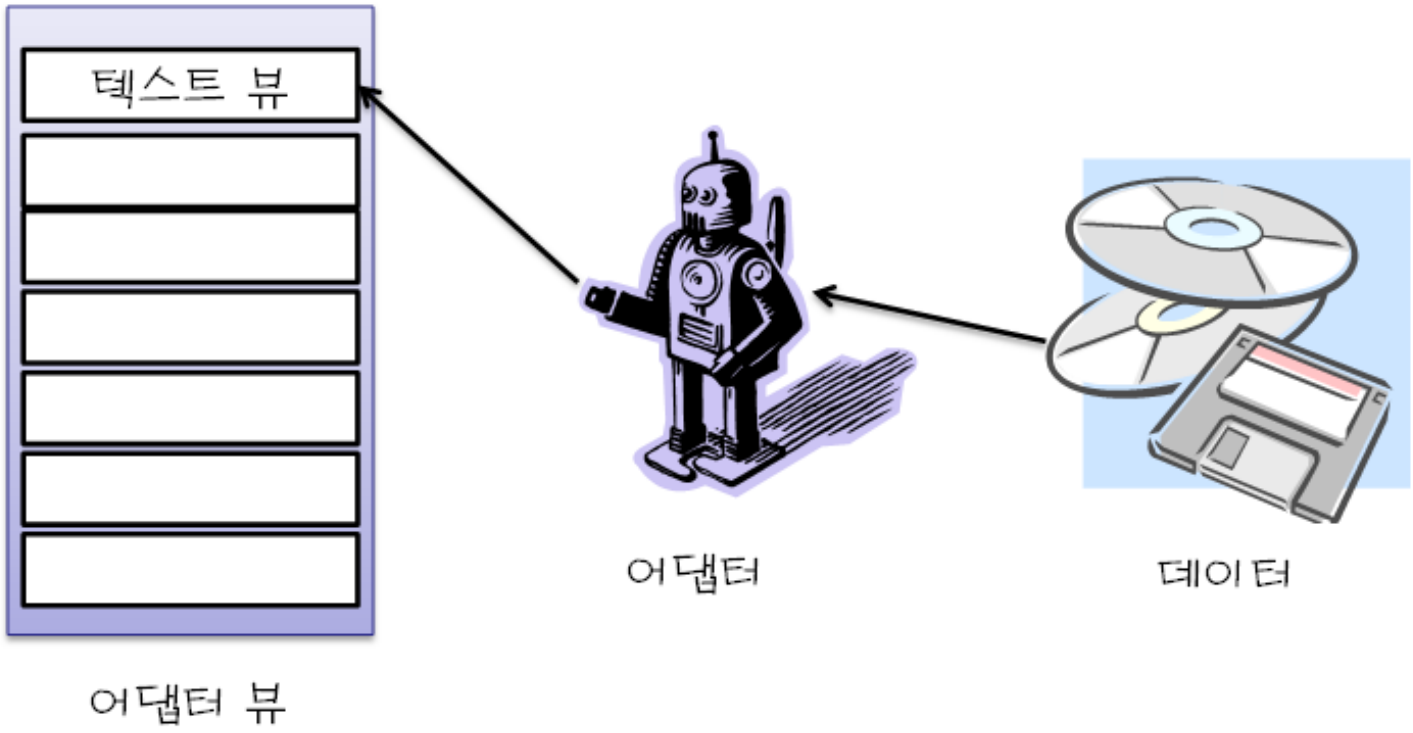
AdapterView

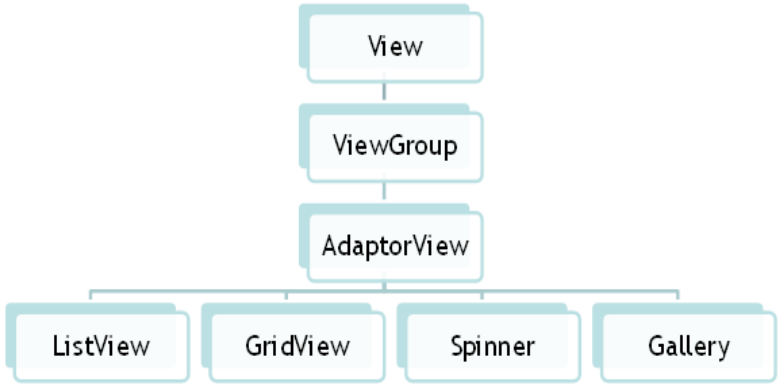
AdapterView

- 배열이나 파일, 데이터베이스에 저장된 데이터를 화면에 표시할 때 유용한 뷰



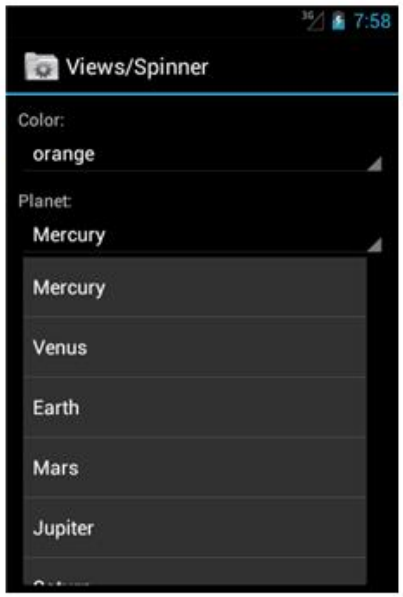
AdapterView

- AdapterView의 종류
  - ListView, Gallery, GridView, Spinner



리스트 뷰

갤러리



스피너

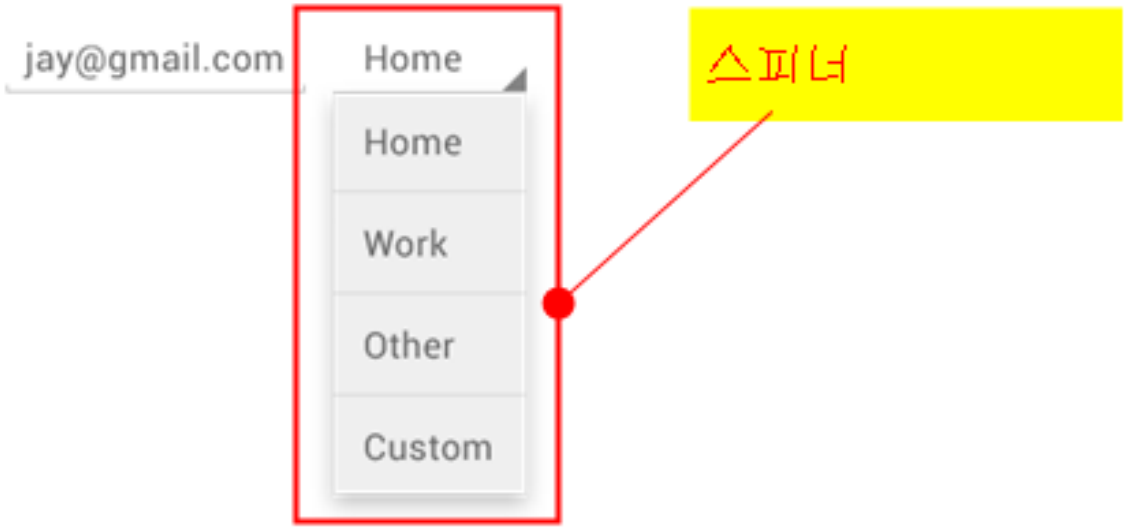
그리드 뷰

AdapterView

Spinner

Spinner

- 항목을 선택하기 위한 드롭 다운 리스트



Spinner

- XML

spinnerdata.xml ➔ values 폴더 선택후 values resource 파일

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="country_prompt">Choose a
country</string>
<string-array name="country_arrays">
<item>Malaysia</item>
<item>United States</item>
<item>Indonesia</item>
<item>France</item>
<item>Italy</item>
<item>Singapore</item>
<item>New Zealand</item>
<item>India</item>
</string-array>
</resources>
```

```
<Spinner
android:id="@+id/spinner1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:entries="@array/country_arrays"
android:prompt="@string/country_prompt" />

<Spinner
android:id="@+id/spinner2"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
```

Spinner

```
val adapter  = ArrayAdapter<String>(this,
android.R.layout.simple_spinner_dropdown_item,
ArrayList<String>())

adapter.add("Item1")
adapter.add("Item2")
adapter.add("Item3")
adapter.add("Item4")

spinner2.adapter = adapter
```

Spinner

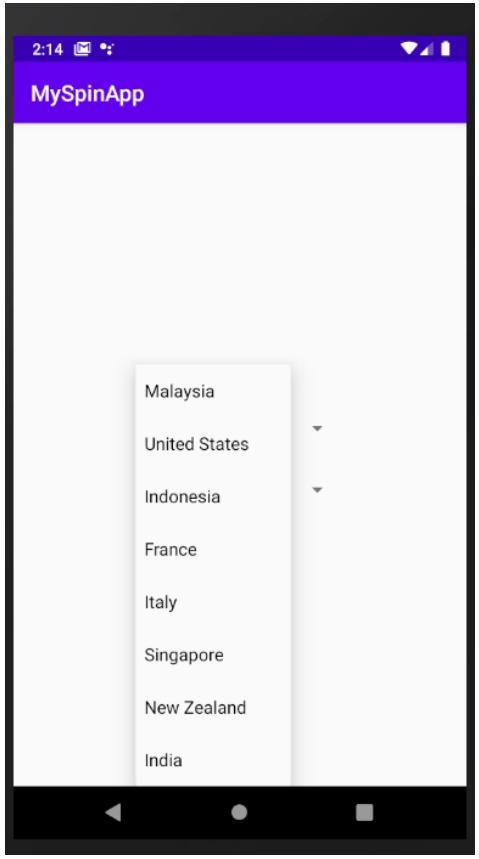
```
inner class CustomOnItemSelectedListener :
AdapterView.OnItemSelectedListener{
override fun onNothingSelected(parent: AdapterView<*>?) {

}
override fun onItemSelected(parent: AdapterView<*>?, view: View?,
position: Int, id: Long) {
        Toast.makeText(parent?.context,
        parent?.getItemAtPosition(position).toString(),
        Toast.LENGTH_SHORT).show()
}
override fun onItemClick(parent: AdapterView<*>?,
                         view: View?, position: Int, id: Long) {
}
}
```

*선택항목 : spinner1.*selectedItem*

Spinner

spinner1.setOnItemSelectedListener(CustomOnItemSelectedListener())

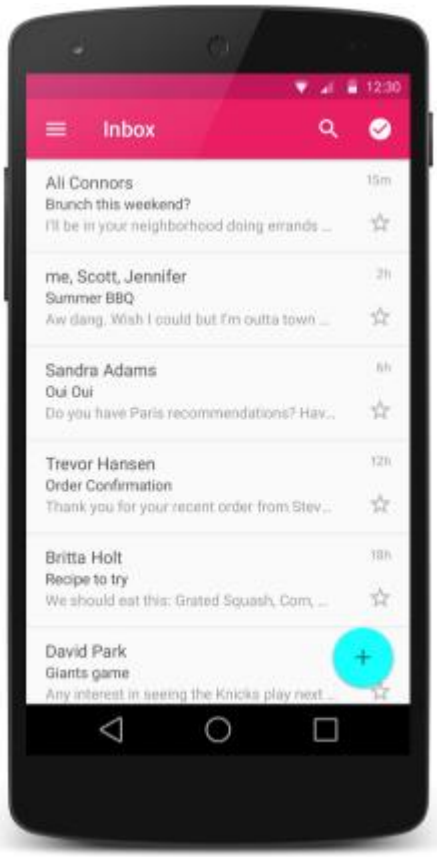
spinner2.setOnItemSelectedListener(CustomOnItemSelectedListener())



RecyclerView

RecyclerView

- RecyclerView
  - ListView보다 향상되고 유연해진 위젯
  - 한정된 수의 뷰를 유지함으로써 매우 효율적으로 스크롤할 수 있음
  - 큰 데이터 집합을 표시하기 위한 컨테이너
    - 사용자 작업 또는 네트워크 이벤트에 따라 런타임에 요소가 변경되는 데이터 컬렉션이 있는 경우 사용하면 효율적임.

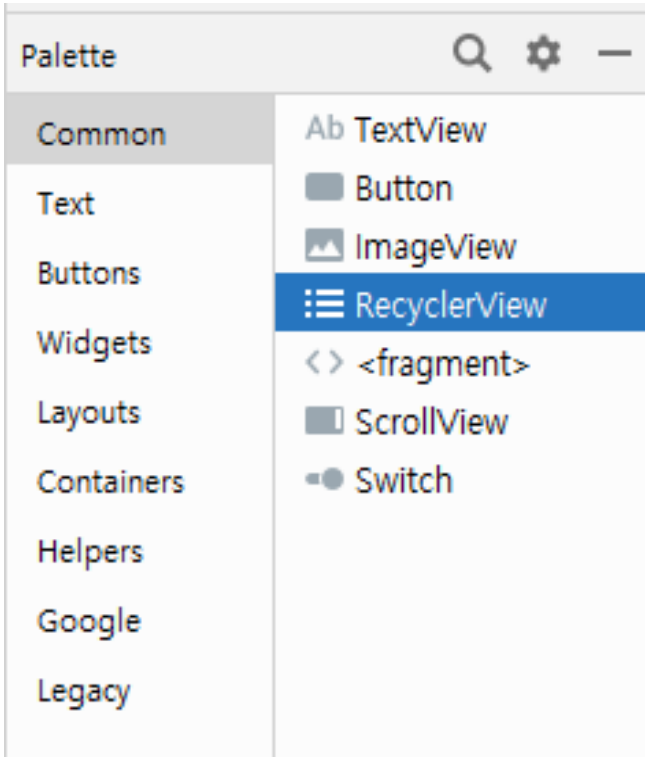


RecyclerView

- RecyclerView
  - 화면에 ChildView 생성
- LayoutManager의 기능 사용
  - 화면에 ChildView 배치
    - 가로, 세로로 배치 가능
- Adapter이용하여 RecyclerView에 데이터 제공
  - ViewHolder 생성 및 데이터 설정

RecyclerView

- RecyclerView 생성




```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

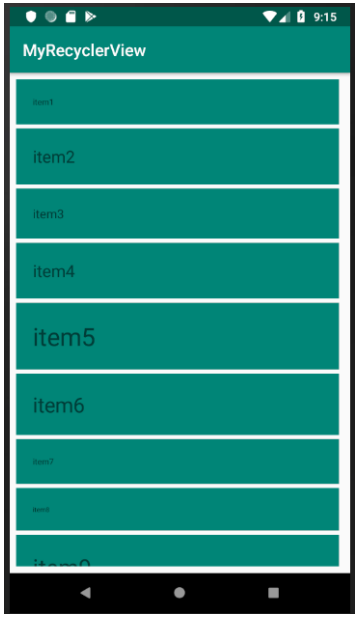
RecyclerView

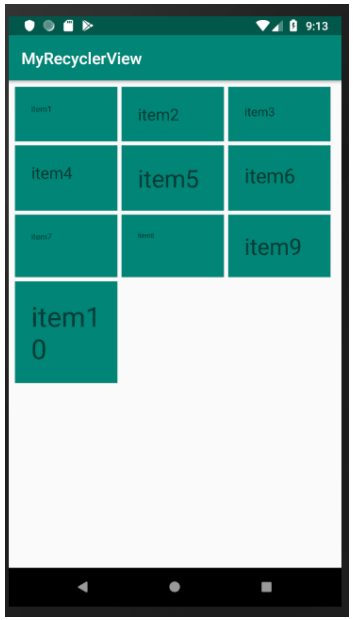
- RecyclerView의 레이아웃 관리자
  - 아이템 View의 크기 및 위치를 결정해 주는 역할 수행
  - LinearLayoutManager
    - 가로, 또는 세로 스크롤 목록 표시
  - GridLayoutManager
    - 그리드 형식으로 항목을 표시
  - StaggeredGridLayoutManager
    - 지그재그형의 그리드 형식으로 항목 표시

```
recyclerView.layoutManager = LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false);
recyclerView. layoutManager = GridLayoutManager(this, 3);
recyclerView. layoutManager = StaggeredGridLayoutManager(3, StaggeredGridLayoutManager.VERTICAL);
```

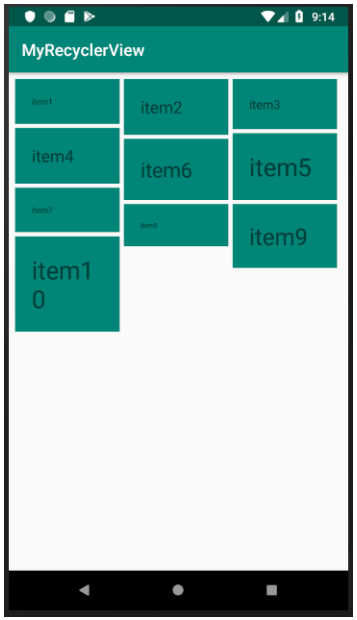
RecyclerView

LinearLayoutManager





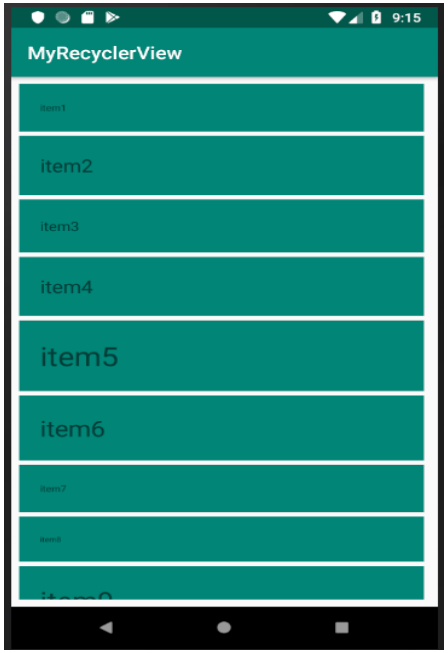
StaggeredGridLayoutManager



GridLayoutManager

RecyclerView

- Adapter 클래스
  - RecyclerView에 표시될 View를 공급하는 클래스
    - **onCreateViewHolder**에서 ViewHolder를 생성하여 반환
    - **onBindViewHolder**에서 ViewHolder에 값을 설정
    - **getItemCount**로 아이템의 개수 반환



- RecyclerView.ViewHolder 클래스
  - RecyclerView에 배치되는 View의 설정
  - onCreateViewHolder에 의해 생성됨
    - ViewHolder에서 관리할 View를 생성자에 전달하여 객체 생성

RecyclerView

- Adapter 클래스 생성

```
class MyAdapter(var items:ArrayList<String>)
                                        :RecyclerView.Adapter<MyAdapter.ViewHolder>(){
inner class ViewHolder(itemView:View) : RecyclerView.ViewHolder(itemView) {
        public var textView:TextView
        init{
           textView = itemView.findViewById(R.id.textView)
        }
    }
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):ViewHolder{
    val v = LayoutInflater.from(parent.context)
                .inflate(R.layout.my_layout, parent, false)
        return ViewHolder(v)
    }
    override fun getItemCount(): Int {
        return items.size
    }
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
            holder.textView.text = items.get(position)
    }}
```

RecyclerView

- Adapter 클래스 생성
  - viewBinding 적용

```
class MyAdapter(var items:ArrayList<String>)
                                        :RecyclerView.Adapter<MyAdapter.ViewHolder>(){
inner class ViewHolder(val binding:MyLayoutBinding) :
                                        RecyclerView.ViewHolder(binding.root) {}
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):ViewHolder{
    val v = MyLayoutBinding.inflate(LayoutInflater.from(parent.context),
                                                    parent, false)
    return ViewHolder(v)
}
override fun getItemCount(): Int {
    return items.size
}
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.binding.textView.text = items[position].textString
}}
```

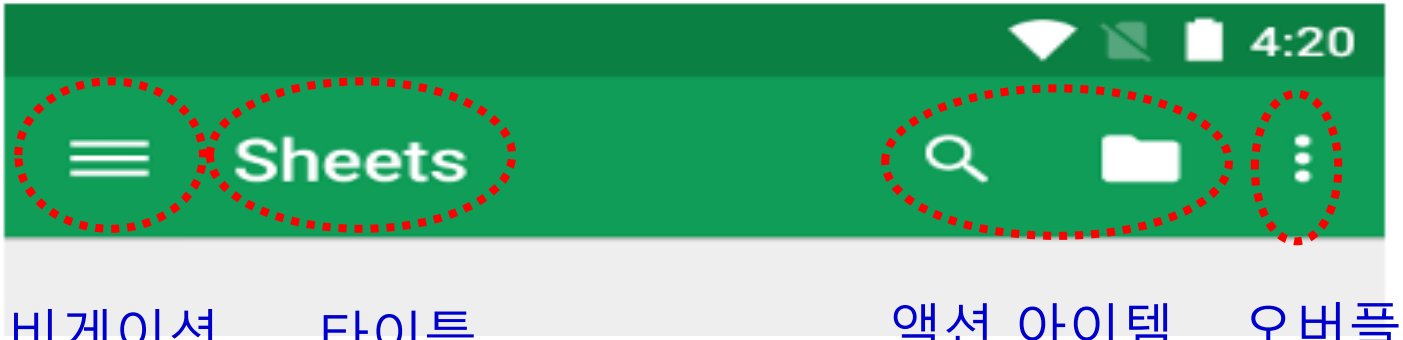
→ body 에서 설정

RecyclerView

메뉴

안드로이드 메뉴

- **AppBar** 구조




```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

안드로이드 메뉴

- 옵션 메뉴
  - 액션 바에 부착되고, 시스템 메뉴버튼을 눌렀을 때 보이는 메뉴
  - 각 화면마다 설정할 수 있음

```
fun onCreateOptionsMenu(menu: Menu?): Boolean
fun onOptionsItemSelected(item: MenuItem): Boolean
```

- 컨텍스트 메뉴
  - 화면을 길게 눌러야 나타나는 팝업 형태의 메뉴
  - 뷰에 설정할 수 있는 메뉴

```
fun onCreateContextMenu(menu: ContextMenu?,v: View?,
                        menuInfo: ContextMenu.ContextMenuInfo?)

fun onContextItemSelected(item: MenuItem): Boolean

fun registerForContextMenu(view:View)
```

옵션 메뉴

- 앱의 상단 타이틀 부분(Action Bar)에 포함되는 메뉴로, 시스템 메뉴 버튼을 눌렀을 때 보이는 메뉴
- 1) 메뉴 리소스 추가 (res >> Android Resource File >> Menu)

```
<?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/new_game"
        android:icon="@drawable/ic_new_game"
        android:title="@string/new_game"
        android:showAsAction="ifRoom"/>
   <item android:id="@+id/help"
        android:icon="@drawable/ic_help"
        android:title="@string/help" />
</menu>
```

옵션 메뉴

- Item 추가
  - android:id : 리소스 ID
  - android:title : 메뉴에 표시되는 글자
  - android:icon : 아이콘으로 표현하고 싶을 때
  - android:showAsAction : 메뉴를 보여주는 방식

- **showAsAction 속성**
  - always : 항상 액션바에 아이템을 표시
  - never : 액션바에 아이템을 추가하여 표시하지 않고, 오버플로우 메뉴에 표시
  - ifRoom : 액션바에 여유공간이 있을 때만 아이템 표시
  - withText : title 속성으로 설정된 제목을 같이 표시
  - collapseActionView : 아이템에 커스텀 액션 뷰가 지정된 경우, 축소형태로 표시

옵션 메뉴

- onCreateOptionsMenu : 액티비티에 메뉴를 지정하는 함수

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    val inflater: MenuInflater = menuInflater
    inflater.inflate(R.menu.game_menu, menu)
    return true
}
```

- onOptionsItemSelected : 이벤트 처리

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    // Handle item selection
    return when (item.itemId) {
        R.id.new_game -> {
            newGame()
            true
        }
        R.id.help -> {
            showHelp()
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}
```

RecyclerView

이벤트 처리

File에서 데이터 읽어오기

- 내부 저장장치에서 파일 읽어오기

*파일 위치 : res/raw/datafile.txt

```
val scan = Scanner(resources.openRawResource(R.raw.datafile))

while(scan.hasNextLine()){
    val line = scan.nextLine()
}

scan.close();
```

*raw : 안드로이드 시스템에 의한 압축이나 변형없이 그대로 저장될 파일

RecyclerView Listener 만들기

- **Adapter 클래스**
  - 인터페이스 정의하기 및 멤버 선언하기

```
interface OnItemClickListener{
   fun OnItemClick(holder:ViewHolder, view:View, data:DataType, position: Int )
}

var itemClickListener : OnItemClickListener? = null
```

  - ViewHolder 클래스내 View에 **OnClickListener** 달기

```
itemView.setOnClickListener{
     val position = adapterPosition
     itemClickListener?.OnItemClick(this, it, items[position], position)
}
```

RecyclerView Listener 만들기

- **Activity 클래스**
  - Adpater 객체의 인터페이스 객체(ex, itemClickListener) 생성

```
adapter.itemClickListener = object : MyAdapter.OnItemClickListener{
    override fun OnItemClick(holder: MyAdapter.ViewHolder,
                             view: View,
                                data: DataType,
                                position: Int) {
      //TODO ~~~
    }
}
```

Text To Speech

- TextToSpeech 클래스
  - https://developer.android.com/reference/android/speech/tts/TextToSpeech.html
  - 간단하게 사용하는 방법
    - TextToSpeech 클래스 객체 생성
    - Speak 함수 호출
  - AndroidManifest.xml 파일 (android 11 이상)

```
<queries>
   <intent>
      <action android:name="android.intent.action.TTS_SERVICE" />
   </intent>
</queries>
```

Text To Speech

- 생성자 : **TextToSpeech(context, Listener)**
  - Listener는 TTS 서비스가 로딩이 완료되면 호출됨. 즉, TTS의 OnInit method가 호출되기 전까지는 대기.

```
val ttsReady =false

val tts =  TextToSpeech(
        this,
       TextToSpeech.OnInitListener{
              ttsReady = true
       }
)
```

Text To Speech

- Speak Method (읽을 문자열, Mode, RequestParams, RequestID)
  - Mode
    - TextToSpeech.QUEUE_ADD : 현재 큐에 추가
    - TextToSpeech.QUEUE_FLUSH : 현재 큐 무시하고, 추가하기

```
if(ttsReady){
    tts.speak(text, TextToSpeech.QUEUE_ADD, null, null)
}
```

RecyclerView의 Swipe / Drag&Drop

- ItemTouchHelper
  - RecyclerView의 swipe와 drag&drop을 지원해 주는 유틸리티 클래스
  - ItemTouchHelper.Callback 클래스의 OnSwiped와 OnMove 함수를 오버라이딩해 주어야 함
  - ItemTouchHelper.Callback.onMove(RecyclerView, ViewHolder, ViewHolder)
  - ItemTouchHelper.Callback.onSwiped(ViewHolder, int)

```
ItemTouchHelper mIth = new ItemTouchHelper(
    new ItemTouchHelper.SimpleCallback(ItemTouchHelper.UP |
                ItemTouchHelper.DOWN, ItemTouchHelper.LEFT) {

boolean onMove(RecyclerView view, ViewHolder holder, ViewHolder target){}
void onSwiped(ViewHolder viewHolder, int direction) {}

}
)
```

RecyclerView의 Swipe & Drag

```
val simpleItemTouchCallback = object :
ItemTouchHelper.SimpleCallback(ItemTouchHelper.UP or
ItemTouchHelper.DOWN, ItemTouchHelper.LEFT ) {
    override fun onMove(p0: RecyclerView, p1:
RecyclerView.ViewHolder, p2: RecyclerView.ViewHolder): Boolean {
        //ToDO something
        return false
    }

    override fun onSwiped(viewHolder: RecyclerView.ViewHolder,
direction: Int) {
        // ToDO something
    }
}

val itemTouchHelper = ItemTouchHelper(simpleItemTouchCallback)
itemTouchHelper.attachToRecyclerView(recyclerView)
```

수고하셨습니다.