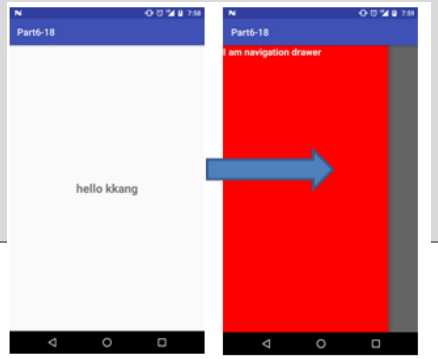
18장. 머티리얼 디자인

18.1 NavigationDrawer, NavigationView

18.1.1. NavigationDrawer

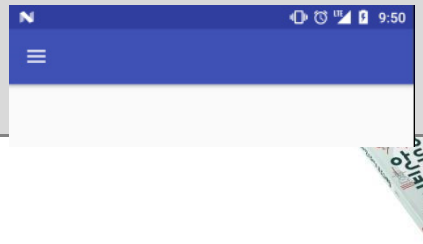
- NavigationDrawer는 support-v4에서 제공하는 클래스
- 레이아웃 XML 파일의 구성

```
<android.support.v4.widget.DrawerLayout>
   <LinearLayout>
      <!-- 중략 -->    </LinearLayout>

   <TextView
      android:layout_gravity="start"
      />

</android.support.v4.widget.DrawerLayout>
```



- ActionBarDrawerToggle

```
drawer = (DrawerLayout) findViewById(R.id.main_drawer);
toggle = new ActionBarDrawerToggle(this, drawer, R.string.drawer_open,
R.string.drawer_close);getSupportActionBar().setDisplayHomeAsUpEnabled(true);
toggle.syncState();
```



18.1 NavigationDrawer, NavigationView

- ActionBarDrawerToggle을 정의할 때 이벤트

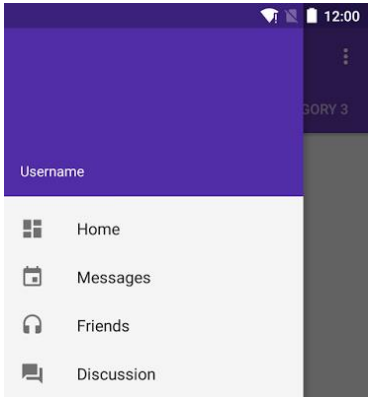
```
toggle = new ActionBarDrawerToggle(this, drawer, R.string.drawer_open, R.string.drawer_close){
    @Override
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);
        //...
    }
    @Override
    public void onDrawerClosed(View drawerView) {
        super.onDrawerClosed(drawerView);
        //...
    }
};
drawer.addDrawerListener(toggle);
```

- 메뉴 이벤트 함수

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (toggle.onOptionsItemSelected(item)) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

18.1 NavigationDrawer, NavigationView

18.1.2. NavigationView

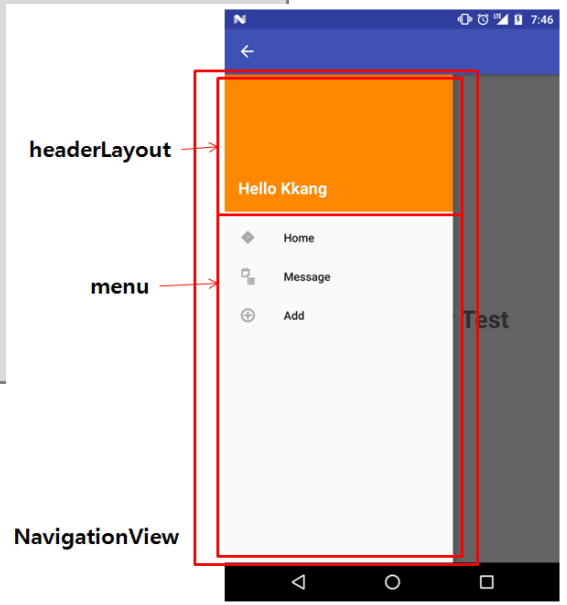


- NavigationView는 design support 라이브러리에 추가된 뷰

```
implementation 'com.android.support:design:26.0.1'
```

- Layout xml

```
<android.support.v4.widget.DrawerLayout>

    <RelativeLayout ... >
                <!--중략-->
    </RelativeLayout>

    <android.support.design.widget.NavigationView
        ...
        android:layout_gravity="start"
        app:headerLayout="@layout/navigation_header"
        app:menu="@menu/menu_drawer" />
</android.support.v4.widget.DrawerLayout>
```



18.1 NavigationDrawer, NavigationView

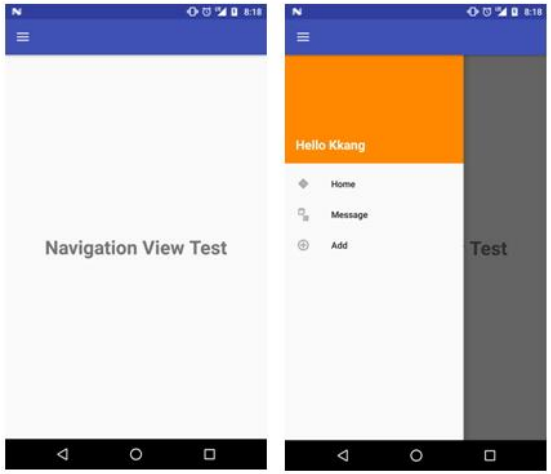
- NavigationView에서의 사용자 항목 선택 이벤트

```
navigationView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(MenuItem menuItem) {
        int id=menuItem.getItemId();
        //...        return false;
    }
});
```

Step by Step 18-1 - NavigationView

NavigationDrawer와 NavigationView 를 테스트

1. Module 생성
2. 파일 복사
3. gradle 설정
4. activity_main.xml 작성
5. strings.xml 작성
6. MainActivity 작성
7. 실행

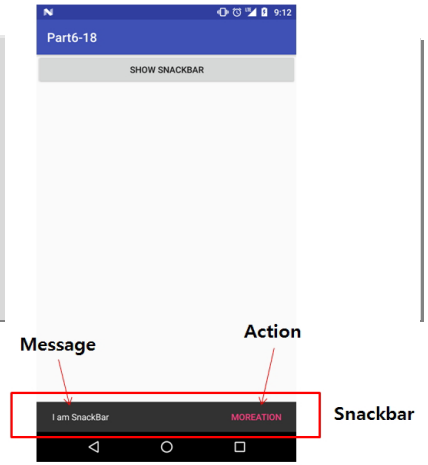


18.2 Snackbar, FloatingActionButton, TabLayout

18.2.1. Snackbar

- Snackbar는 design support 라이브러리에서 제공하는 클래스로 사용자에게 간단한 문자열 메시지를 보여줄 목적으로 사용
- 토스트(Toast)와 비슷하지만, Snackbar는 문자열 메시지가 잠깐 보이는 동안 사용자의 추가 이벤트를 받을 수 있다.

```
Snackbar.make(relativeLayout, "I am SnackBar", Snackbar.LENGTH_LONG)
        .setAction("MoreAtion", new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //...
    }
}).show();
```

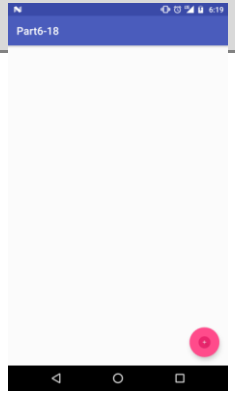


18.2 Snackbar, FloatingActionButton, TabLayout

18.2.2. FloatingActionButton

- FloatingActionButton은 클래스명처럼 액티비티 화면에 떠 있는 것처럼 보이는 둥근 테두리의 이미지 버튼

```
<android.support.design.widget.FloatingActionButton
    ...
    android:src="@drawable/ic_floating"
    app:fabSize="normal"
    app:rippleColor="#FFFFFF" />
```



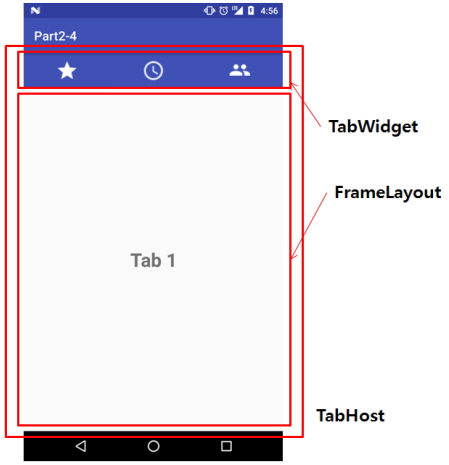
- fabSize: 버튼의 크기 설정(normal 혹은 mini)
- rippleColor: 버튼을 눌렀을 때 나타나는 효과의 색상
- src: 버튼 안에 사용할 아이콘

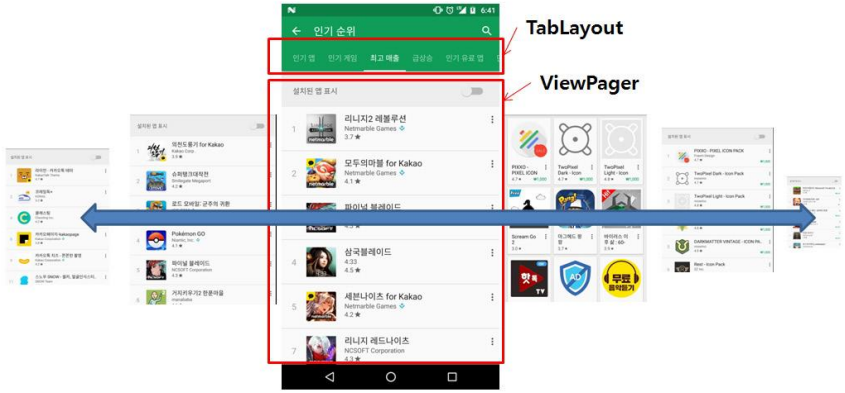
18.2 Snackbar, FloatingActionButton, TabLayout

18.2.3. TabLayout

- TabHost를 이용한 탭 화면
- TabHost는 탭 버튼을 다양하게 제공하기 힘들다.
- ViewPager 연동이 자동화되지 않다.

- design support 라이브러리에서 TabLayout 제공
- TabLayout은 탭 화면의 버튼 부분을 다양하게 제공할 목적
- 탭 본문을 ViewPager로 구현한다면 ViewPager와 연동





18.2 Snackbar, FloatingActionButton, TabLayout

```
<android.support.design.widget.TabLayout
    android:id="@+id/lab2_tabs"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    app:tabMode="scrollable" />
```

- ViewPager와 연동

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.lab2_tabs);
tabLayout.setupWithViewPager(viewPager);
tabLayout.addOnTabSelectedListener(this);
```

- ViewPager의 Adapter

```
class MyPagerAdapter extends FragmentPagerAdapter {
    //...
    private String titles[] = new String[]{"TAB1", "TAB2", "TAB3"};
    //...
    @Override
    public CharSequence getPageTitle(int position) {
        return titles[position];
    }
}
```

18.2 Snackbar, FloatingActionButton, TabLayout

- 탭 버튼을 사용자가 클릭했을 때 이벤트

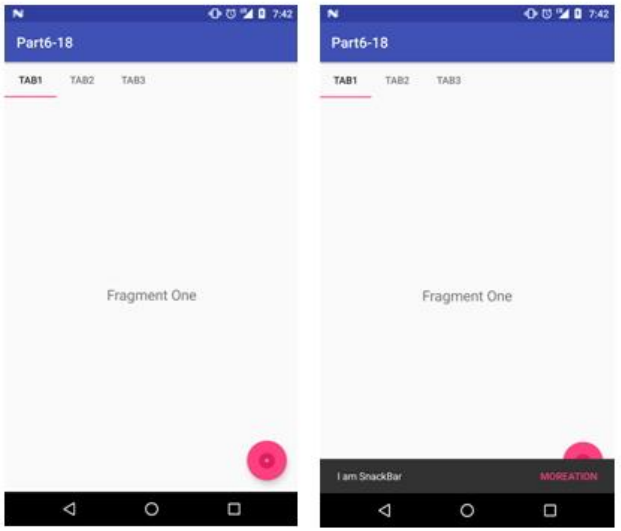
```
public void onTabSelected(TabLayout.Tab tab) {
    viewPager.setCurrentItem(tab.getPosition());
}
```

Step by Step 18-2 – TableLayout, FloatingActionButton, Snackbar

TabLayout과 FloatingActionButton, Snackbar 테스트

•TabLayout에 의한 탭 버튼은 ViewPager와 연동
•ViewPager의 화면은 단순 문자열이 출력되는 Fragment로 작성

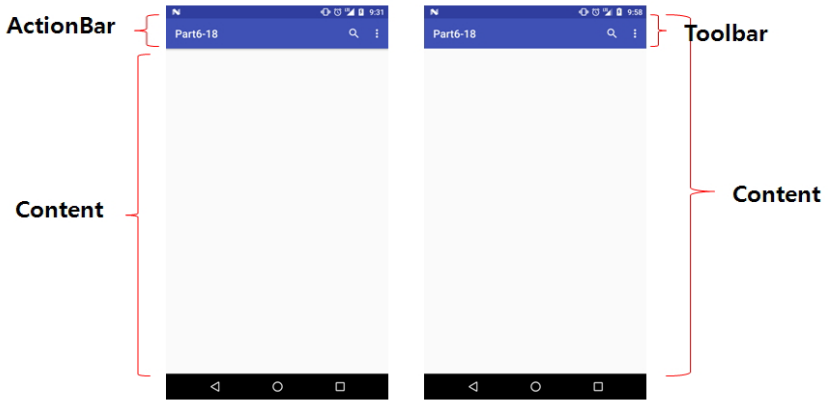
1. Activity 생성
2. 파일 복사
3. activity_lab18_2.xml 작성
4. Lab18_2Activity 작성
5. Lab18_2Activity.java 실행



18.3 Toolbar, AppBarLayout, CoordinationLayout

18.3.1. Toolbar

- Toolbar는 API Level 21(Android 5.0)부터 표준 라이브러리에서 제공하는 뷰
- 하위 호 환성 문제로 대부분 표준 라이브러리의 클래스를 사용하지 않고 appcompat-v7 라이브러리에서 제 공하는 같은 이름의 클래스를 이용



- ActionBar를 출력하지 않게 설정

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

18.3 Toolbar, AppBarLayout, CoordinationLayout

- 레이아웃X ML 파일에 Toolbar를 등록

```
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="@color/colorPrimary"/>
```

- Toolbar에 적용

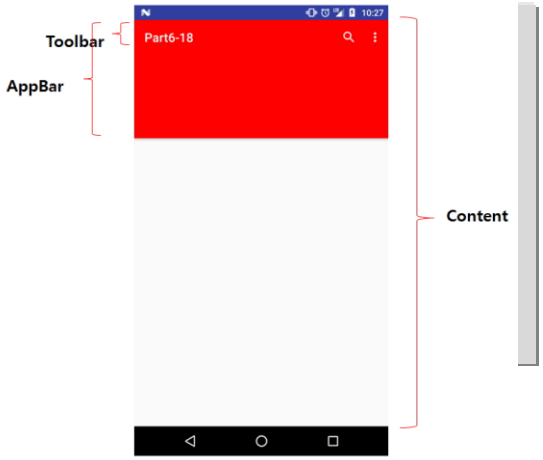
```
toolBar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolBar);
```

18.3 Toolbar, AppBarLayout, CoordinationLayout

18.3.2. AppBarLayout

- design support 라이브러리에서 제공
- AppBarLayout은 Toolbar를 포함하는 개념으로 액티비 티 화면의 상단을 다양하게 꾸미기 위한 레이아웃

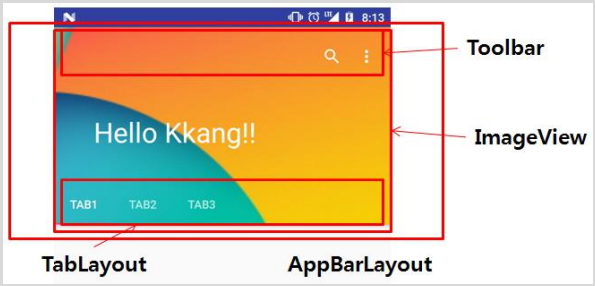
```
<android.support.design.widget.AppBarLayout
    android:id="@+id/appbar"
    android:layout_width="match_parent"
    android:layout_height="192dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    android:background="#FFFF0000">

    <android.support.v7.widget.Toolbar ... />
</android.support.design.widget.AppBarLayout>
```



18.3 Toolbar, AppBarLayout, CoordinationLayout

- ImageView나 TabLayout, CollapsingToolbarLayout을 포함하여 작성 가능

```
<LinearLayout ... >

   <android.support.design.widget.AppBarLayout
      android:id="@+id/lab3_appbar"
      android:layout_width="match_parent"
      android:layout_height="242dp"
      android:fitsSystemWindows="true"
      android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

      <android.support.design.widget.CollapsingToolbarLayout
         android:id="@+id/lab3_collapsing"
         android:layout_width="match_parent"
         android:layout_height="242dp"
         app:expandedTitleMarginBottom="100dp"
         app:expandedTitleMarginStart="48dp"          app:title="Hello Kkang!!">

         <ImageView  .../>

         <android.support.v7.widget.Toolbar  ... />
         <android.support.design.widget.TabLayout  ... />
      </android.support.design.widget.CollapsingToolbarLayout>
   </android.support.design.widget.AppBarLayout>

   <android.support.v4.view.ViewPager ... />

</LinearLayout>
```



18.3 Toolbar, AppBarLayout, CoordinationLayout

18.3.3. CoordinatorLayout

- CoordinatorLayout은 뷰 간의 상호 작용을 목적
- FloatingActionButton과 Snackbar의 상호 연동, AppBar와 RecyclerView의 상호 연동

- Snackbar의 스크롤에 의한 FloatingActionButton이 함께 스크롤

```
<android.support.design.widget.CoordinatorLayout ... >
    <android.support.design.widget.FloatingActionButton
        ...
        />
</android.support.design.widget.CoordinatorLayout>
```

- CoordinatorLayout을 지정

```
Snackbar.make(coordinatorLayout, "I am SnackBar", Snackbar.LENGTH_LONG).show();
```

18.3 Toolbar, AppBarLayout, CoordinationLayout

- CoordinatorLayout을 이용하여 AppBar와 RecyclerView 연동

```
<android.support.design.widget.CoordinatorLayout ... >

   <android.support.design.widget.AppBarLayout ... >

      <android.support.design.widget.CollapsingToolbarLayout
         ...          app:contentScrim="?attr/colorPrimary"
         app:layout_scrollFlags="scroll|exitUntilCollapsed">

         <ImageView
            ...
            app:layout_collapseMode="parallax" />

         <android.support.v7.widget.Toolbar
            ...              app:layout_collapseMode="pin" />
      </android.support.design.widget.CollapsingToolbarLayout>
   </android.support.design.widget.AppBarLayout>

   <android.support.v7.widget.RecyclerView
      ...
      app:layout_behavior="@string/appbar_scrolling_view_behavior" />

</android.support.design.widget.CoordinatorLayout>
```

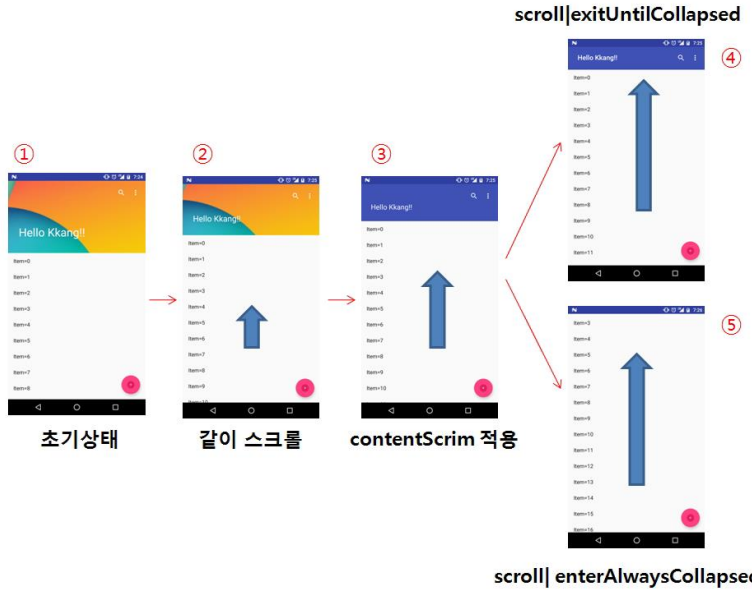
18.3 Toolbar, AppBarLayout, CoordinationLayout

CollapsingToolbarLayout의 layout_scrollFlags 속성

- scroll: 스크롤 되게 지정
- exitUntilCollapsed: minHeight(Toolbar의 세로 크기)까지만 스크롤 되게 지정
- enterAlwaysCollapsed: 스크롤 되어 전체가 사라지게 지정

layout_collapseMode

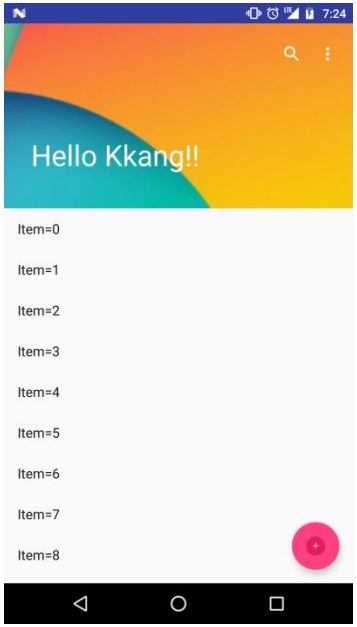
- parallax: 초기 스크롤부터 함께 스크롤 되게 지정
- pin: 초기 고정 상태로 스크롤 되지 않게 지정



Step by Step 18-3 – CoordinatorLayout, AppBarLayout

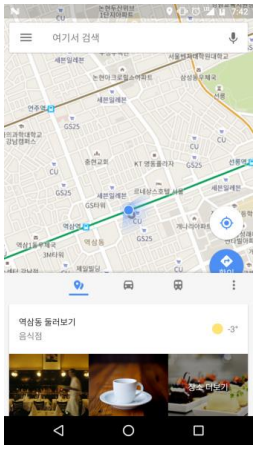
CoordinatorLayout – AppBarLayout의 상호 연동 부분을 테스트

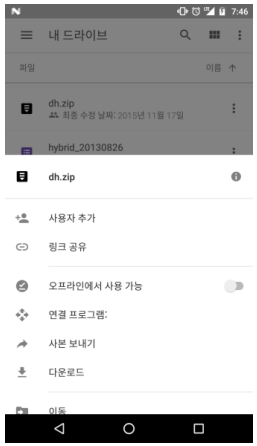
1. Activity 생성
2. 파일 복사
3. style 정의
4. activity_lab18_3.xml 작성
5. Lab18_3Activity.java 실행



18.4 Bottom Sheet

- Bottom Sheet는 머티리얼 디자인과 함께 소개
- design support 라이브러리에서 제공
- Bottom Sheet는 액티비티 창의 Content 영역 구성과 별도로 하단에 부가 내용을 보여주기 위해 사용
- persistent bottom sheet와 modal bottom sheet 두 가지 형태





18.4 Bottom Sheet

18.4.1. persistent bottom sheet

- persistent bottom sheet는 화면을 인앱(In-App) 개념으로 구성하기 위한 뷰
- persistent bottom sheet는 액티비티 출력 시 초 기부터 화면에 출력할 수 있으며, 사용자가 스크롤 하여 확대, 축소 가능
- persistent bottom sheet가 화면에 보이더라도 액티비티의 뷰 개념이므로 메인 콘텐츠 부분을 사용자가 얼마든지 이용 가능

- modal bottom sheet는 다이얼로그 개념
- 자바 코드에서 show( ) 함수로 출력

- persistent bottom sheet 를 위한 뷰

```
<android.support.design.widget.CoordinatorLayout ... >
   <!-- Activity Main Content View-->
   <LinearLayout ... >
      ...
   </LinearLayout>    <!-- Bottom Sheet View-->
   <LinearLayout ...
      android:layout_height="250dp"
      app:behavior_peekHeight="120dp"
      app:layout_behavior="android.support.design.widget.BottomSheetBehavior">
      <!-- 중략-->
   </LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```

18.4 Bottom Sheet

- layout_height: bottom sheet의 최대 스크롤 크기
- behavior_peekHeight: bottom sheet의 초기 크기, 최소 스크롤 크기
- behavior_hideable: 사용자 스크롤에 의해 사라질 것인지 설정

bottom sheet 적용

```
View bottomSheet = coordinatorLayout.findViewById(R.id.lab4_bottom_sheet);
persistentBottomSheet = BottomSheetBehavior.from(bottomSheet);
```

- 이벤트

```
persistentBottomSheet.setBottomSheetCallback(new                    BottomSheetBehavior.BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
    }

    @Override
    public void onSlide(@NonNull View bottomSheet, float slideOffset) {
    }
});
```

18.4 Bottom Sheet

18.4.2. modal bottom sheet

- modal bottom sheet
- modal bottom sheet을 위한 레이아웃 XML 파일

```
<android.support.v7.widget.RecyclerView ...
    app:behavior_peekHeight="120dp" />
```

- bottom sheet로 나타나게

```
View view = getLayoutInflater().inflate(R.layout.lab4_modal_sheet, null);
//...
modalBottomSheet = new BottomSheetDialog(this);
modalBottomSheet.setContentView(view);
modalBottomSheet.show();
```

Step by Step 18-4 – Bottom Sheet

Bottom Sheet 테스트

•화면은 단순 처리 할것이며 persistent bottom sheet와 modal bottom sheet를 모두 테스트

1. Activity 생성
2. 파일 복사
3. activity_lab18_4.xml 작성
4. Lab18_4Activity 작성
5. Lab18_4Activity.java 실행