● ActionBar: 액티비티가 출력되는 전체 창을 Window라 부르며, Window는 크게 Content 영역과 ActionBar 영역으로 나누어 진다. Content 영역은 액티비티 내에서 setContentView() 함수로 작성한 내용이 나오는 곳이고, ActionBar은 Window의 상단에 출력된다. ActionBar 구성을 개발자가 특별하게 하지 않으면 기본 타이틀 문자열이 나온다. 하지만 Navigation Icon, Title 문자열, Menu(Action Button, Overlfow Menu) 등이 출력되도록 구성할 수도 있다.

▶ Overlay 모드로 ActionBar 출력하기: ActionBar을 Content 영역 위에, 즉, 떠 있는 듯하게 출력(Overlay 모드)해야 할 때도 있다. Theme을 정의하는 XML 설정으로 간단하게 구현할 수 있다. Theme을 정의한 XML에 windowActionBarOverlay 속성을 추가해 이 값을 true로 설정하면 된다. ActionBar 자체의 백그라운드 색상 때문에 Content의 일부분이 완전히 가려지는 것을 방지하기 위해 colorPrimary 속성값을 투명하게 지정할 수도 있다.

```
<style name = "Theme.AndroidLab" parent = "Theme.MaterialComponents.DayNight.DarkActionE
    <item name = "colorPrimary">#00000000</item>
    <!--- 생략 ----->
    <item name = "windowActionBarOverlay">true</item>
</style>
```

▶ ActionBar 출력 동적 제어: 앱이 실행되는 도중에 ActionBar가 사라지거나, 다시 나타나게 구현해야 할 때가 있다. 먼저 코드에서 ActionBar 객체를 획득해야 한다. getSupportActionBar() 함수로 획득하며, 레벨 호환성을 고려해 androidx.appcompat.app.ActionBar 사용을 권장한다. 그리고 ActionBar 객체의 show(), hide() 함수를 이용해 원하는 순간에 ActionBar가 사라지거나 다시 나타나도록 구현한다.

```
// androidx.appcompat.app.ActionBar
ActionBar actionBar = getSupportActionBar();

binding.showButton.setOnClickListener(view -> {
    actionBar.show();
});

binding.hideButton.setOnClickListener(view -> {
    actionBar.hide();
});
```

▶ Up 버든 설정: 출력되는 액티비티가 첫 화면이 아닐 경우 사용자가 Back 버든을 눌러 이전 화면으로 되돌아갈 수 있는데, ActionBar에 Up 버든을 제공하여 사용자가 클릭 시 이전 화면으로 되돌아갈 수 있도록 할 수 있다. XML 파일 설정하는 법도 있고, 코드로 작성하는 방법도 있다.

1) XML 파일 설정: AndroidManifest.xml 파일에 아래와 같이 <activity> 태그 내에 parentActivityName 속성을 지정해 주면, 자동으로 Up 버든이 출력되며, 사용자가 클릭할 때 parentActivityName 속성에 지정한 액티비티로 화면이 전환된다.

```
<activity
    android:name = ".DetailActivity"
    android:exported= "false"
    android:parentActivityName = ".MainActivity"/>
```

2) 코드: setDisplayHomeAsUpEnabled() 함수를 사용하는데, 이는 Up 버튼을 출력하는 기능만 한다. 사용자가 Up 버튼을 클릭했을 때 특정 로직이 실행되도록 하려면, onSupportNavigateUp() 함수를 오버라이드 해야 한다.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // 생략...
    ActionBar actionBar = getSupportActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
}

@Override
public boolean onSupportNavigateUp() {
    onBackPressed();
    return super.onSupportNavigateUp();
}
```

● 메뉴: 개발자가 액티비티를 개발할 때 선택적으로 추가할 수 있으며, Action Button과 Overlfow Menu가 있다.

▶ 메뉴 작성 방법: 자바 코드로 작성하는 방법과 리소스 XML 파일을 이용하는 두 가지 방법이 있다.
1) 자바 코드: 메뉴를 구성하는 함수 onCreateOptionsMenu() 함수를 재정의 해야 한다. 매개변수로 Menu 객체가 넘어오고, 이 객체에 메뉴를 넣으면 된다. 이때 다음과 같은 add()함수를 사용한다.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuItem item1 = menu.add(0, 0, 0, "슬라이드쇼");
    MenuItem item2 = menu.add(0, 1, 0, "앨범에추가");
    return true;
}
```

```
add(CharSequence title)
add(int groupId, int itemId, int order, int titleRes)
add(int groupId, int itemId, int order, CharSequence title)
```

add() 함수의 매개변수 중 itemId 값은 메뉴의 식별자로 사용자의 메뉴 선택 이벤트를 처리할 때 어느 메뉴가 눌렸는지 확인할 수 있다. title 매개변수는 메뉴에 표시할 문자열이다. add()함수는 추가된 메뉴 하나를 지칭하는 menuItem 객체를 반환한다. 액티비티에 메뉴를 추가하면 ActionBar의 오른쪽에 메뉴가 있다는 표시로 오버플로우(Overflow) 아이콘이 나타나고, 사용자가 이 아이콘을 클릭하면 메뉴가 확장된다. 이런 것을 오버플로 메뉴(Overflow Menu)라고 부른다. 다음은 MenuItem의 메뉴 구성을 위한 함수들이다.

```
setIcon(int iconRes)                // 아이콘 이미지
setTitle(CharSequence title)        // 문자열
setVisible(boolean visible)         // 화면에 보이는 상태
setEnabled(boolean enabled)         // 활성 상태
```

메뉴는 사용자 이벤트를 목적으로 한다. 사용자가 메뉴를 클릭했을 때 이벤트 처리는 onOptionsItemSelected() 함수에서 처리한다. onOptionsItemSelected() 함수는 화면에서 메뉴가 선택된 순간 자동 호출되며, 매개변수로 선택한 메뉴 객체가 전달된다. MenuItem의 getItemId() 함수로 이벤트가 발생한 메뉴의 id 값을 추출하여 이벤트 로직을 작성하면 된다.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == 0) {
       // ...
    } else if (item.getItemId() == 1) {
       // ...
    }
    return super.onContextItemSelected(item);
}
```

2) 리소스 XML 이용: 액티비티가 실행될 때마다 항상 똑같다면 리소스 XML을 이용하여 메뉴를 구현하는 방법도 있다. XML를 만들고 리소스화해서 메뉴를 구현하는 방법이다. 메뉴 XML파일이 저장될 위치는 res 폴더 하위의 menu 폴더이다.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menu1"
        android:icon="@drawable/ic_menu_1"
        android:title="선택..." />
    <item
        android:id="@+id/menu2"
        android:icon="@drawable/ic_menu_2"
        android:title="레이아웃" />
</menu>
```

이렇게 XML 파일에 구성한 메뉴를 액티비티에 적용하기 위해서, 자바 코드에서 MenuInflator을 이용한다. 각 <item> 태그에 메뉴를 식별하기 위한 id 값을 R 파일에 등록하게 되며, 자바 코드에서 메뉴를 식별할 때 R 파일의 int 변수로 이벤트를 처리하면 된다.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflator inflator = getMenuInflator();
    inflator.inflate(R.menu.menu_main, menu);
    return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.menu1) {
        // ...
    } else if (item.getItemId() == R.id.menu2) {
        // ....
    }
    return super.onContextitemSelected(item);
}
```

* 화면을 문자열 이미지 등의 메인 데이터로 구성하고, 화면에서의 이벤트를 위한 버든을 메뉴를 이용해 별도로 제공하면 화면이 간단해진다. 머터리얼 디자인(Material Design) 철학에서도 강조하는 부분이다.

▶ 액션 버든: 오버플로 메뉴 중 사용자 관점에서 빈번하게 사용되는 메뉴는 ActionBar에 아이콘으로 올려서 바로 사용자가 클릭할 수 있게 할 수 있다. 액션 버든도 메뉴처럼 <item> 태그로 추가하는데, 차이는 app:showAsAction 속성값으로 오버플로 메뉴가 나올지 액션 버든으로 나올지가 결정된다.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <!---중략----->
    <item
        android:id="@+id/menu4"
        android:icon="@android:drawable/ic_menu_send"
        android:title="Actions"
        app:showAsActions="always"/>
</menu>
```

never	기본 값이며, showAsAction을 선언하지 않은 경우. 항상 오버플로 메뉴로 구성.
always	항상 액션 버든으로 구성.
ifRoom	ActionBar에 아이콘이 올라갈 공간이 있으면 액션 버든으로 나타나고, 그렇지 않으면 오버플로 메뉴로 나타남. 자동 판단.

▶ ActionView: ActionBar에 제공되는 뷰이다. 개발자가 직접 준비하지 않고 ActionBar에서 제공하는 뷰를 그대로 이용할 수 있어서 편리하다. 대표적인 ActionView에는 SearchView가 있으며, 사용자 검색을 제공할 때 유용하게 사용된다. ActionView에는 메뉴 기법이 이용된다. SearchView를 이용하려면 메뉴 XML 파일에 다음처럼 <item> 태그를 하나 준비한다.

```
<item
    android:id="@+id/menu_seach"
    android:title="search"
    android:icon="@android:drawable/ic_menu_search"
    app:showAsAction="always"
    app:actionViewClass="androidx.appcompat.widget.SearchView"/>
```

나머지는 액션 버든을 등록할 때와 같지만, SearchView를 등록할 때 app:actionViewClass = "androidx.appcompat.widget.SearchView"를 이용하는 부분이 다르다. 이 메뉴 설정만으로 ActionBar의 SearchView를 이용할 수 있다. 기본 검색 아이콘으로 출력되었다가 사용자 클릭으로 검색어 입력을 위한 SearchView가 나타난다. 검색 작업을 진행하려면 SearchView 객체를 자바 코드에서 획득해야 한다. SearchView는 메뉴 기법으로 이요하므로, 자바 코드에서 획득하려면 먼저 SearchView가 등록된 MenuItem 객체를 얻어야 한다. 그리고 그 MenuItem에 등록된 SearchView 객체를 획득하는 방법을 이용한다. 그리고 나서 사용자의 검색 입력 이벤트 처리를 위해 OnQueryTextListener을 구현한 클래스를 준비하면 된다. OnQueryTextChange() 함수는 사용자가 검색어를 한 자 한 자 입력할 때마다 반복해서 호출되며, 입력한 검색어가 매개변수로 전달된다. 사용자가 검색어를 모두 입력 후 검색을 위해 키보드에서 제공하는 검색 버든을 누든 순간에 호출된다. 최종 검색어는 매개변수로 전달되므로 이 데이터로 다양한 검색 작업을 진행하면 된다.

```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflator().inflate(R.menu.menu_main, menu);
    //id 값으로 MenuItem 객체 획득
    MenuItem menuItem = menu.findItem(R.id.menu_search);
    // MenuItem에 포함된 SearchView 객체 획득
    // androidx.appcompat.widget.SearchView
    searchView = (SearchView) menuItem.getActionView();
    // 검색 힌트 문자열 지정
    searchView.setQueryHint(getResources().getString(R.string.query_hint));
    // 사용자 검색어 입력 이벤트 등록
    searchView.setOnQueryTextListener(queryTextListener);
    return super.onCreateOptionsMenu(menu);
}
```

```java
// 사용자 검색 이벤트 핸들러
private SearchView.onQueryTextListener queryTextListener = new SearchView.OnQueryTextLis
    // 키보드에서 검색 버튼이 눌린 순간의 이벤트
    @Override
    public boolean onQueryTextSubmit(String query) {
        // ....
        return false;
    }

    // 검색 글이 한 자 한자 입력 되는 순간마다
    @Override
    public boolean onQueryTextChange(String newText) {
        return false;
    }
};
```

```java
// 메뉴 XML 파일(res > new > Android Resource Directory - Resources type: menu)
// menu > new > Menu Resources File - 파일명: menu_main
```

```
// menu_main.xml

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/menu_search"
        android:title="search"
        android:icon="@android:drawable/ic_menu_search"
        app:showAsAction="always"
        app:actionViewClass="androidx.appcompat.widget.SearchView"/>
    <item
        android:id="@+id/menu_save"
        android:title="save"
        android:icon="@android:drawable/ic_menu_save"
        app:showAsAction="always"/>
    <item
        android:id="@+id/menu_setting"
        android:title="setting"/>



</menu>
```

```
// strings.xml

<resources>
    <string name="app_name">Part3-8</string>
    <string name="query_hint">검색어를 입력하세요</string>
</resources>
```

```
// MainActivity.java

package com.android.practiceactionbar;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    SearchView searchView;

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
```

```
        MenuItem menuItem = menu.findItem(R.id.menu_search);
        searchView = (SearchView) menuItem.getActionView();
        searchView.setQueryHint(getResources().getString(R.string.query_hint));
        searchView.setOnQueryTextListener(queryTextListener);
        return super.onCreateOptionsMenu(menu);
    }

    SearchView.OnQueryTextListener queryTextListener = new SearchView.OnQueryTextListene
        @Override
        public boolean onQueryTextSubmit(String query) {
            searchView.setQuery("", false);
            searchView.setIconified(true);
            Toast t = Toast.makeText(MainActivity.this, query, Toast.LENGTH_SHORT);
            t.show();
            return false;
        }

        @Override
        public boolean onQueryTextChange(String newText) {
            return false;
        }
    };

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.getItemId() == R.id.menu_save) {
            Toast t = Toast.makeText(this, "save click", Toast.LENGTH_SHORT);
            t.show();
        } else if (item.getItemId() == R.id.menu_setting) {
            Toast t = Toast.makeText(this, "settings click", Toast.LENGTH_SHORT);
            t.show();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

● ToolBar: ActionBar에서 발전된 형태로 ActionBar은 액티비티를 위한 창 출력 시 자동으로 제공되지만, ToolBar은 개발자 뷰이다. 즉, 개발자가 창의 content 영역에 일반 뷰를 이용하여 화면을 구성하듯이, 직접 뷰로 이용할 수 있는 ActionBar이다. ActionBar은 액티비티의 창이 출력되면서 자동으로 출력되고, Toolbar은 개발자가 레이아웃 XML 파일에 명시해야 출력되는 뷰이다. ToolBar을 굳이 사용하는 이유는 개발자가 일반 뷰처럼 제어하면 훨씬 더 다양한 화면을 구성할 수 있기 때문이다. Toolbar을 사용하려면 액티비티가 창을 준비할 때 ActionBar을 출력하지 않게 해야 한다.

```
<style name = "Theme.Toolbar" parent = "Theme.MaterialComponents.DayNight.NoActionBar">
    <!-- 생략 -->
</style>
```

그리고 액티비티를 위한 레이아웃 XML 파일에 Toolbar을 등록한다.

```
<androidx.appcompat.widget.Toolbar
    android:id = "@+id/toolbar"
    android:layout_width = "match_parent"
    android:layout_height = "wrap_content"
    style = "@style/Widget.MaterialComponents.Toolbar.Priamry" />
/>
```

이렇게 명시한 ToolBar을 자바 코드에서 획득해 ActionBar에 적용할 메뉴, 타이틀 문자열들이 그대로 Toolbar에 적용되게 한다.

```
setSupportActionBar(binding.toolbar);
```

Resources: 깡쌤의 안드로이드 프로그래밍