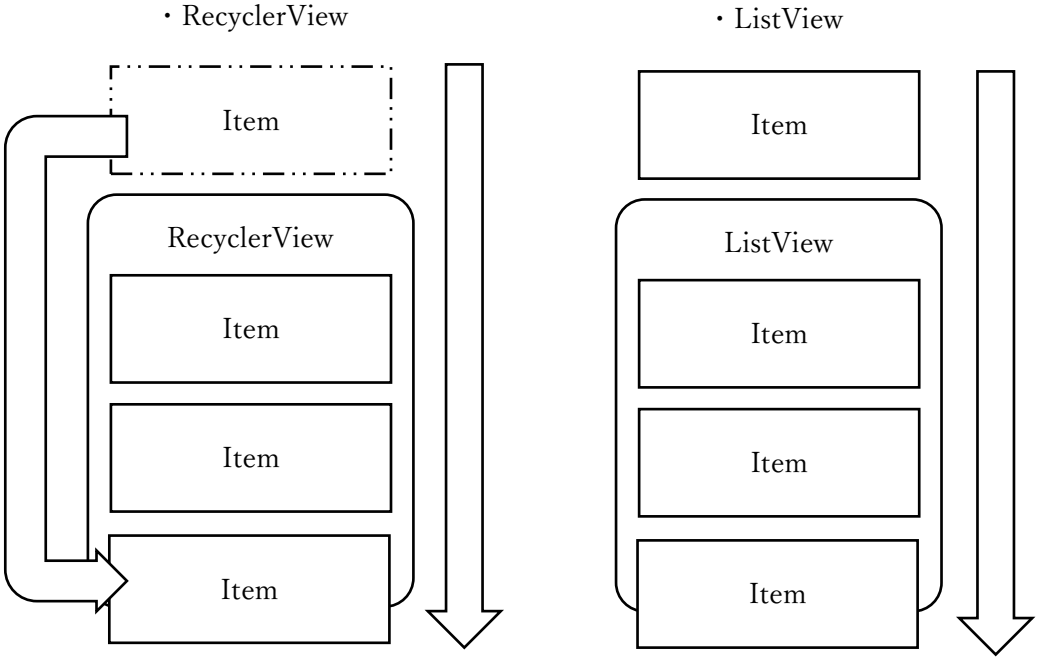
RecyclerView

RecyclerView とは

ビュー内の個々の要素を再利用しながら描画を行うビューグループです。RecyclerView を使用すると、アイテムが画面外にスクロールされてもアイテムのビューを廃棄せず、画面上にスクロールされた新しいアイテムのビューに再利用されます。
そのため、ListView などよりもパフォーマンスの改善、アプリの応答性の向上、消費電力の減少をもたらします。



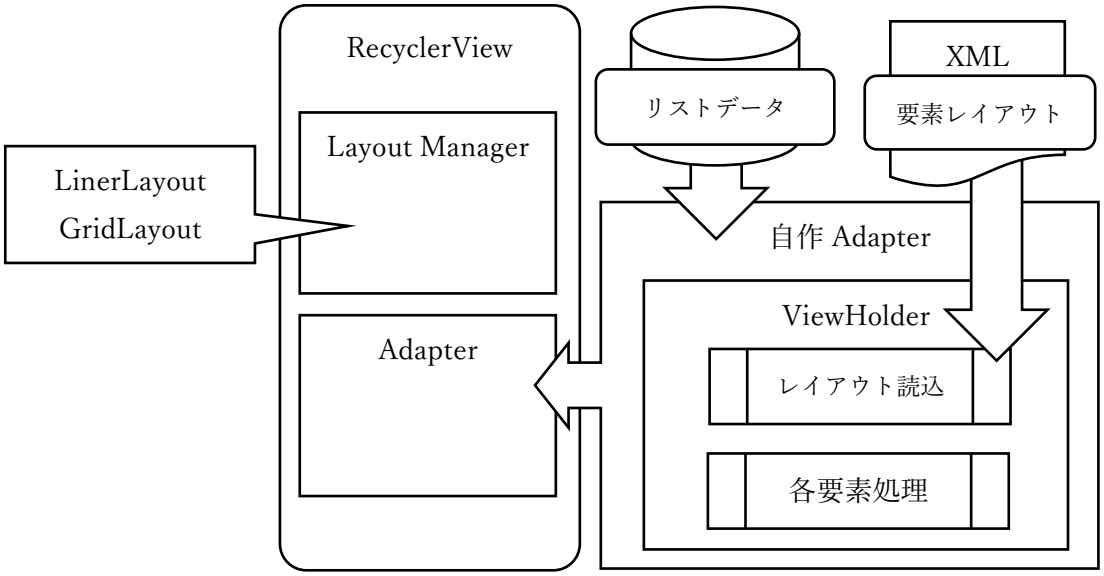
新しいアイテムは、画面外にスクロールされたビューを再利用します。
ListView は、読み込んだアイテムを保持するので大量のデータには向きません。

RecyclerView を実装する手順

RecyclerView を使用する場合は、いくつかの手順が必要になります。

1．リストやグリッドなど RecyclerView で使用する外観を決める。
　通常、RecyclerView の標準レイアウトマネージャーを使用して設定を行う。
2．各要素の表示方法や動作方法を設計する。
　この設計に基づいて ViewHolder クラスを拡張します。
3．データを ViewHolder に関連付ける Adapter を定義する。

・RecyclerView 実装イメージ



レイアウトの設定

RecyclerView のアイテムは LayoutManager クラスによって並び替えられます。
RecyclerView ライブラリには 3 つのレイアウトマネージャーがあり、レイアウトの標準的な状況を処理します。

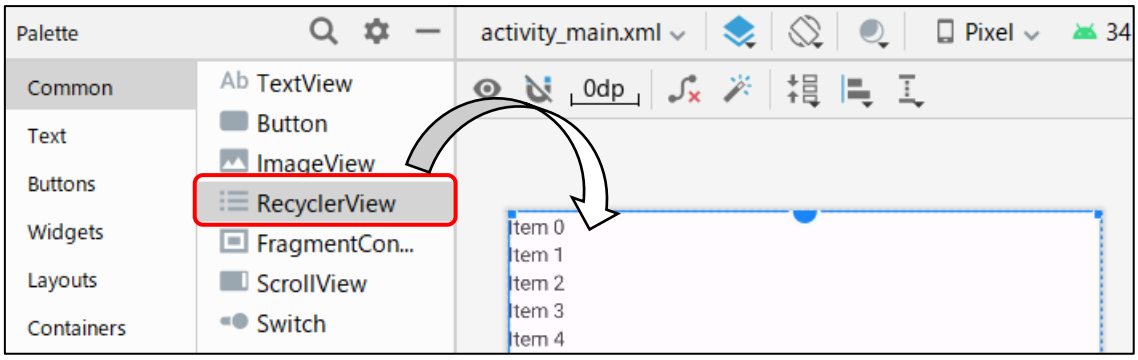
レイアウト方式	概要
LinearLayoutManager	１次元リスト内にアイテムを配置
GridLayoutManager	２次元グリッド内にあるすべてのアイテムを配置
StaggeredGrid LayoutManager	グリッドと似ているが、各アイテムの高さ、または幅をアイテムごとに変更することが可能

ハンズオン　RecyclerView のレイアウトを設定する

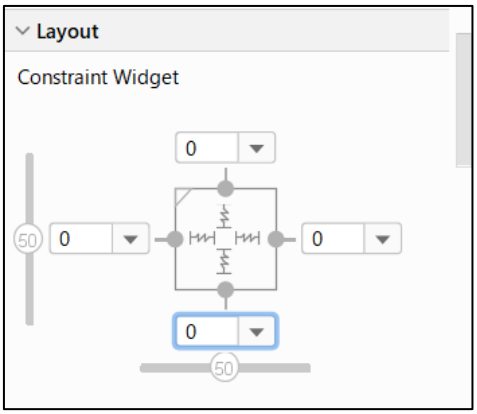
1. 新しいプロジェクトを作成する

テンプレート	Empty Views Activity
Name	HandsOnProject
Language	Kotlin
Minimum SDK	API26 (Android8.0)

2. Activity_main.xml に RecyclerView を配置する



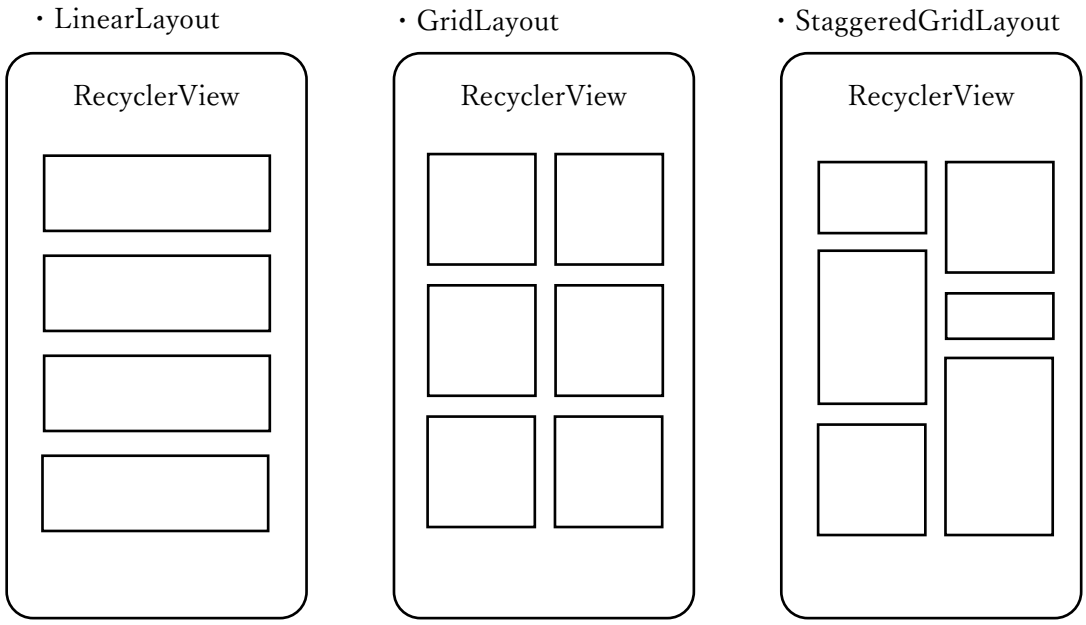
Id	sampleRecyclerView
Constraint Widget	左記のとおり
Layout_width	0dp
Layout_height	0dp



3. Activity で RecyclerView の宣言および、レイアウトとの紐づけを行う

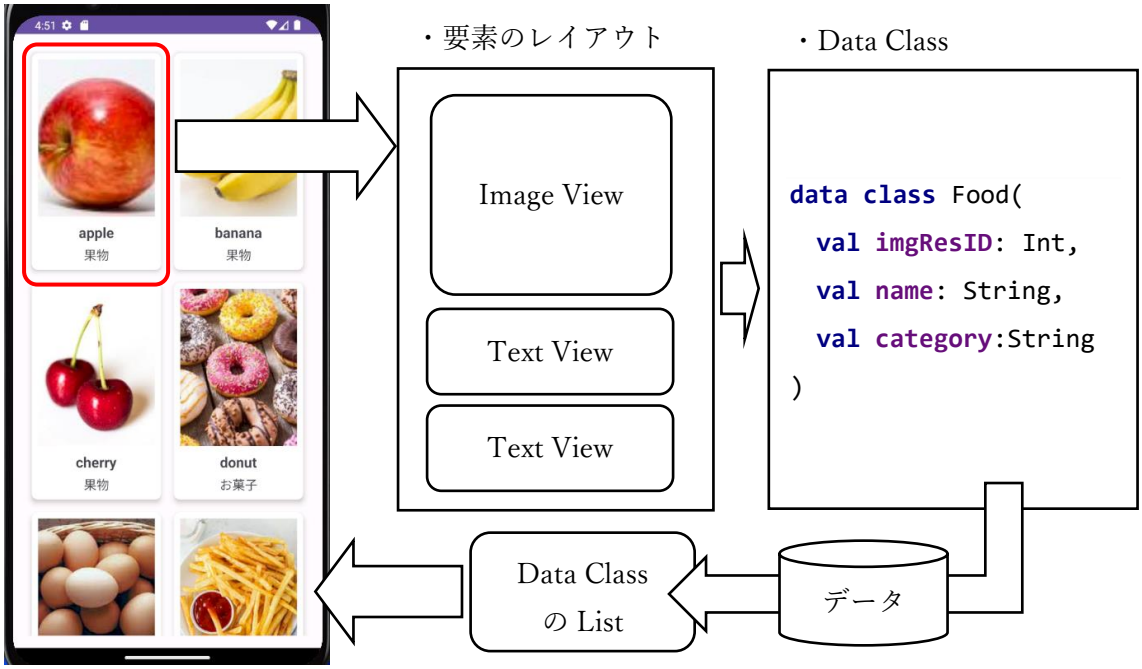
```
class MainActivity : AppCompatActivity() {

    lateinit var srv : RecyclerView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        srv = findViewById(R.id.sampleRecyclerView)
        srv.layoutManager = LinearLayoutManager(applicationContext)
    }
}
```

データおよび Adapter の設定ができていないので画面には、まだ何も表示されません。
各レイアウトマネージャーを使用したレイアウトイメージ図は以下の通りです。



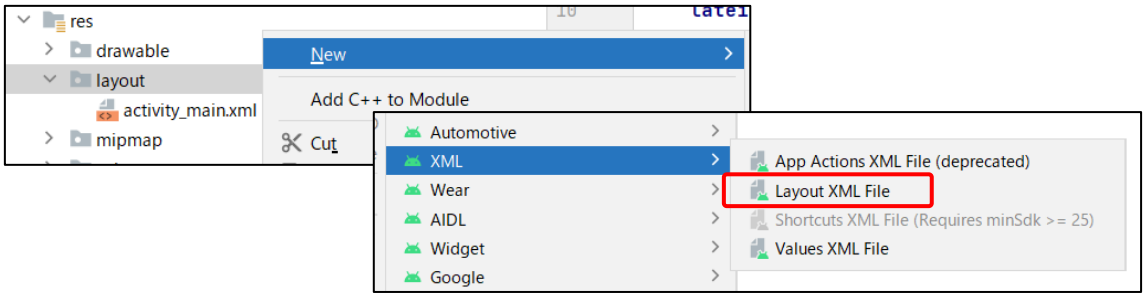
Data Class

Kotlin には、データを保持することを目的とした Data Class が存在します。RecyclerView では、各要素の情報を Data Class にすることで、ひとまとまりとしてリスト化することができます。



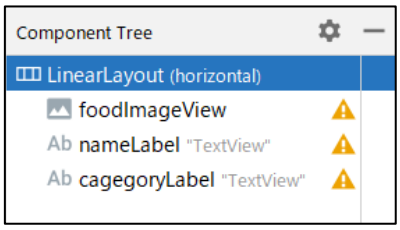
ハンズオン　要素のレイアウトと項目を設定する

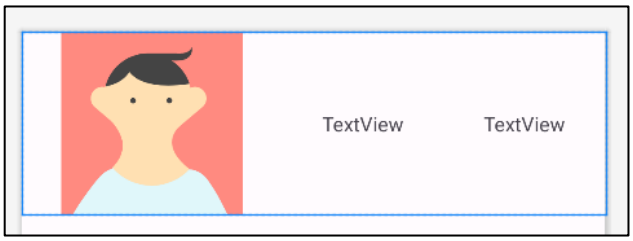
1.　1 件の画面レイアウトファイルを作成する



Layout File Name	list_row
Root Tag	LinearLayout

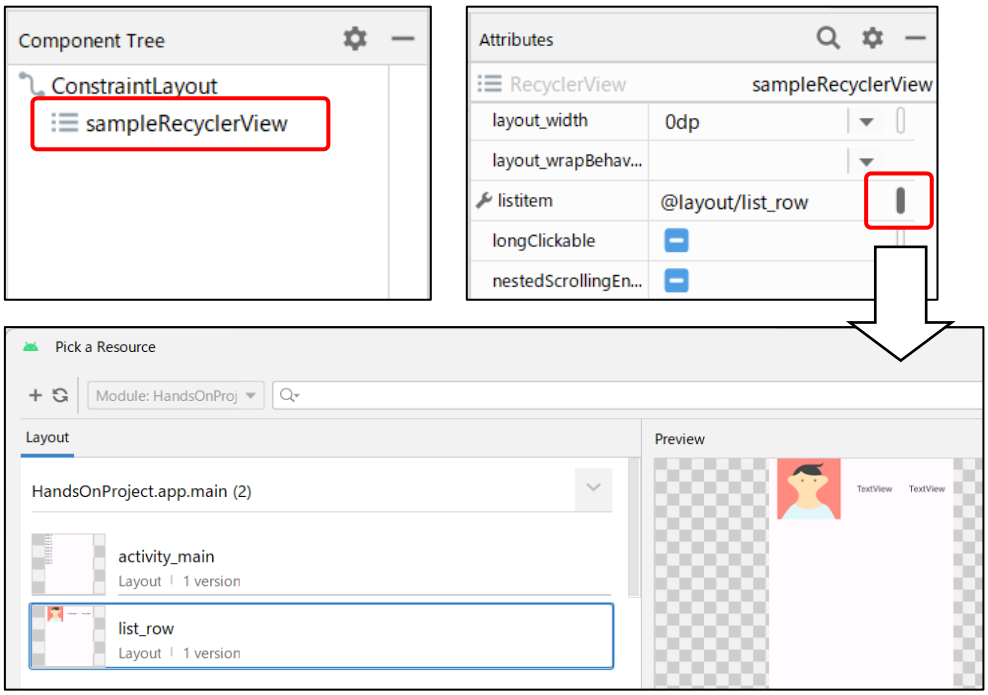
2.　RecyclerView で表示する 1 件の画面デザインを設定する



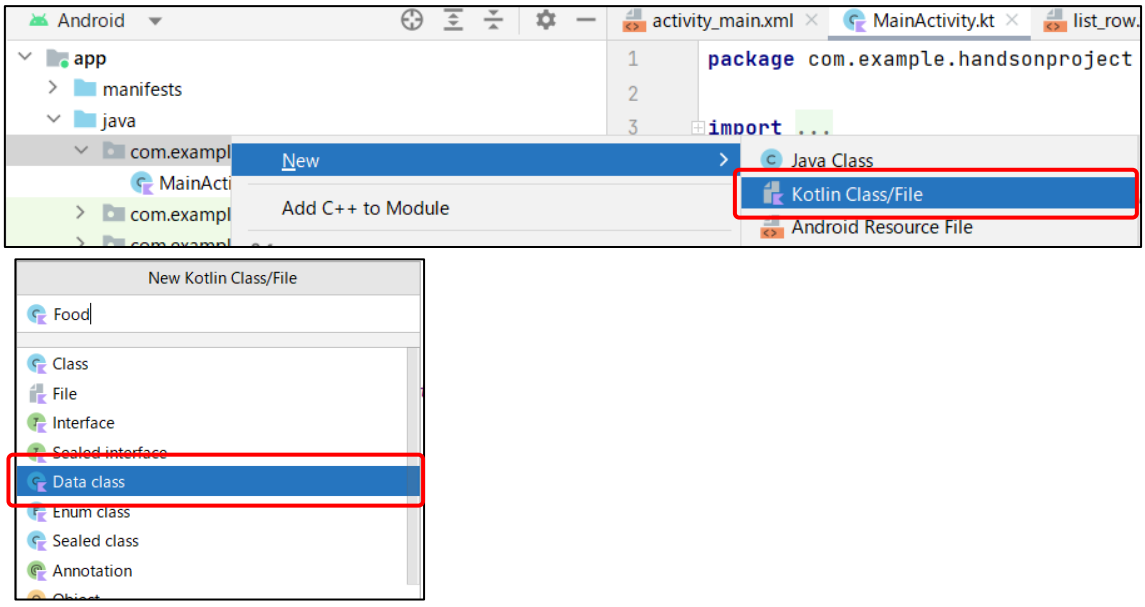


コンポーネント	属性	設定値
LinearLayout	Layout_height	Wrap_content
ImageView	Id	foodImageView
	Layout_width	100dp
	Layout_height	100dp
	scaleType	centerCrop
TextView	Id	nameLabel
	Layout_height	match_parent
	Gravity	Center
TextView	Id	categoryLabel
	Layout_height	match_parent
	Gravity	Center

3. Activity に設置した RecyclerView に展開イメージを表示させる



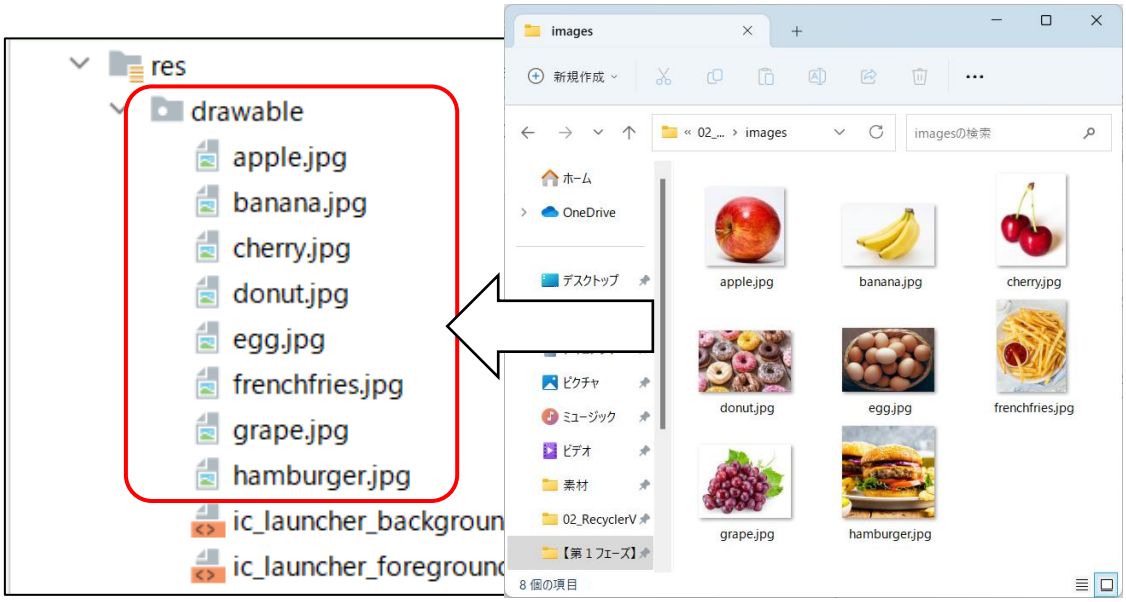
4. Data Class を新規作成する



5. Data Class に 1 件の各項目を宣言する

```
data class Food(
    val imgResID: Int,    // 画像のリソース ID
    val name: String,     // 食べ物の名前
    val category: String  // 食べ物のカテゴリ（例：果物、お菓子など）
)
```

6. テストデータ用の画像をプロジェクトに格納する



7. Activity 内で Food クラスのテストデータを List で用意する

```
// デモ用の食べ物データを作成します。
// このデータは RecyclerView に表示されるものです。
val foods = listOf(
    Food(R.drawable.apple, "apple", "果物"),
    Food(R.drawable.banana, "banana", "果物"),
    Food(R.drawable.cherry, "cherry", "果物"),
    Food(R.drawable.donut, "donut", "お菓子"),
    Food(R.drawable.egg, "egg", "食材"),
    Food(R.drawable.frenchfries, "frenchfries", "料理"),
    Food(R.drawable.grape, "grape", "果物"),
    Food(R.drawable.hamburger, "hamburger", "料理"),
)
```

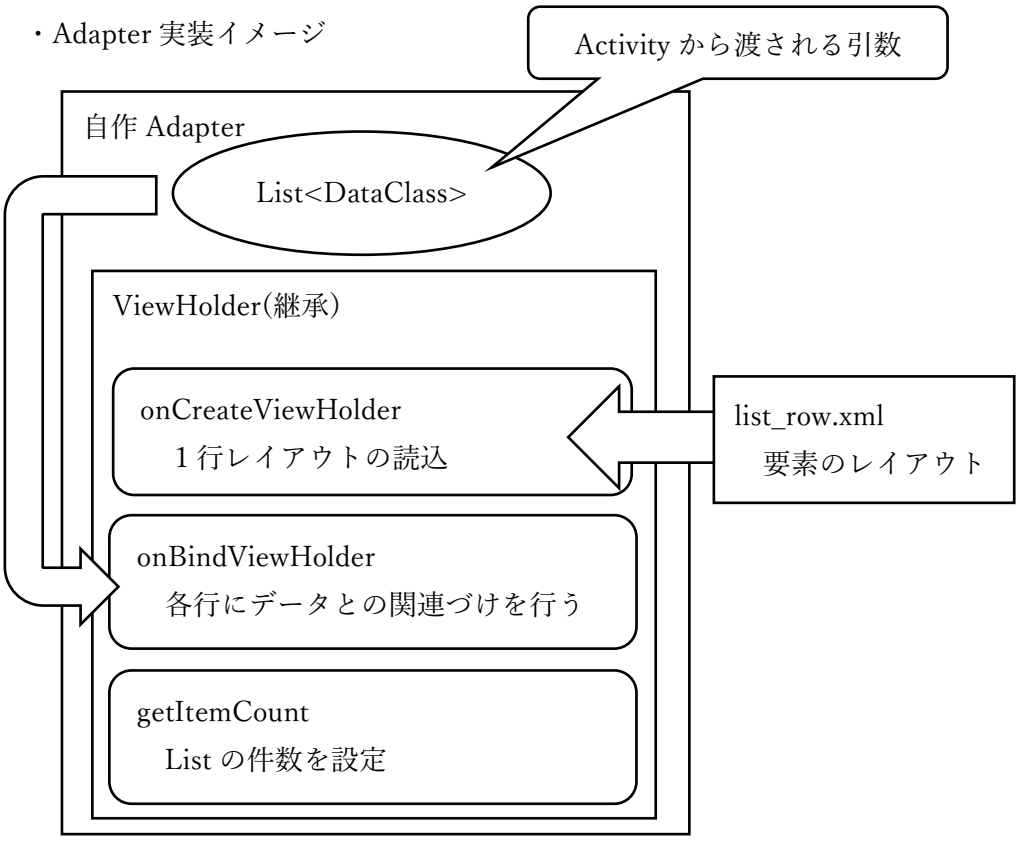
Adapter と ViewHolder の設定

レイアウトが決まったら、Adapter と ViewHolder を実装する必要があります。これら 2 つのクラスが連携して、データの表示方法を定義します。ViewHolder は、リスト内の各アイテムのレイアウトを含む View のラッパーです。Adapter は、必要に応じて ViewHolder オブジェクトを生成するとともに、これらのビューのデータを設定します。
ビューをこれらのデータに関連づけるプロセスをバインディングと呼びます。

Adapter を定義する際は、3 つの主要メソッドをオーバライドする必要があります。

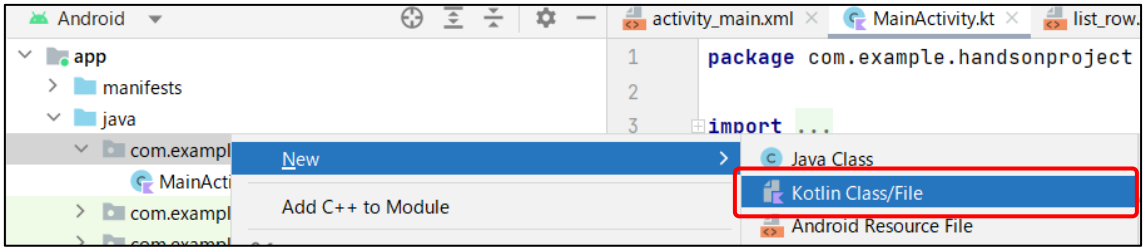
オーバライドメソッド	概要
onCreateViewHolder()	ViewHolder とそれに関連する View の初期化を行う。RecyclerView は新しい ViewHolder を作成する必要があるたびにこのメソッドを呼び出す。
onBindViewHolder()	データを ViewHolder に関連づけるメソッド。 1 件ごとのデータはここで設定が行われる。
getItemCount()	リストサイズ。基本的にデータの合計数を設定する。RecyclerView はこの情報を使用して、表示できるアイテムがないか判断を行う。

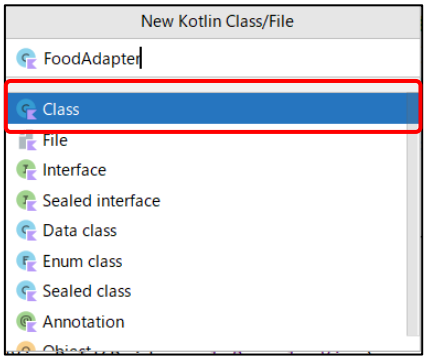
・Adapter 実装イメージ



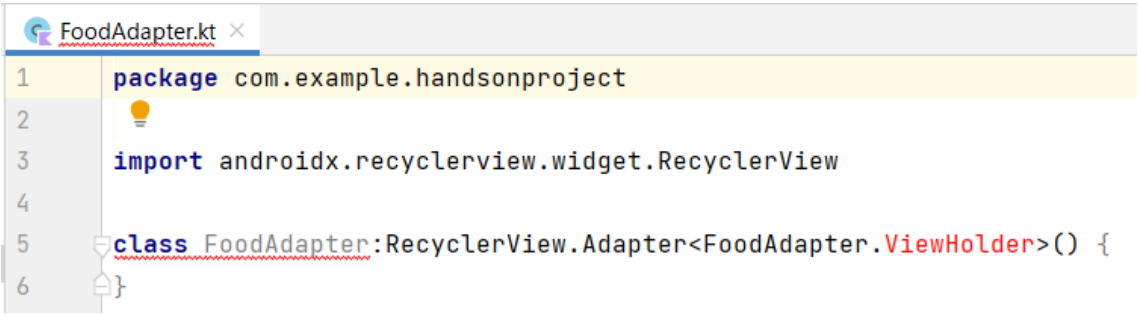
ハンズオン　Adapter の実装を行う

1．RecyclerView 用の Adapter を作成する

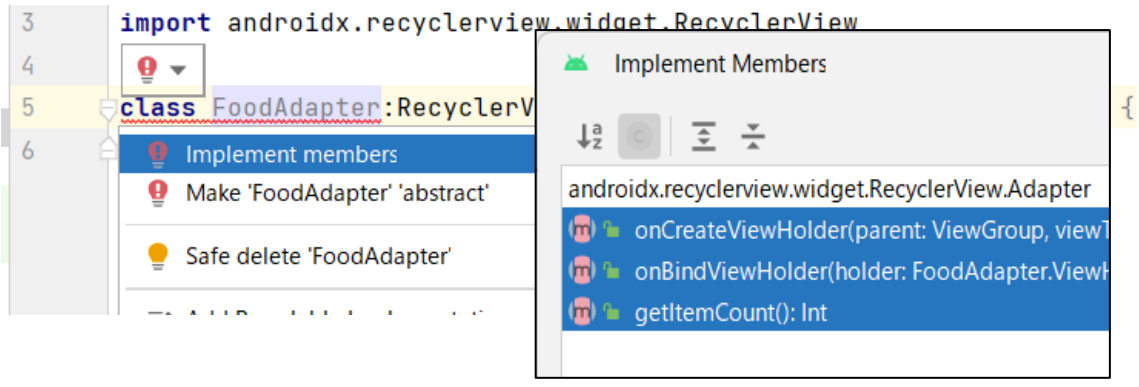




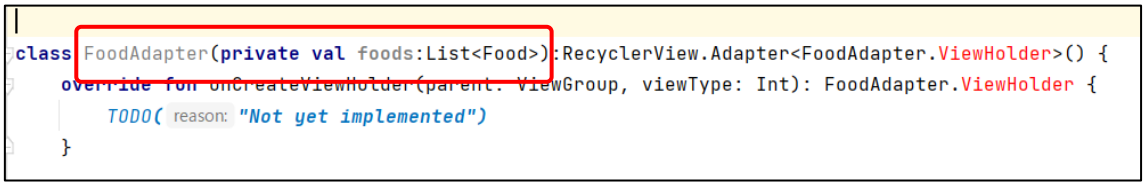
2．Adapter クラスと ViewHolder を継承する



3．オーバライドが必要なメソッドを実装する



4．Adapter で取り扱うデータを引数として受け取る



5．ViewHolder で取り扱うオブジェクトの設定を行う

```
class ViewHolder(item: View) : RecyclerView.ViewHolder(item) {

    val foodImageView : ImageView
    val nameLabel : TextView
    val categoryLabel : TextView

    init {
        foodImageView = item.findViewById(R.id.foodImageView)
        nameLabel = item.findViewById(R.id.nameLabel)
        categoryLabel = item.findViewById(R.id.cagegoryLabel)
    }
}
```

6．onCreateViewHolder 内で 1 件レイアウトの読込を行う

```
override fun onCreateViewHolder(
    parent: ViewGroup, viewType: Int
): FoodAdapter.ViewHolder {
    val view = LayoutInflater.from(parent.context)
                    .inflate(R.layout.list_row, parent, false)
    return ViewHolder(view)
}
```

7．onBindViewHolder 内で各データの紐づけと動作を実装する

```
override fun onBindViewHolder(
    holder: FoodAdapter.ViewHolder, position: Int
) {
    holder.foodImageView.setImageResource(foods[position].imgResID)
    holder.nameLabel.text = foods[position].name
    holder.categoryLabel.text = foods[position].category
}
```

8．getItemCount 内でデータの件数を設定する

```
override fun getItemCount(): Int {
    return foods.size
}
```

ここまで、実装できれば自作 Adapter の完成です。
あとは、Activity にある RecyclerView の Adapter に自作した Adapter をセットすれば RecyclerView の完成です。

9．RecyclerView の Adapter に自作した Adapter をセットする

```
srv.adapter = FoodAdapter(foods)
```