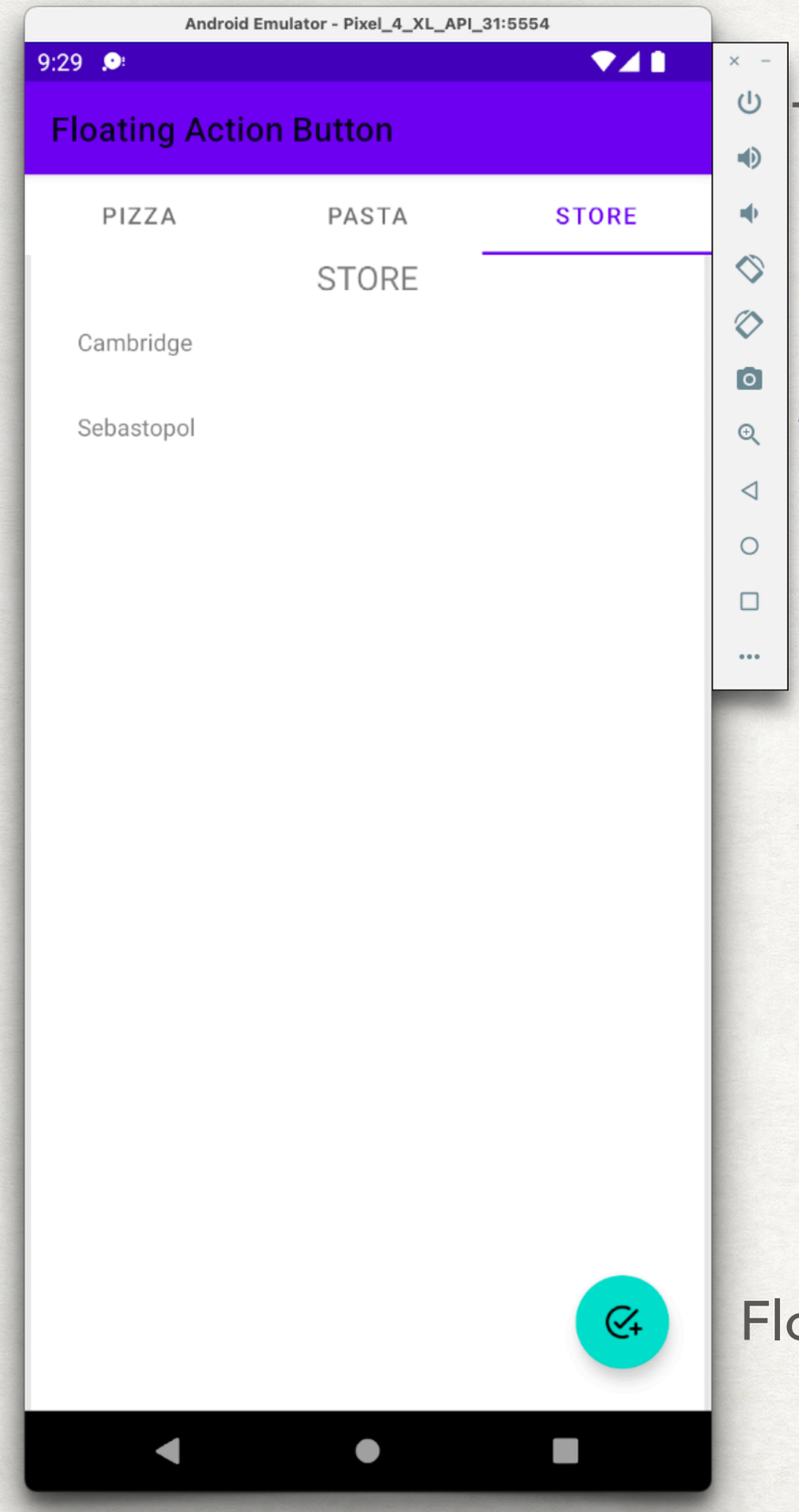
PROGRAMMATION DE COMPOSANTS MOBILES (ANDROID)

WIESLAW ZIELONKA



Toolbar (menu)

TabLayout

ViewPager2 (affichant un fragment)

FloatingActionButton (FAB)

Pager2

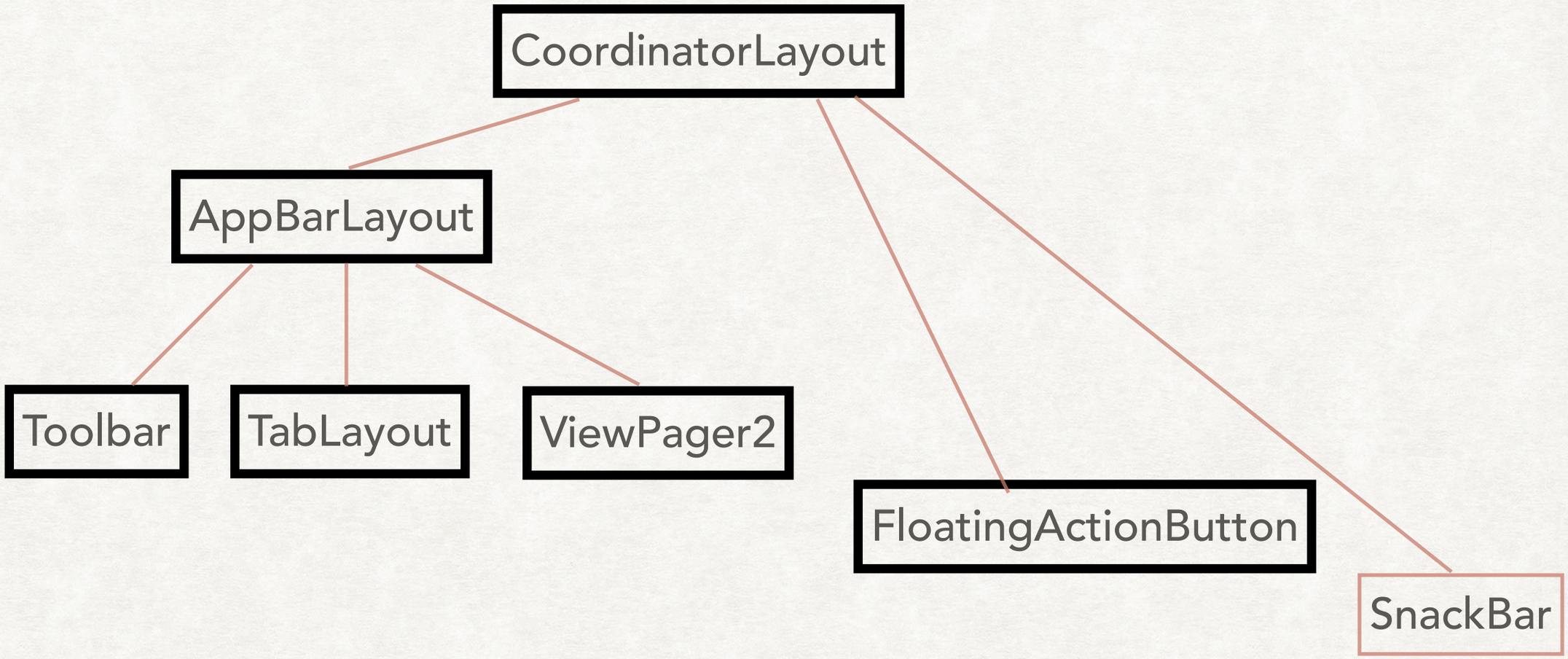
new Project choisir Basic Activity

crée une activité avec FloatingActionButton et avec un fichier layout adapté.

Ajouter dans build.gradle la dépendance

```
implementation "androidx.viewpager2:viewpager2:1.0.0"
```

layout de l'activité



CoordinationLayout à la racine nécessaire pour que toutes les fonctionnalités soient implémentées.
Toolbar implemente le menu.
ViewPager2 contient les fragments. TabLayout permet de choisir le fragment affiché dans ViewPager2.
FloatingActionButton (FAB) fait apparaitre le SnackBar en bas de l'écran

layout de l'activité

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
        ........
        tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
         ....>

         <androidx.appcompat.widget.Toolbar
             .....            />

         <com.google.android.material.tabs.TabLayout
             ..../>

         <androidx.viewpager2.widget.ViewPager2
             .../>

    </com.google.android.material.appbar.AppBarLayout>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
       />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Pager2

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/material_on_primary_disabled">



    </com.google.android.material.appbar.AppBarLayout>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:layout_width="wrap_content"
        android:id="@+id/fab"
        android:layout_height="wrap_content"
        android:layout_gravity="end|bottom"
        android:layout_margin="25dp"
        android:src="@drawable/round_add_task_black_24dp"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

les icônes pour le FAB

https://design.google.com/icons

donne le lien vers un GitHub, on y trouve de centaines icônes.

Dans **app/src/main/res** créer les sous-répertoires drawable-hdpi, drawable-mdpi, drawable-xhdpi, drawable-xxhdpi, drawable-xxxhdpi et copier les icônes de votre choix. Android cherche les icônes en fonction de densité/ taille d'écran.

layout de l'activité

```
<com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/material_on_primary_disabled">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|enterAlways"
            android:elevation="4dp"
            />

        <com.google.android.material.tabs.TabLayout
            android:id="@+id/tabLayout"
            app:layout_scrollFlags="scroll|enterAlways"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>


        <androidx.viewpager2.widget.ViewPager2
          android:id="@+id/pager"
          android:layout_weight="1"
          android:layout_width="match_parent"
          android:layout_height="match_parent" />

</com.google.android.material.appbar.AppBarLayout>
```

Toolbar et TabLayout peuvent sortir de l'écran si le swap vers le haut

l'activité (avec binding activé)

```
class MainActivity : AppCompatActivity() {

    val binding : ActivityMainBinding by lazy{ ActivityMainBinding.inflate(layoutInflater) }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(binding.root)

        setSupportActionBar(binding.toolbar)  /* toolbar comme le menu */

        val names = listOf("PIZZA", "PASTA", "STORE")
        val pizzaNames = resources.getStringArray(R.array.pizzas).toMutableList()
        val pastaNames = resources.getStringArray(R.array.pasta).toMutableList()
        val storeNames = resources.getStringArray(R.array.stores).toMutableList()
        val pizzas = MutableList<PizzaItem>(pizzaNames.size) { PizzaItem(pizzaNames[it]) }
        val pastas = MutableList<PastaItem>(pastaNames.size) { PastaItem(pastaNames[it]) }
        val stores = MutableList<StoreItem>(storeNames.size) { StoreItem(storeNames[it]) }

        /* création de RecyclerView.Adapters pour les trois fragments */
        val pizzaAdapter = PizzaAdapter(pizzas)
        val pastaAdapter = PastaAdapter(pastas)
        val storeAdapter = StoreAdapter(stores)

        /* création de trois fragments */
        val pizzaFragment =
            PizzaItemFragment.newInstance(names[0], pizzaAdapter, R.layout.fragment_layout)
        val pastaFragment =
            PastaItemFragment.newInstance(names[1], pastaAdapter, R.layout.fragment_layout)
        val storeFragment =
            StoreItemFragment.newInstance(names[2], storeAdapter, R.layout.fragment_layout)
```

l'activité (avec binding activé)

```
class MainActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {

   ............... ( page précédente )

   /* création de pagerAdapter qui fournira le contenu à ViewPager2 */
   val pagerAdapter = ScreenSlidePagerAdapter(
        this,
        mutableListOf<Fragment>(pizzaFragment, pastaFragment, storeFragment)
    )
   /* attacher pagerAdapter à ViewPager2 */
   binding.pager.adapter = pagerAdapter


    /* donner les noms au tabs et les attacher aux fragments correspondants*/
    TabLayoutMediator(binding.tabLayout, binding.pager) { tab, position ->
        tab.text = names[position]
    }.attach()

    /* le listen sur le FloatingActionButton */
    binding.fab.setOnClickListener { view ->
        Snackbar.make(view, "Here's a Snackbar", Snackbar.LENGTH_LONG)
            .setAction("Action"){  /* l'action quand on tape sur le SnackBar */  }.show()
    }

   /* les animations pour le movement swap */
   // binding.pager.setPageTransformer( ZoomOutPageTransformer())
   // binding.pager.setPageTransformer( DepthPageTransformer())
}/* fin onCreate() */
```

les fragments à mettre dans ViewPager2

l'activité (avec binding activé)

```
class MainActivity : AppCompatActivity() {
.....

/* onBackPressed appelé quand on active le bouton back */
override fun onBackPressed() {
     if (binding.pager.currentItem == 0) {
     // If the user is currently looking at the first step, allow
the system to handle the
     // Back button. This calls finish() on this activity and pops
the back stack.
     super.onBackPressed()
     //binding.pager.currentItem =
binding.pager.adapter!!.getItemCount() - 1
     } else {
     // Otherwise, select the previous step.
     binding.pager.currentItem = binding.pager.currentItem - 1
        }
    }
}
```

l'activité (avec binding activé)

```
class MainActivity : AppCompatActivity() {
.....

/* pour le menu */
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return super.onOptionsItemSelected(item)
}

override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    return super.onCreateOptionsMenu(menu)
}
```

Implementer PagerAdapter

PagerAdapter est la classe responsable de gestion de fragments pour le compte ViewPager2. Sur la page 10 de transparents vous avez le code suivant à l'intérieur de onCreate() de l'activité :

```
/* création de pagerAdapter qui fournira le contenu à ViewPager2 */
   val pagerAdapter = ScreenSlidePagerAdapter(
        this,
        mutableListOf<Fragment>(pizzaFragment, pastaFragment, storeFragment)
    )
   /* attacher pagerAdapter à ViewPager2 */
   binding.pager.adapter = pagerAdapter
```

La classe ScreenSlidePagerAdapter est définie de façon suivante :

```
class ScreenSlidePagerAdapter(fa : FragmentActivity, var
fragmentList : MutableList<Fragment>)
    : FragmentStateAdapter(  fa ){
    override fun getItemCount(): Int = fragmentList.size
    override fun createFragment(position: Int): Fragment =
fragmentList[position]
}
```

Donc il fallait réécrire deux méthodes, getItemCount() qui retourne le nombre de fragments et createFragment(position : Int) qui est sensée de retourner le fragment à la position indiquée (comme d'habitude les positions commencent à 0).

Implementer les transformations

Les transformations permettent d'implémenter des effets visuels pendant le changement d'onglet dans TabLayout.

Je donne ci-dessous deux exemples de transformations. Pour installer une de ces transformation il faut dé-commenter une de deux lignes à la fin de onCreate() de notre activité.

les transformations (facultatif)

```
private const val MIN_SCALE = 0.85f
private const val MIN_ALPHA = 0.5f

class ZoomOutPageTransformer : ViewPager2.PageTransformer {

    override fun transformPage(view: View, position: Float) {
        view.apply {
            val pageWidth = width
            val pageHeight = height
            when {
                position < -1 -> { // [-Infinity,-1)
                    // This page is way off-screen to the left.
                    alpha = 0f
                }
                position <= 1 -> { // [-1,1]
                    // Modify the default slide transition to shrink the page as well
                    val scaleFactor = Math.max(MIN_SCALE, 1 - Math.abs(position))
                    val vertMargin = pageHeight * (1 - scaleFactor) / 2
                    val horzMargin = pageWidth * (1 - scaleFactor) / 2
                    translationX = if (position < 0) {
                        horzMargin - vertMargin / 2
                    } else {
                        horzMargin + vertMargin / 2
                    }

                    // Scale the page down (between MIN_SCALE and 1)
                    scaleX = scaleFactor
                    scaleY = scaleFactor

                    // Fade the page relative to its size.
                    alpha = (MIN_ALPHA +
                            (((scaleFactor - MIN_SCALE) / (1 - MIN_SCALE)) * (1 - MIN_ALPHA)))
                }
                else -> { // (1,+Infinity]
                    // This page is way off-screen to the right.
                    alpha = 0f
                }
            }}}
```

transformations (facultatif)

```
class DepthPageTransformer : ViewPager2.PageTransformer {
    companion object{
        private const val MIN_SCALE = 0.75f
    }
    override fun transformPage(view: View, position: Float) {
        view.apply {
            val pageWidth = width
            when {
                position < -1 -> { // [-Infinity,-1)
                    // This page is way off-screen to the left.
                    alpha = 0f
                }
                position <= 0 -> { // [-1,0]
                    // Use the default slide transition when moving to the left page
                    alpha = 1f
                    translationX = 0f
                    scaleX = 1f
                    scaleY = 1f
                }
                position <= 1 -> { // (0,1]
                    // Fade the page out.
                    alpha = 1 - position

                    // Counteract the default slide transition
                    translationX = pageWidth * -position

                    // Scale the page down (between MIN_SCALE and 1)
                    val scaleFactor = (MIN_SCALE + (1 - MIN_SCALE) * (1 - Math.abs(position)))
                    scaleX = scaleFactor
                    scaleY = scaleFactor
                }
                else -> { // (1,+Infinity]
                    // This page is way off-screen to the right.
                    alpha = 0f
                }
            }
        }
    }
}
```

layout de fragment

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/title"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:textSize="20sp"
    />


<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recycler"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
    app:layoutManager="LinearLayoutManager"
     />

</androidx.appcompat.widget.LinearLayoutCompat>
```

un fragment

```
class PastaItemFragment(val adapter : RecyclerView.Adapter<PastaAdapter.VH>, layout : Int ) : Fragment(layout) {

    private lateinit var title : String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        arguments?.let {
            title = it.getString(TITLE) ?: ""
        }
    }
    private var binding : FragmentLayoutBinding? = null

    override fun onDestroyView() {
        super.onDestroyView()
        binding = null
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding = FragmentLayoutBinding.bind(view)  /* créer le binding on lui passant le view de la racine*/
        binding!!.title.text = title
        binding!!.recycler.adapter = adapter
        adapter.notifyDataSetChanged()
    }

    companion object {
        private const val TITLE = "title"
        @JvmStatic
        fun newInstance(title: String, adapter : RecyclerView.Adapter<PastaAdapter.VH>,
                    l    ayout : Int) =
            PastaItemFragment(adapter, layout).apply {
                arguments = Bundle().apply {
                    putString(TITLE, title)
                }
            }
    }
}
```

RecyclderView.Adapter

```
private const val TAG="PizzaAdapter"

class PizzaAdapter( val values: MutableList<PizzaItem> )
    : RecyclerView.Adapter<PizzaAdapter.VH>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): VH {
        val holder =
VH( RecyclerItemBinding.inflate( LayoutInflater.from(parent.context), parent,
            false))
        return holder
    }

    override fun onBindViewHolder(holder: VH, position: Int) {
        Log.d(TAG, "position=$position")
        val item = values[position]
        Log.d(TAG, "pizza=$item")
        holder.binding.content.text  = item.content
        holder.pizza = item
    }

    override fun getItemCount(): Int = values.size

    class VH(val binding: RecyclerItemBinding) :
RecyclerView.ViewHolder(binding.root) {
        lateinit var pizza: PizzaItem

    }
}
```