 신고될 수 있으니 조심하세요! 광고를 테스트하실 때는 adUnitId를 테스트 ID를 사용하시거나 디바이스를 테스트 기기에 반드시 등록해주세요.


또한, 광고 요청과 광고 클릭이 비정상적으로 판단되는 앱은 애드몹에서 퀄리티가 낮은 광고 지면으로 판단해서 광고가 안채워질 수 있어요.

(7) 구분을 위한 기기 이름을 등록해주시고 플랫폼은 'Android'를 클릭한 뒤 광고 ID/IDFA를 추가해주세요. 광고 ID/IDFA는 안드로이드 디바이스의 '설정 > Google > 광고(Ads)'로 이동하면 나오는 'Your advertising ID' 값이에요.

(8) 제일 중요한 마지막 설정! 광고로 생긴 수익을 지급받으려면 지급 설정을 해야겠죠? 애드몹 관리 페이지에서 메뉴의 '지급' 으로 이동해주세요. '지급 설정하기' 버튼을 눌러 지급을 설정해주세요.

 지급 정보가 등록되어 있지 않으면 광고가 게시되지 않아요. 지급 정보를 꼭 추가해주세요. 지급 정보등록은 바로되지 않고 시간이 걸리는 작업이에요. 애드몹에서는 '이 과정은 보통 최대 소요 시간 이 24시간이지만 드물게는 최대 2주까지 걸릴 수 있습니다. 인증이 완료되면 계정을 통해 알림이 전송됩니다.'라고 가이드하고 있습니다.


 지급 설정 진행중 문제가 발생하시는 경우 애드몹 관리 페이지의 메뉴 '의견'을 활용하시는 것이 더 빠른 도움을 받아 해결할 수 있어요.

 더 다양한 광고를 다루고 싶으시다면 모바일 광고 SDK(Android) 공식 문서의 '광고 형식 선택' 항목([링크](#))를 참고해주세요. 배너 광고 외에도 화면을 전부 가리는 전면 광고, 동영상상을 보여주고 다음 동작을 진행하는 보상형 광고도 있어요. 문서에서 가이드하는대로 따라서 진행하시면 쉽게 구현하실 수 있을 거예요. 진행하실 때 언어는 꼭 Kotlin을 선택해주세요!

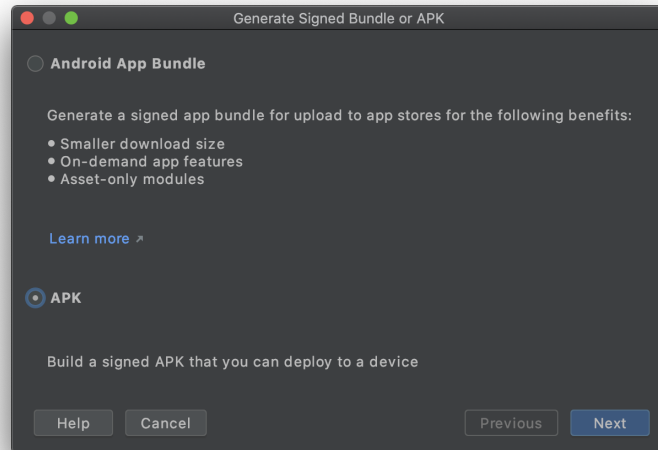
다른 유형의 광고 동작에 대한 테스트를 진행하고 싶으시다면 '애드몹 > 테스트 광고 사용 설정' 페이지([링크](#))를 참고해주세요.

09. 유형 검사 앱 배포하기 (1)

▼ 11) APK 파일 생성하기

 apk는 exe, dmg 같은 설치파일이에요. 구글 플레이에 등록하지 않은 상태로도 apk파일을 이용하면 앱을 설치할 수 있어요. 다만 모르는 apk 파일 설치의 보안상 매우 위험하니 가급적 앱을 설치할 때는 구글 플레이를 사용해주세요.

(1) 안드로이드 스튜디오의 메뉴바에서 'Build > Generate Signed Bundle/APK'를 클릭하면 나오는 창에서 'apk' 타입을 선택해 Next를 눌러주세요.



(2) Key를 선택해 등록해야하는데 생성한 Key가 없네요. 'Create new...'를 눌러 앱 서명 키 스토어(Key Store)를 생성해주세요.

☞ Key store path: 키 스토어를 생성할 위치를 선택해주시고 파일명과 함께 확장자를 꼭 `.jks` 로 설정해주세요.
Password: 키 스토어의 비밀번호를 설정해주세요.

☞ Key
Alias: 키의 이름을 입력해주세요.
Password: 키의 비밀번호를 설정해주세요. 키 스토어의 비밀번호와 달라야해요.
Certificate: 인증서에 개인 정보를 추가해주세요. 앱에는 표시되지 않지만 인증서에 포함돼요.

☞ 키 스토어를 생성할 때 입력한 alias와 비밀번호는 다음 배포를 할 때 꼭 필요한 정보니 잊지말고 기억해주세요.

☞ 앱 서명 키는 사용자의 기기에 설치된 apk에 서명하는 데 사용하는 키로 추후 앱의 업데이트가 진행되는 경우 원래 작성자가 제공한 신뢰할 수 있는 앱인지 확인하는 용도로 쓰입니다.
만든 키 스토어 파일은 공인 인증서와 유사한 역할이니 잘 보관해주세요.

(3) 생성된 키 정보를 입력하고 'Next'를 눌러 주세요.

☞ 'Remember passwords'를 눌러 비밀번호를 기록해두면 나중에 apk 생성시 편리할 수 있지만 보안의 문제가 있을 수도 있으니 각자의 선택에 맡길게요.

(4) Build Variants 를 'Release'를 선택해주시고 Signature Version은 v1과 v2 모두 체크한 뒤 'Finish'를 눌러 apk 생성을 진행해주세요.

(5) 생성이 완료되면 '프로젝트 폴더 > app > release' 폴더로 이동하면 app-release.apk 파일이 생성되어 있는 것을 확인할 수 있어요.

👉 생성이 완료되면 오른쪽 아래에 나타난 메시지 창의 'locate' 버튼을 누르면 편하게 이동가능하답니다.

👉 apk 파일이 완성되었어요! 주변 안드로이드 사용자들이 apk 파일을 이용해 앱을 설치할 수 있어요!

▼ 12) 앱 버전 올리고 AAB(Android App Bundle) 파일 생성하기

👉 AAB는 앱의 모든 컴파일된 코드와 리소스를 포함하고 APK생성 및 서명을 구글 플레이에 맡기는 방식이에요. 구글 플레이는 앱 번들을 사용해 기기별로 최적화된 apk를 생성해서 설치를 지원해준답니다. 장점은 구글플레이에서 자동으로 기기별로 최적화를 해주고 보안 수준이 좋다는 점이에요. 기존 APK와 다른 점은 구글 플레이를 통해서만 설치할 수 있는 방식이라 AAB 파일을 주변에 전달해도 설치할 수 없어요.

(1) 코드의 수정이 생겨서 업데이트를 해야한다고 가정하고 앱의 버전을 올려볼까요? module 레벨의 build.gradle 파일의 defaultConfig에 있는 versionCode와 versionName을 변경하고 'Sync Now'를 눌러주세요.

```
...
android {
    ...
    defaultConfig {
        ...
        versionCode 1
        versionName "1.0"
        versionCode 2
        versionName "1.1"
        ...
    }
    ...
}
```

👉 Sync가 끝나면 앱의 버전 변경 완료!
간단하죠? versionCode와 versionName 을 수정하는 방식은 간단하답니다. 다만, versionCode와 versionName을 어떻게 관리할지는 안드로이드 공식 '앱 게시 > 앱 버전 지정' 문서([링크](#))를 참고해주시면 좋아요.

(2) AAB를 생성하는 방법은 APK 생성방식과 유사해요. 안드로이드 스튜디오의 메뉴바에서 'Build > Generate Signed Bundle/APK'를 클릭해주세요.

(3) 이번에는 apk가 아닌 'Android App Bundle' 타입을 선택해 Next를 눌러주세요. APK 파일 생성할 때 사용했던 키스토어 정보 그대로 기입해서 next를 눌러주세요.

(4) 'Release'를 선택해 'Finish'를 눌러 AAB 생성을 기다려주세요.

(5) 생성이 완료되면 '프로젝트 폴더 > app > release' 폴더로 이동하면 app-release.aab 파일이 생성되어 있는 것을 확인할 수 있어요.

👉 앱 버전을 올리는 방법도 연습했고 AAB 파일도 생성했겠다 이젠 배포만 하면 되겠네요!

10. 유형 검사 앱 배포하기 (2)

▼ 13) 구글 플레이에 앱 기본 정보 등록하기

👉 구글 플레이에 배포를 하려면 플레이 콘솔 개발자 계정으로 등록해야해요. 등록하는데 등록 수수료 25달러를 1회 결제로 평생 개발자 계정이 유지돼요.

(1) 구글플레이에 배포를 하기 위한 플레이 콘솔([링크](#))로 이동해주세요.

👉 계정이 없으면 '새 개발자 계정 만들기' 화면이 나올거예요. '공개 개발자 이름', '보조 연락처 이메일 주소', '연락처 전화번호', '개발자 계약 및 서비스 약관'의 내용을 확인 후 '개정 생성 및 결제' 버튼을 눌러 25달러 결제를 진행해주세요!
결제가 완료되면 플레이 콘솔 페이지가 나올거예요.

(2) 플레이 콘솔에서 '앱 만들기'를 이용해 앱을 등록할게요. 앱 세부정보를 내용을 채우고 '앱 만들기' 버튼을 누르면 앱 대시보드 화면과 함께 '앱 설정 시작'이라고 표시가 될 거예요.

👉 '앱 이름'은 프로젝트 앱 이름과 동일한 필요는 없으며 나중에 변경할 수 있어요.
'기본 언어'는 한국어로 작성되었으니 '한국어'로, 게임은 아니니까 '앱'을 선택해주시고 광고를 붙였으니 '무료'앱으로 설정할게요.

(3) 메뉴 바의 '프로덕션'으로 이동해 '새 버전 만들기'를 눌러 앱을 업로드를 시작해볼게요. 'Play 앱 서명' 항목의 '계속'을 눌러 Play 앱 서명 서비스 약관에 동의해주세요.

프로덕션 버전 만들기

프로덕션 버전은 선택한 국가의 모든 사용자에게 제공됩니다.

1 준비 — 2 검토 및 출시

[버전 삭제](#)

Play 앱 서명

✔ Google에서 앱 서명 키 보호 중


Google에서 앱 서명 키를 만들고 보호하며 각 버전에 서명하는 데 사용합니다. 이를 통해 개발자 본인이 제공한 업데이트가 맞는지 확인할 수 있습니다. Android App Bundle을 이용해 앱을 게시하려면 앱 서명이 필요합니다. [자세히 알아보기](#)

[계속](#)

[앱 서명 관리](#)


(4) 서비스 약관에 동의하면 활성화되는 App Bundle 및 Apk 항목의 '업로드' 버튼을 눌러 'app-release.aab' 파일을 선택해 업로드해주세요. 업로드가 완료되면 버전을 확인해보면 2 (1.1) 로 설정된 것을 확인할 수 있어요.


App Bundle 및 APK



Android App Bundles(.aab) 또는 APK를 여기에 놓아 업로드하세요.

[업로드](#) [라이브러리에서 추가](#)

app-release.aab 

파일 형식	버전	API 수준	타겟 SDK	화면 레이아웃	ABI	필수 기능
App bundle	2 (1.1)	26 이상	29	4	전체	1  →

👉 2021년 8월부터 구글 플레이에 앱을 업로드할 때는 APK 파일 형식은 사용할 수 없고 AAB(Android App Bundle)형식을 사용해야해요. 검색을 통해서 APK 관련 정보를 얻게되면 오래된 정보라는 것을 기억하시고 보시면 좋아요!

(5) 버전 세부정보를 기입해주세요. 출시 노트의 'ko-KR 출시 노트를 여기에 입력하거나 붙여넣으세요.' 내용을 '첫 출시'라고만 변경해 '버전 검토' 옆에 있는 '저장' 버튼을 눌러주세요.

```
<ko-KR>
첫 출시
</ko-KR>
```



출시명은 플레이 콘솔에서만 확인할 수 있는 정보로 일반적으로 기본생성된 버전 정보를 이용해 관리를 한답니다.

출시 노트는 기존 사용자들에게 업데이트가 나타날 때 변경사항으로 보여지는 정보예요. 출시 노트의 포맷은 xml 형태로 언어별로 다르게 작성 가능해요.

(6) '저장' 버튼을 누르면 활성화된 '버전 검토'를 눌러 출시를 시작하려하니 '검토 및 출시' 화면에서 오류가 나타나네요. 오류를 수정해볼게요. 오류 밑의 '더보기'를 눌러 오류를 해결해주세요.



'오류'는 해결되기 전까지 앱을 출시할 수 없어요. '경고'는 출시가 가능하지만 플레이 콘솔에서 권장사항에 대해서 제안하는 항목이에요. 해결하고 싶다면 '더보기'를 눌러 가이드에 맞춰서 검색하고 진행해주시면 된답니다.

(7) '대시보드'로 이동해 '앱 설정'의 '앱에 관한 정보 제공 및 스토어 등록정보 설정'에서 '할 일 보기'를 누르면 앱 설정 필수항목이 나와요. 앱의 성격에 맞게 설정 저장 후 대시보드로 이동하며 앱 설정을 완료해주세요.

앱 설정



앱에 관한 정보 제공 및 스토어 등록정보 설정

Google Play에 앱의 콘텐츠를 알리고 Google Play에서 앱이 조직 및 제시되는 방식을 관리하세요.

할 일 숨기기 ^

앱 콘텐츠에 관한 정보 입력

- 앱 액세스 권한 >
- 광고 >
- 콘텐츠 등급 >
- 타겟층 >
- 뉴스 앱 >

앱이 분류 및 표시되는 방식 관리

- 앱 카테고리 선택 및 연락처 세부정보 제공 >
- 스토어 등록정보 설정 >



[유형 검사 앱 기준 설정값]

앱 액세스 권한: '특수한 액세스 권한 없이 모든 기능 이용 가능'

광고: '예, 앱에 광고가 있습니다.'

콘텐츠 등급 - 카테고리: 유틸리티, 생산성, 통신 및 기타

콘텐츠 등급 - 설문지: 모두 아니요

타겟층 - 13~15세, 16~17세, 만 18세 이상

타겟층 - 어린이의 관심을 유도함 - 아니요

뉴스 앱 - 아니요

👉 앱이 분류 및 표시되는 방식 관리는 구글 플레이에 노출되는 정보로 신경써서 작성해주시면 좋아요!

👉 앱 아이콘은 512×512, 그래픽 이미지는 1024×500 로 사이즈가 정확히 맞아야해요. 이미지 편집기(그림판, 포토샵)을 이용해 만들어주세요. 이미지 편집기가 없는 경우 [pixlr\(링크\)](#)를 사용해보세요.

- 고생하셨어요! 이제 출시를 위한 사전준비가 마무리되었어요. 이제 출시만 남았답니다!

▼ 14) 구글 플레이에 앱 출시하기

(1) '대시보드' 항목에 'Google Play에 앱 게시'의 '국가 및 지역 선택'으로 들어가 앱 제공 국가와 지역을 설정해주세요. '국가/지역' 옆에 있는 체크박스를 이용해 전체선택할 수 있어요.

👉 만약 대시보드에서 항목을 못찾으시겠다면 '출시 > 프로덕션 > 국가/지역'에서도 '국가/지역'을 관리할 수 있습니다.

(2) '프로덕션' 메뉴로 이동하면 오른쪽 위에 '버전 수정' 버튼을 눌러 전에 진행하던 앱을 배포해볼게요. '버전 검토'를 누르면 이전과 달리 오류가 나타나지 않네요! 이제 '프로덕션 트랙으로 출시 시작' 버튼을 눌러주세요. 안내 문구를 확인하고 '출시'를 누르면 완료!

!! 앗, 검색했는데 앱이 조회되지 않나요?
플레이 콘솔에서 출시를 누르고 24시간정도 시간이 지나야 실제로 앱이 구글플레이에 노출이 돼요.

👉 구글플레이에 등록을 완료했으니 애드몹에 스토어를 추가해볼까요? 애드몹 페이지([링크](#))로 이동해주세요. 메뉴의 '앱 > 등록된 앱 > 앱 설정'으로 이동해 '앱 스토어' 항목에 있는 '추가' 버튼을 눌러 안내하는 내용에 맞춰서 진행해주시고 '저장'을 누르면 완료!
아직 구글 플레이에서 검색되지 않는다면 구글 플레이에 출시된 이후에 잊지말고 애드몹에서 '앱 스토어' 추가 진행해주세요!

👉 이후 앱에 기능을 추가하거나 버그를 해결해서 스토어에 출시를 하실 때 주의하실 점은 새로 업로드하는 앱의 버전코드는 항상 가장 높아야해요. 일반적으로 업데이트할때마다 버전코드를 1씩 증가시키면 된답니다.

11. 5주차 끝 & 숙제 설명

💬 나만의 유형 검사 앱을 완성시켜 앱의 apk파일을 업로드해주세요.

나만의 유형 검사 앱을 배포한 뒤 구글플레이 링크를 공유해주시면 더 좋아요!

▼ 15) 수업 마무리



5주 동안 고생하셨습니다!!!

처음 마우스로만 프로젝트를 생성해 레이아웃을 겨우겨우 다루다 이제 앱으로 광고로 수익도 기대해볼 정도로 여러분은 5주동안 정말 많이 많이 성장하셨습니다!

다른 안드로이드 앱들과 비교하자니 5주동안 만든 앱이 너무 하찮게 느껴지시나요? 구글플레이에 등록된 유명한 앱들은 여러명의 경력자들이 팀을 이뤄 만든 앱이에요. 여러분은 이제 막 튜토리얼을 끝내고 초보자를 벗어난 상태로 기획, 디자인, 개발을 혼자 다하며 앱을 만들었습니다. 전혀 부끄러운게 아니에요. 앞으로 꾸준히 레이아웃 연습하며 디자인을 추가하고 코틀린 연습하면서 기능을 추가해나가다보면 어느새 부끄럽지 않을 인기 앱이 될 수 있어요!

여전히 영어도 코드 갖고 안그래도 있던 영어 울렁증에 코드 울렁증까지 생긴 것 같다고요? 이제 5주되었으니 어렵고 실수하는 것은 당연한 것입니다. 그래도 현재 여러분의 개발 실력은 언어와 비교하면 아무리못해도 여행지에서 생존하며 여행할 정도의 실력입니다. 여행용 언어도 갑자기 복잡한 문장을 만나거나 고급어휘를 쓰면 당황스럽지만 연습하고 부딪히다보면 어느새 간단한 회화가 가능해지고 현지에서 지내는데 어려움이 없 어지잖아요! 마찬가지로 안드로이드 앱 개발도 하나씩 계속 검색하고 만들다보면 프로 개발자로 성장해 있을 거예요!

여러분의 앞날에 에러와 버그가 많지 않기를 기원하며 수업 마치겠습니다.

Copyright © TeamSparta All rights reserved.