



Reskilling 4Employment Software Developer

Acesso móvel a sistemas de informação

Bruno Santos

bruno.santos.mcv@msft.cesae.pt

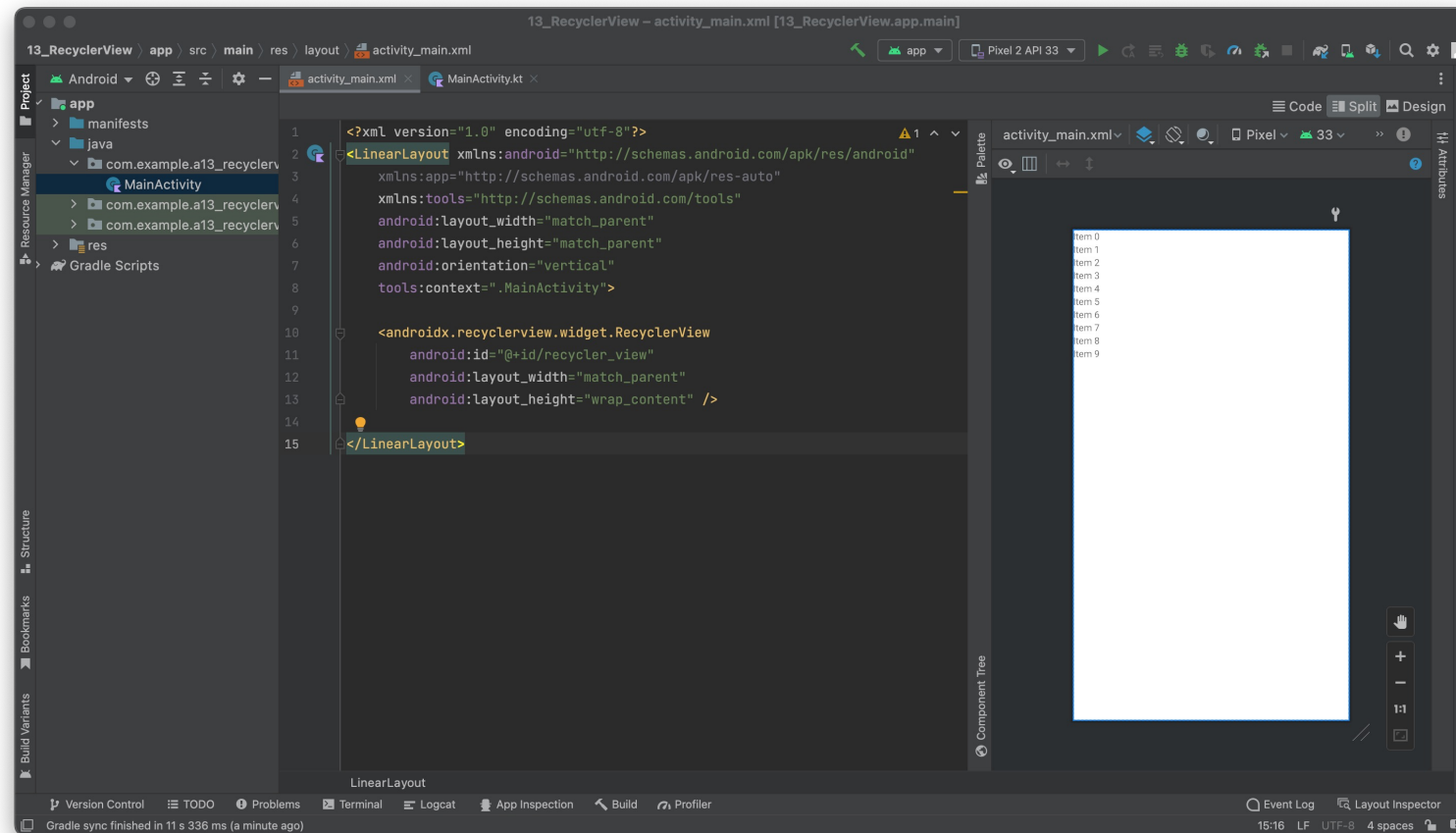
Tópicos

- RecyclerView

RecyclerView

- A RecyclerView é o padrão criado pela Google para substituição e aumento de performance comparativamente com a ListView
- Vamos abrir o layout da Activity principal, remover os elementos dentro do Layout e acrescentar a RecyclerView.
- Ao escrever <RecyclerView devemos seleccionar a opção sugerida
- Finalmente vamos interligar a RecyclerView com a Activity.

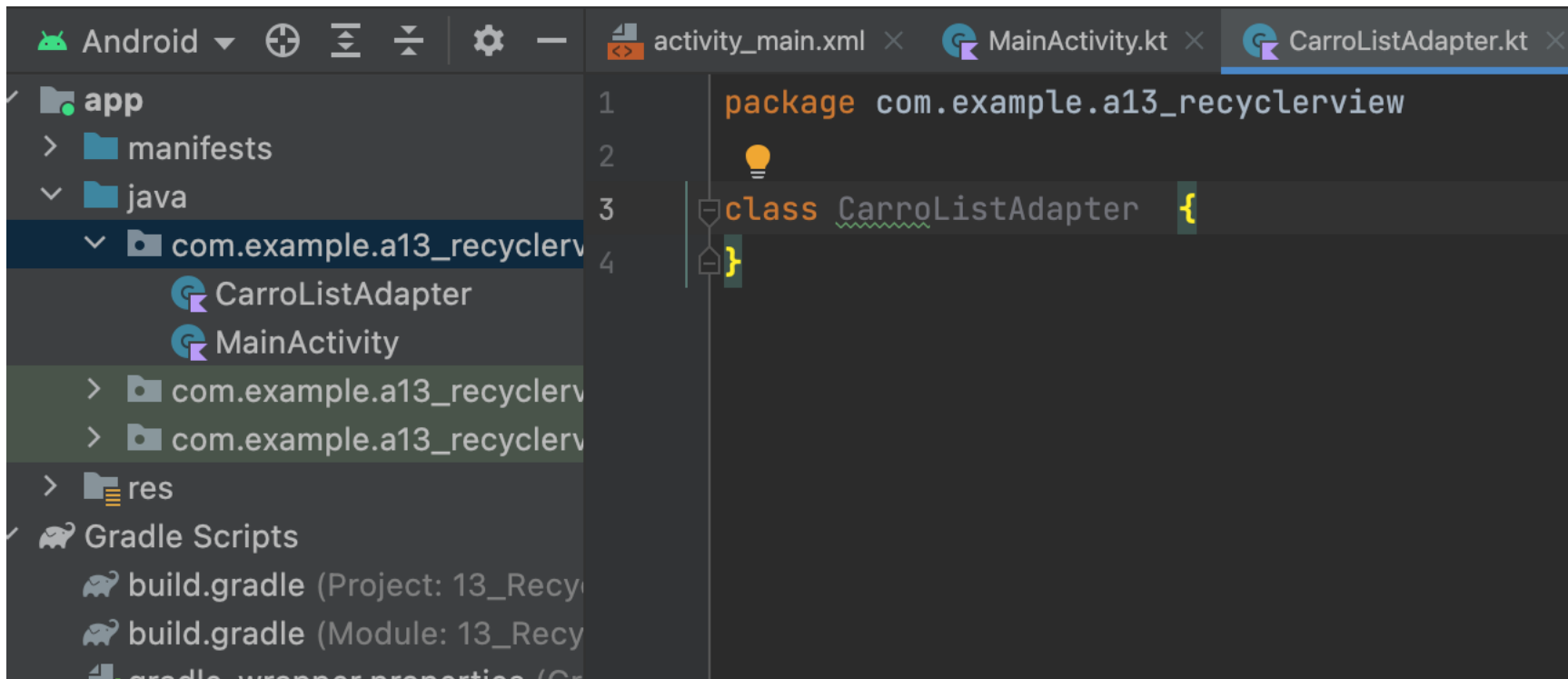
RecyclerView



RecyclerView

- De seguida temos de realizar 2 operações (a ordem é indiferente)
 - Definir um Adapter
 - Definir um Layout
- Na ListView utilizávamos o ArrayAdapter para aplicar um ArrayList numa ListView, neste caso vamos fazer exatamente o mesmo indicando como é feito esse mapeamento.
- Para começar vamos criar uma classe Kotlin dentro do package onde temos os ficheiros Kotlin das nossas Activity, com o nome CarroListAdapter

RecyclerView



The screenshot shows the Android Studio IDE with the following components:

- Toolbar:** Android icon, dropdown menu, add, remove, zoom in, zoom out, settings, and close buttons.
- Project Explorer (Left):**
 - app
 - manifests
 - java
 - com.example.a13_recyclerv
 - CarroListAdapter
 - MainActivity
 - com.example.a13_recyclerv
 - com.example.a13_recyclerv
 - res
 - Gradle Scripts
 - build.gradle (Project: 13_Recy)
 - build.gradle (Module: 13_Recy)
 - gradle-wrapper.properties (Gr)
- Editor (Right):**
 - activity_main.xml
 - MainActivity.kt
 - CarroListAdapter.kt (active)

The code in CarroListAdapter.kt is as follows:

```
1 package com.example.a13_recyclervview
2
3 class CarroListAdapter {
4 }
```

RecyclerView

- Dentro desta vamos criar a classe CarroViewHolder.

```
1      package com.example.a13_recyclerview
2
3      class CarroListAdapter {
4
5          class CarroViewHolder {
6
7          }
8      }
```

RecyclerView

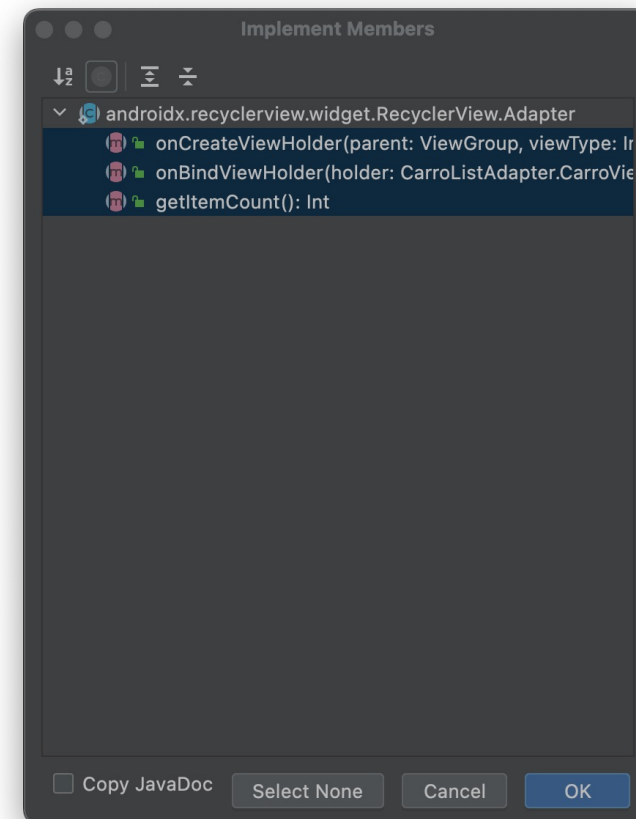
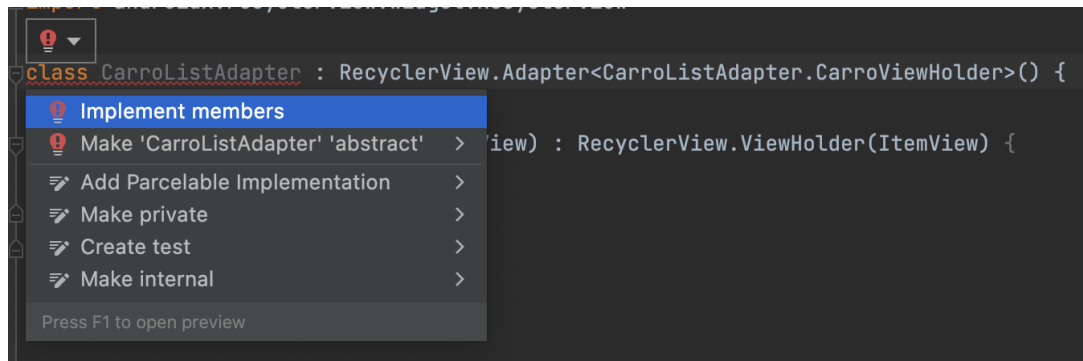
- A classe CarroViewHolder irá estender de RecyclerView.ViewHolder
- A classe CarroListAdapter irá estender de RecyclerView.Adapter

```
class CarroListAdapter : RecyclerView.Adapter<CarroListAdapter.CarroViewHolder>() {  
  
    class CarroViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
  
    }  
}
```

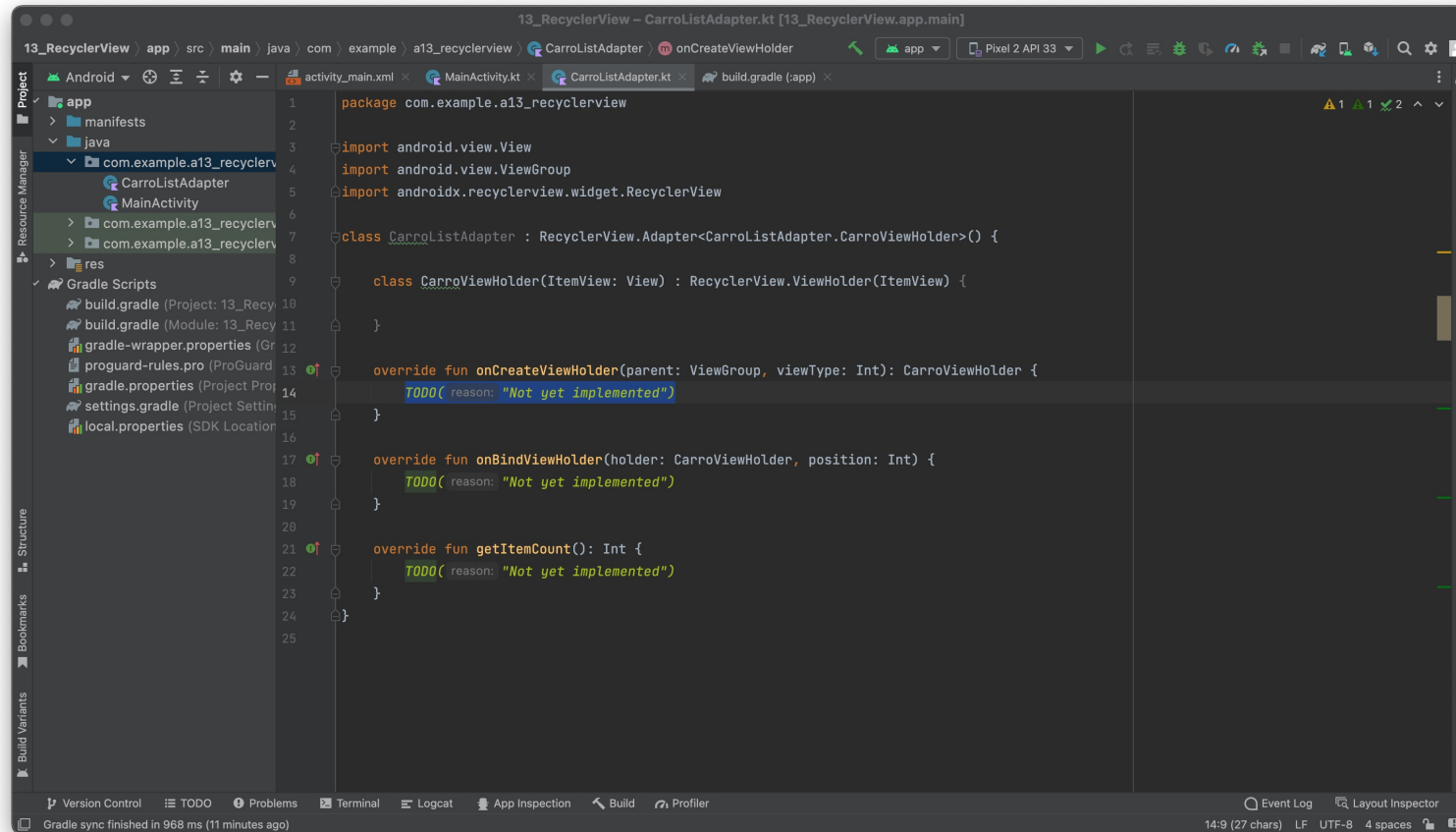

RecyclerView

- De seguida implementamos os métodos necessários no Adapter clicando na lâmpada e selecionar “Implement Methods”

RecyclerView



RecyclerView

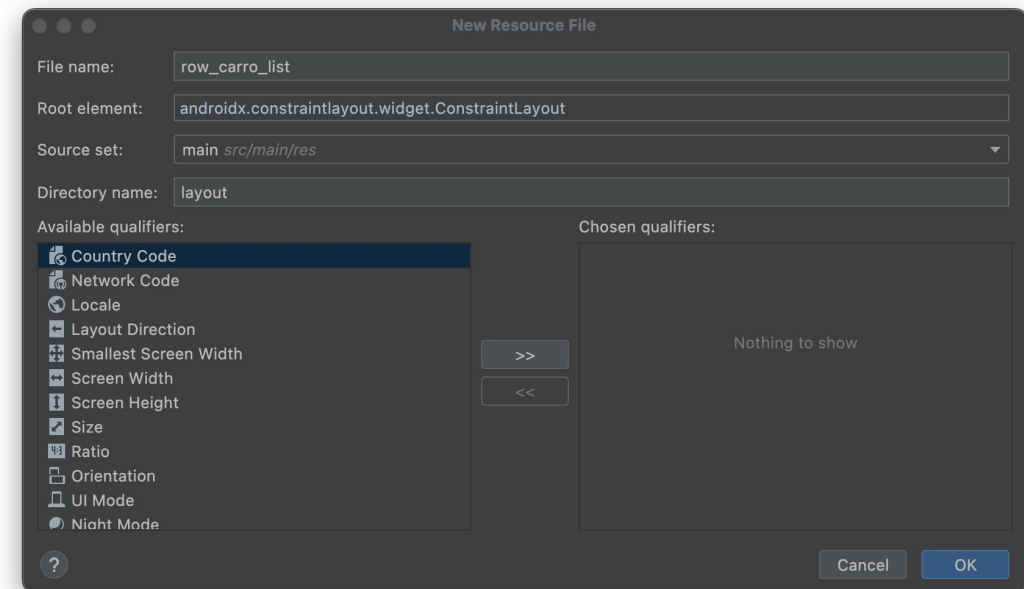
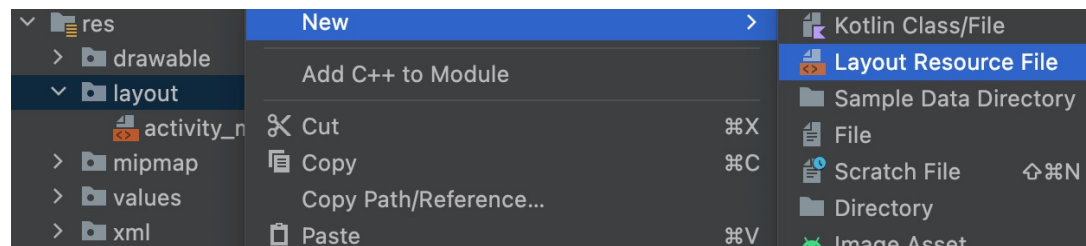


```
1 package com.example.a13_recyclerview
2
3 import android.view.View
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6
7 class CarroListAdapter : RecyclerView.Adapter<CarroListAdapter.CarroViewHolder>() {
8
9     class CarroViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
10
11     }
12
13     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CarroViewHolder {
14         TODO( reason: "Not yet implemented")
15     }
16
17     override fun onBindViewHolder(holder: CarroViewHolder, position: Int) {
18         TODO( reason: "Not yet implemented")
19     }
20
21     override fun getItemCount(): Int {
22         TODO( reason: "Not yet implemented")
23     }
24 }
25
```

RecyclerView

- Neste passo vamos necessitar de ter os layout da interface criada, assim devemos criar um Layout resource file dentro da pasta res>layout

RecyclerView



RecyclerView

- No layout vamos:
 - Trocar o layout para LinearLayout;
 - O parâmetro layout_height trocar para wrap_content;
 - Colocar uma TextView para apresentar o modelo do carro a apresentar

RecyclerView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/text_modelo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

RecyclerView

- Voltando à ViewHolder vamos definir o elemento de layout utilizando o comando `findViewById`

```
class CarroViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
    val textView: TextView = itemView.findViewById(R.id.text_modelo)  
}
```


RecyclerView

- Voltando agora ao Adapter vamos implementar os vários métodos.
- Começando pelo método onCreateViewHolder, que será semelhante ao método onCreate das Activity que temos vindo a trabalhar.

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

RecyclerView

- `parent.getContext();`
 - Permite-nos ir buscar o contexto em que estamos a trabalhar
- `inflater.inflate(R.layout.row_carro_list, parent, false);`
 - Ao fazer inflate indicamos o elemento que vamos acrescentar, neste caso o layout que criamos, depois o parent que é o ViewGroup passado por parâmetro e finalmente o último parâmetro é passado false se o layout que adicionamos já tem um elemento pai (LinearLayout por exemplo) dentro ou true se não tem e queremos criar.
- `return CarroViewHolder(view);`
 - Retorna um CarroViewHolder que é o pedido na assinatura do método

RecyclerView

- Os dois outros métodos serão implementados mais tarde, antes vamos voltar à MainActivity e definir o adapter e o layout já criados

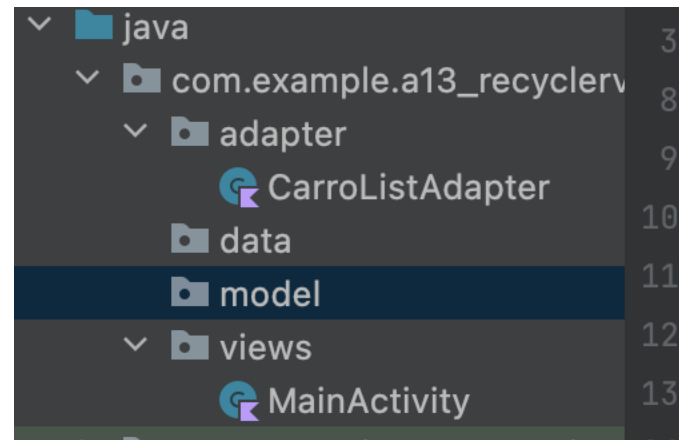
```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    binding = ActivityMainBinding.inflate(layoutInflater)  
    setContentView(binding.root)  
  
    binding.recyclerView.layoutManager = LinearLayoutManager(context: this)  
    binding.recyclerView.adapter = CarroListAdapter()  
}
```

- Uma RecyclerView necessita sempre do LayoutManager, neste caso como estamos a trabalhar com LinearLayout vamos instanciar um LinearLayoutManager e passar esse Layout para a RecyclerView

RecyclerView

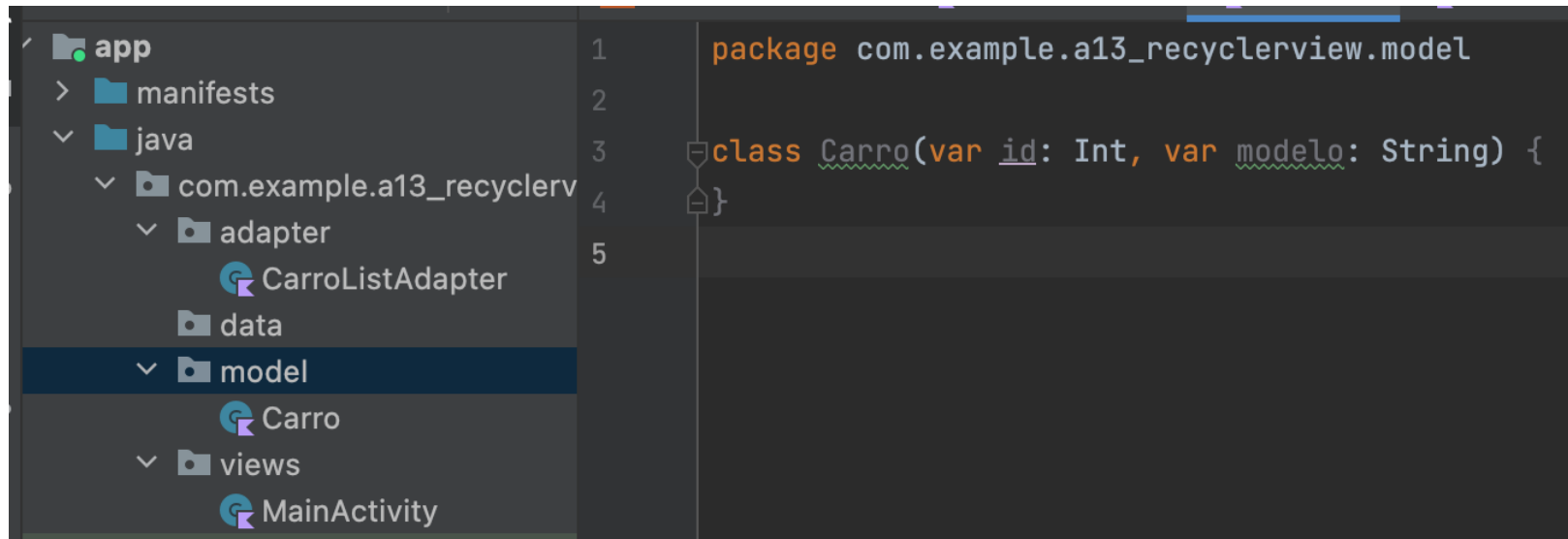
- Neste momento a RecyclerView está construída e funcional sendo que falta popular a mesma com elementos.
- Antes de iniciar este passo vamos organizar os nossos ficheiros em pastas, para isso em cima do package onde temos as classes vamos criar um novo package chamado Adapter e colocar lá dentro o ficheiro CarroListAdapter.
- De seguida repetir o processo para o ficheiro MainActivity no package views.
- Vamos ainda criar dois outros package: Model e Data

RecyclerView



RecyclerView

- Vamos ainda criar a classe Carro dentro do Model



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure is as follows:

- app
 - manifests
 - java
 - com.example.a13_recyclerv
 - adapter
 - CarroListAdapter
 - data
 - model
 - Carro
 - views
 - MainActivity

The code editor shows the following code:

```
1 package com.example.a13_recyclerv.model
2
3 class Carro(var id: Int, var modelo: String) {
4 }
5
```

RecyclerView

- E a classe CarroMock dentro do Data. Esta classe servirá para criar uma lista de exemplo

```
class CarroMock() {  
    var listaCarros = ArrayList<Carro>()  
  
    init {  
        for (i in 0 ≤ .. ≤ 100) {  
            listaCarros.add(Carro(i, i.toString()))  
        }  
    }  
}
```

RecyclerView

- Voltando ao CarListAdapter vamos criar um construtor para associar a lista de carros à RecyclerView.
- Notar que vamos ter um erro na classe MainActivity após o próximo passo, algo que iremos resolver depois

RecyclerView

```
class CarroListAdapter(val listaCarro: ArrayList<Carro>) :  
    RecyclerView.Adapter<CarroListAdapter.CarroViewHolder>() {  
  
    class CarroViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
        val textView: TextView = itemView.findViewById(R.id.text_modelo)  
    }  
}
```

RecyclerView

- Aproveitamos também para preencher o método getItemCount. Como já temos a lista basta retornar o size da mesma

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

RecyclerView

- O método `onBindViewHolder` é invocado sempre que uma linha é inserida no layout. Vamos preencher o método com o seguinte:

```
override fun onBindViewHolder(holder: CarroViewHolder, position: Int) {  
    val carro = listaCarro[position]  
  
    holder.textView.setText(carro.modelo)  
}
```

RecyclerView

- Para corrigir o erro da MainActivity vamos alterar na CarListAdapter para incluir o Mock

```
binding.recyclerView.layoutManager = LinearLayoutManager(context: this)

val mock = CarroMock()
binding.recyclerView.adapter = CarroListAdapter(mock.listaCarros)
```

RecyclerView



RecyclerView

- A RecyclerView foi criada e está totalmente funcional. Para percebermos o porquê de este padrão ser muito mais performático do que uma ListView vamos acrescentar duas variáveis na classe CarroListAdapter para contar quantas vezes é executado o método onCreateViewHolder e onBindViewHolder.
- Após essa criação vamos iniciar a aplicação e verificar os valores.

RecyclerView

```
var contadorOnCreate = 0
var contadorOnBind = 0

class CarroViewHolder(ItemView: View) : RecyclerView.ViewHolder(ItemView) {
    val textView: TextView = itemView.findViewById(R.id.text_modelo)
}

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CarroViewHolder {
    contadorOnCreate++
    val view = LayoutInflater.from(parent.context)
        .inflate(R.layout.row_carro_list, parent, attachToRoot: false)

    return CarroViewHolder(view)
}

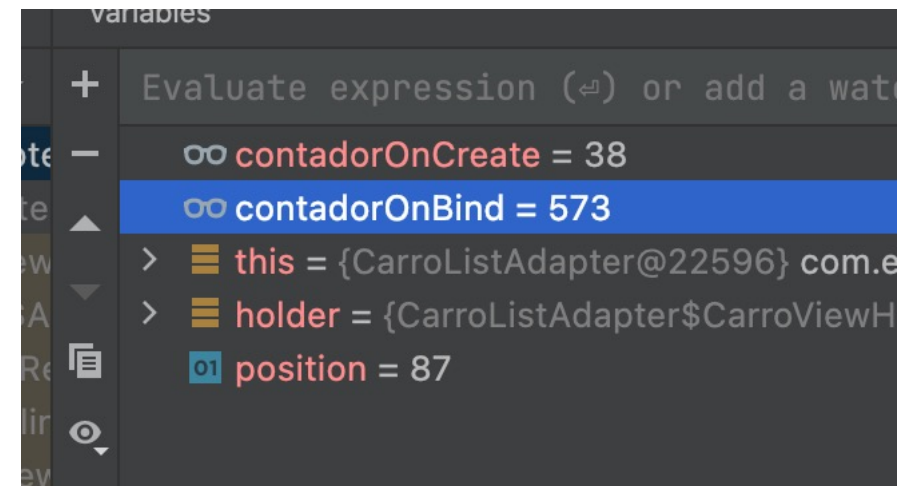
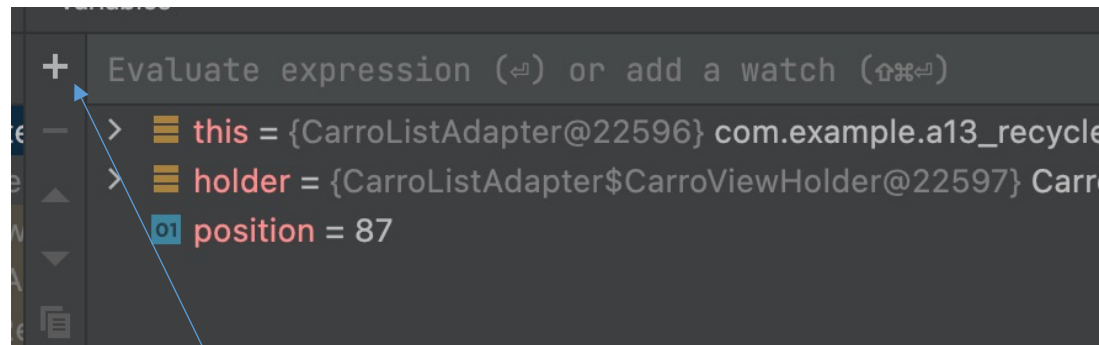
override fun onBindViewHolder(holder: CarroViewHolder, position: Int) {
    contadorOnBind++
    val carro = listaCarro[position]

    holder.textView.setText(carro.modelo)
}
```

RecyclerView

- Após executar a aplicação em modo Debug e fazendo scroll algumas vezes para cima e para baixo colocamos um breakpoint na linha `contadorOnBind++`;
- Aqui vamos analisar os valores das variáveis contador

RecyclerView



RecyclerView

- Conseguimos verificar a diferença entre o número de execuções concluindo que a própria estrutura recicla as linhas já criadas colocando novos elementos e esse é o principal aumento de performance da RecyclerView

RecyclerView

- Para permitir realizar um clique num elemento da RecyclerView temos de, dentro do Adapter:
 1. Adicionar o evento OnClickListener no construtor do Adapter
 2. Criar e implementar a classe OnClickListener com a função onClick
 3. Associar o evento no método onBindViewHolder

RecyclerView

```
class CarroListAdapter(val listaCarro: ArrayList<Carro>, val onClickListener: OnClickListener) :  
    RecyclerView.Adapter<CarroListAdapter.CarroViewHolder>() {  
  
    var contadorOnCreate = 0  
    var contadorOnBind = 0  
  
    class CarroViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
        val textView: TextView = itemView.findViewById(R.id.text_modelo)  
    }  
  
    class OnClickListener(val clickListener: (carro: Carro) -> Unit) {  
        fun onClick(carro: Carro) = clickListener(carro)  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CarroViewHolder {  
        contadorOnCreate++  
        val view = LayoutInflater.from(parent.context)  
            .inflate(R.layout.row_carro_list, parent, attachToRoot: false)  
  
        return CarroViewHolder(view)  
    }  
  
    override fun onBindViewHolder(holder: CarroViewHolder, position: Int) {  
        contadorOnBind++  
        val carro = listaCarro[position]  
  
        holder.textView.setText(carro.modelo)  
        holder.itemView.setOnClickListener { it: View!  
            onClickListener.onClick(carro)  
        }  
    }  
}
```

RecyclerView

```
binding.recyclerView.layoutManager = LinearLayoutManager( context: this)

val mock = CarroMock()
binding.recyclerView.adapter =
    CarroListAdapter(mock.listaCarros, CarroListAdapter.OnClickListener { carro ->
        Toast.makeText( context: this, carro.modelo, Toast.LENGTH_SHORT).show()
    })
```

RecyclerView

