● JetPack: 서포트 라이브로리와 아키텍처 컴포넌트를 모아둔 패키지 묶음. JetPack에서 제공하는 아키텍처 구성 요소(Android Architecture Component, AAC)는 안드로이드 내에 코드를 구조화하는 데 도움을 주는 라이브러리다. Androidx로 시작하는 패키지명을 갖는 다양한 라이브러리를 묶어 둔 것이다.

androidx.appcompat	API 레벨 호환성 해결
androidx.recyclerview	목록 화면 구성
androidx.viewpager2	스와이프 화면 구성
androidx.fragment	액티비티처럼 동작하는 뷰 제공
androidx.drawerlayout	서랍처럼 열리는 화면 구성

● 안드로이드 API Level과 하위 호환성: minSdk는 개발하는 앱이 몇 버전까지 지원하는가를 설정하는 것인데, minSdk 하위 버전의 스마트폰에서는 앱 자체가 설치되지 않는다. 그렇게 때문에 설정한 버전까지는 에러가 발생하지 않게 신경을 써야 하는데, 이 부분이 하위 호환성이다. 프로그램을 작성하면서 minSdk보다 상위 버전에서 제공하는 API를 사용한다면, 그보다 하위 버전에서 발생하는 에러를 신경 쓰면서 개발해야 한다. 이를 위해선 크 게 세 가지 방법이 있는데, 구글 라이브러리의 도움을 받거나, 오픈 소스 라이브러리(인터넷 상에서 무료로 구할 수 있는 다양한 라이브러리)의 도움을 받거나, 아니면 개발자 코드에서 버전을 직접 식별해서 처리하면 된다.

▶ JetPack (구글 라이브러리): 예를 들면 ActionBar과 Fragment 등은 플랫폼 API로, android.app.ActionBar을 사용하면 하위 호환성 문제가 발생할 수 있다. 이런 하위 호환성을 해결하기 위해 androidx의 appcompat 라이브러리를 사용한다.
안드로이드 스튜디오에서 플랫폼 API 이외의 라이브러리를 이용하려면, 그레이들 파일에서 dependencies로 연결해야 한다. 앱의 그레이들 파일엔 자동적으로 appcompat 라이브러리가 추가돼 있고, 이 라이브러리에서 앱의 기초적인 API 하위 호환성을 제공한다.

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.4.1'
    // ...
}
```

```
public class MainActivity extends AppCompatActivity {
    // 생략
}
```

▶ 개발자 코드로 버전 확인: androidx 라이브러리가 모든 하위 호환성을 제공할 수는 없어서, 때로는 개발자 코드에서 직접 API Level을 체크해서 분기문으로 작성해야 한다. 다음은 하위 호환성에 걸리는 API를 하위 버전에서 실행되지 않게 개발자가 직접 제어한 예이다.

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
    // ...
} else {
    // ...
}
```

Resources: 깡쌤의 안드로이드 프로그래밍