UDB VIRTUAL

Universidad Don Bosco, El Salvador

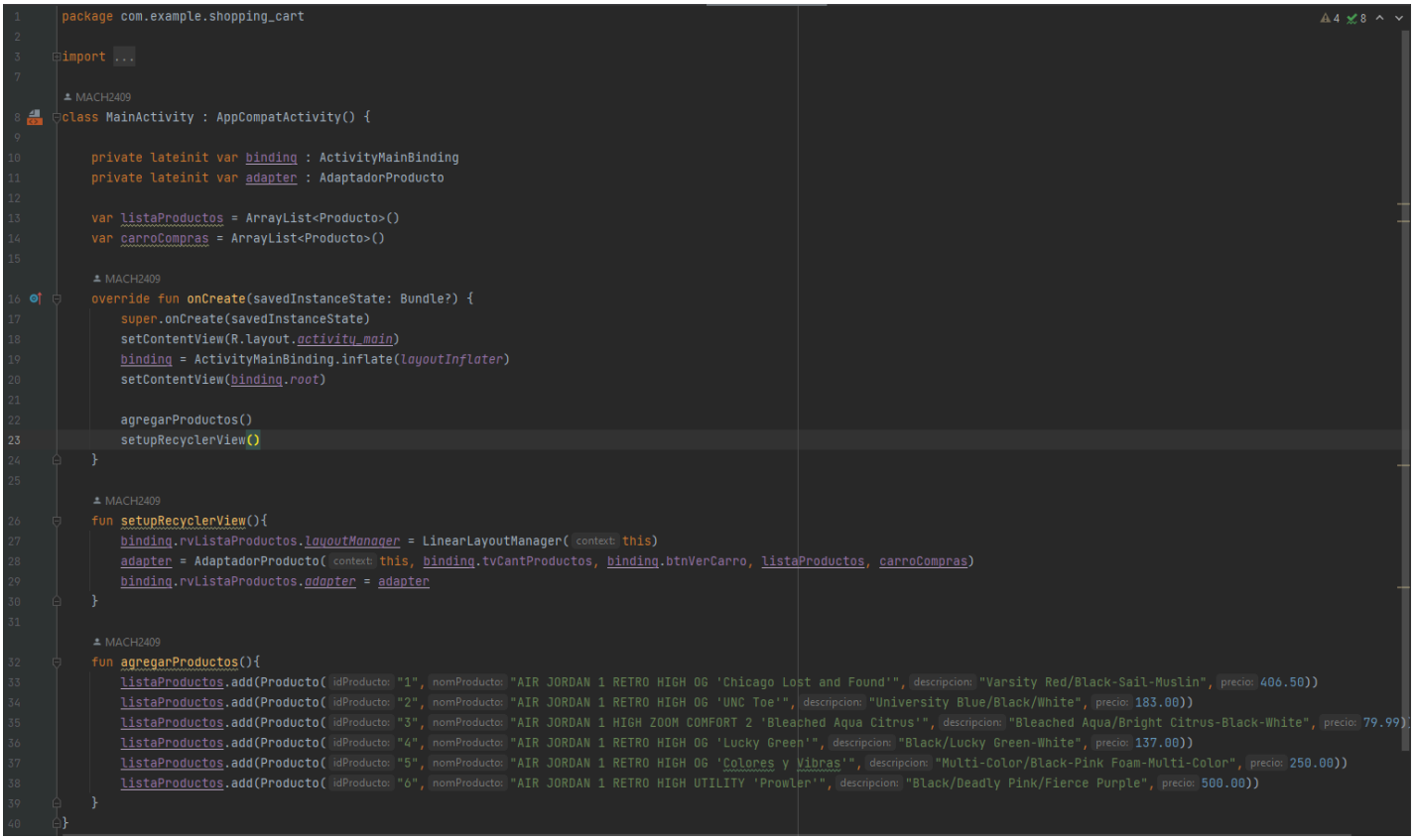
DESARROLLO DE SOFTWARE PARA APLICACIONES

Trabajo de Investigación

Miguel Antonio Campos Hernández CH212519

Jennyfer Liseth Soto Chacón SC151874

MainActivity.kt



```kotlin
package com.example.shopping_cart

import ...

MACH2409
class MainActivity : AppCompatActivity() {

    private lateinit var binding : ActivityMainBinding
    private lateinit var adapter : AdaptadorProducto

    var listaProductos = ArrayList<Producto>()
    var carroCompras = ArrayList<Producto>()

    MACH2409
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        agregarProductos()
        setupRecyclerView()
    }

    MACH2409
    fun setupRecyclerView(){
        binding.rvListaProductos.layoutManager = LinearLayoutManager( context: this)
        adapter = AdaptadorProducto( context: this, binding.tvCantProductos, binding.btnVerCarro, listaProductos, carroCompras)
        binding.rvListaProductos.adapter = adapter
    }

    MACH2409
    fun agregarProductos(){
        listaProductos.add(Producto( idProducto: "1", nomProducto: "AIR JORDAN 1 RETRO HIGH OG 'Chicago Lost and Found'", descripcion: "Varsity Red/Black-Sail-Muslin", precio: 406.50))
        listaProductos.add(Producto( idProducto: "2", nomProducto: "AIR JORDAN 1 RETRO HIGH OG 'UNC Toe'", descripcion: "University Blue/Black/White", precio: 183.00))
        listaProductos.add(Producto( idProducto: "3", nomProducto: "AIR JORDAN 1 HIGH ZOOM COMFORT 2 'Bleached Aqua Citrus'", descripcion: "Bleached Aqua/Bright Citrus-Black-White", precio: 79.99))
        listaProductos.add(Producto( idProducto: "4", nomProducto: "AIR JORDAN 1 RETRO HIGH OG 'Lucky Green'", descripcion: "Black/Lucky Green-White", precio: 137.00))
        listaProductos.add(Producto( idProducto: "5", nomProducto: "AIR JORDAN 1 RETRO HIGH OG 'Colores y Vibras'", descripcion: "Multi-Color/Black-Pink Foam-Multi-Color", precio: 250.00))
        listaProductos.add(Producto( idProducto: "6", nomProducto: "AIR JORDAN 1 RETRO HIGH UTILITY 'Prowler'", descripcion: "Black/Deadly Pink/Fierce Purple", precio: 500.00))
    }
}
```

Método onCreate

Se realiza la configuración inicial en el método onCreate.
Se infla el diseño (activity_main.xml) usando View Binding.
Se llama a agregarProductos() para inicializar la lista de productos.
Se llama a setupRecyclerView() para configurar el RecyclerView.

Método setupRecyclerView

Configura el RecyclerView con un LinearLayoutManager.
Crea una instancia del adaptador personalizado (AdaptadorProducto) y lo asigna al RecyclerView.

Método agregarProductos

Agrega productos a la lista listaProductos. Cada producto es una instancia de la clase Producto.

Producto.kt

```kotlin
package com.example.shopping_cart

import java.io.Serializable

MACH2409
data class Producto(
    var idProducto: String,
    var nomProducto: String,
    var descripcion: String,
    var precio: Double
): Serializable
```

idProducto: Un identificador único para el producto (presumiblemente un código o número único).
nomProducto: El nombre del producto.
descripcion: Una descripción del producto.
precio: El precio del producto (representado como un valor de tipo Double).
Serializable: Implementa la interfaz Serializable, permitiendo que las instancias de esta clase se serialicen y deserialicen.

AdaptadorProducto.kt

```kotlin
package com.example.shopping_cart

import ...

MACH2409
class AdaptadorProducto(
    var context: Context,
    var tvCantProductos:TextView,
    var btnVerCarro:Button,
    var listaProductos: ArrayList<Producto>,
    var carroCompras: ArrayList<Producto>
): RecyclerView.Adapter<AdaptadorProducto.ViewHolder>() {

    MACH2409
    class ViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {
        val tvNomProducto = itemView.findViewById(R.id.tvNomProducto) as TextView
        val tvDescripcion = itemView.findViewById(R.id.tvDescripcion) as TextView
        val cbCarro = itemView.findViewById(R.id.cbCarro) as CheckBox
        val tvPrecio = itemView.findViewById(R.id.tvPrecio) as TextView

    }

    MACH2409
    override fun onCreateViewHolder(
        parent: ViewGroup,
        viewType: Int
    ): ViewHolder {
        var vista = LayoutInflater.from(parent.context).inflate(R.layout.item_rv_productos,parent, attachToRoot: false)
        return ViewHolder(vista)
    }
```

Clase AdaptadorProducto
La clase AdaptadorProducto extiende RecyclerView.Adapter y utiliza un ViewHolder personalizado.

Clase interna ViewHolder
Se define una clase interna ViewHolder que extiende RecyclerView.ViewHolder. Esta clase almacena referencias a las vistas individuales dentro del elemento de la lista.

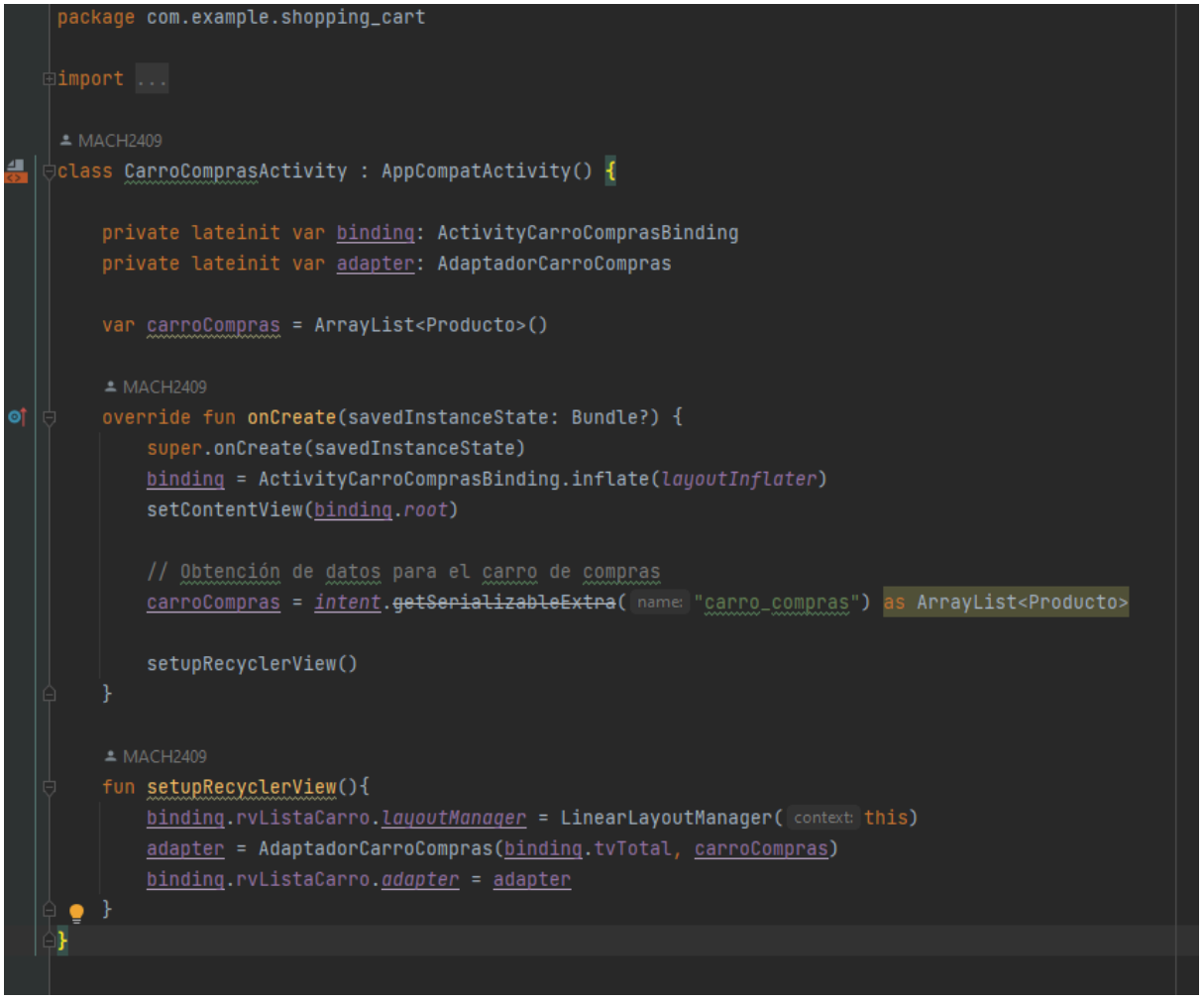
Método onCreateViewHolder
Este método se llama cuando el RecyclerView necesita un nuevo ViewHolder. Infla la vista del elemento de la lista (item_rv_productos) y devuelve una instancia de ViewHolder que contiene las referencias a las vistas.

```
MACH2409
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val producto = listaProductos[position]

    holder.tvNomProducto.text = producto.nomProducto
    holder.tvDescripcion.text = producto.descripcion
    holder.tvPrecio.text = producto.precio.toString()

    holder.cbCarro.setOnCheckedChangeListener { compoundButton, b ->
        if (holder.cbCarro.isChecked){
            tvCantProductos.text = "${Integer.parseInt(tvCantProductos.text.toString().trim()) + 1}"
            carroCompras.add(listaProductos[position]) // Agregar el producto al carro de compras
        } else {
            tvCantProductos.text = "${Integer.parseInt(tvCantProductos.text.toString().trim()) - 1}"
            carroCompras.remove(listaProductos[position]) // Quitar el producto del carro de compras
        }
    }

    // Acción al presionar el botón del Carrito de Compras
    btnVerCarro.setOnClickListener{ it: View!
        val intent = Intent(context, CarroComprasActivity::class.java)
        intent.putExtra( name: "carro_compras", carroCompras)
        context.startActivity(intent)
    }
}

// Regresamos el tamaño de la lista de Productos
MACH2409
override fun getItemCount(): Int {
    return listaProductos.size

}
}
```

Método onBindViewHolder
Este método se llama para vincular datos a las vistas en una posición específica. Aquí, se establece el contenido de las vistas y se definen acciones para el cambio en el estado del CheckBox y para el clic en el botón del carrito de compras.

Método getItemCount
Devuelve el tamaño de la lista de productos, indicando cuántos elementos debe gestionar el RecyclerView.

CarroComprasActivity.kt

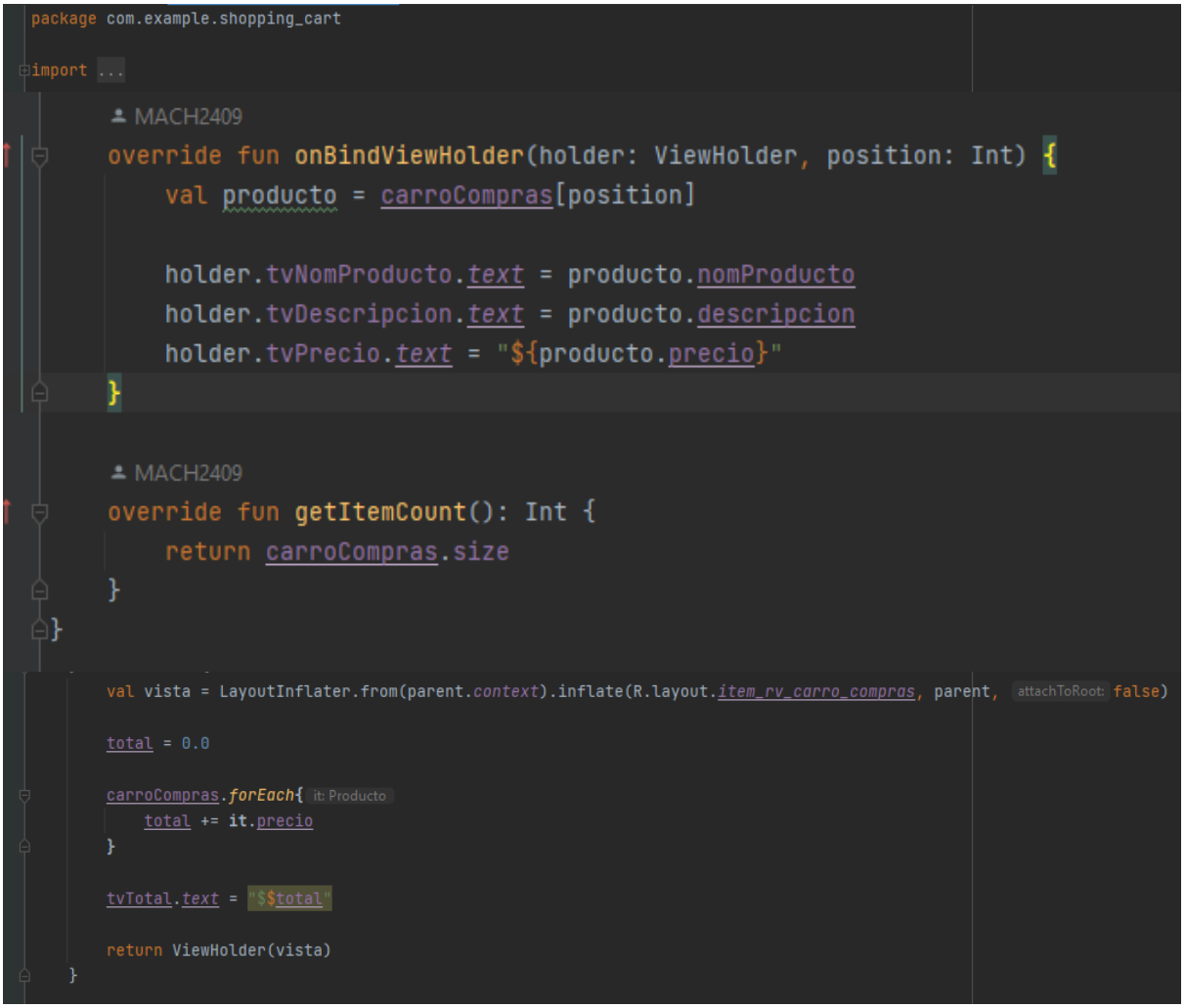


```
package com.example.shopping_cart

import ...

MACH2409
class CarroComprasActivity : AppCompatActivity() {

    private lateinit var binding: ActivityCarroComprasBinding
    private lateinit var adapter: AdaptadorCarroCompras

    var carroCompras = ArrayList<Producto>()

    MACH2409
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityCarroComprasBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Obtención de datos para el carro de compras
        carroCompras = intent.getSerializableExtra( name: "carro_compras") as ArrayList<Producto>

        setupRecyclerView()
    }

    MACH2409
    fun setupRecyclerView(){
        binding.rvListaCarro.layoutManager = LinearLayoutManager( context: this)
        adapter = AdaptadorCarroCompras(binding.tvTotal, carroCompras)
        binding.rvListaCarro.adapter = adapter
    }
}
```

Método onCreate
En el método onCreate, se realiza la configuración inicial de la actividad.
Se infla el diseño (activity_carro_compras.xml) usando View Binding.
Se obtienen los datos del carrito de compras pasados desde la actividad anterior (MainActivity) mediante intent.getSerializableExtra().
Se llama a setupRecyclerView() para configurar el RecyclerView que mostrará los productos en el carrito.

Método setupRecyclerView
Configura el RecyclerView con un LinearLayoutManager.
Crea una instancia del adaptador personalizado (AdaptadorCarroCompras) y lo asigna al RecyclerView.
Se pasa una referencia al TextView binding.tvTotal que mostrará el total de la compra al adaptador.

AdaptadorCarroCompras.kt



```
package com.example.shopping_cart

import ...

    MACH2409
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val producto = carroCompras[position]

        holder.tvNomProducto.text = producto.nomProducto
        holder.tvDescripcion.text = producto.descripcion
        holder.tvPrecio.text = "${producto.precio}"
    }

    MACH2409
    override fun getItemCount(): Int {
        return carroCompras.size
    }
}

    val vista = LayoutInflater.from(parent.context).inflate(R.layout.item_rv_carro_compras, parent, attachToRoot: false)

    total = 0.0

    carroCompras.forEach{ it: Producto
        total += it.precio
    }

    tvTotal.text = "$$total"

    return ViewHolder(vista)
}
```

Clase AdaptadorCarroCompras
La clase AdaptadorCarroCompras extiende RecyclerView.Adapter y utiliza un ViewHolder personalizado.

Propiedad Total
Una propiedad que se utilizará para calcular y almacenar el total de la compra.

Clase interna ViewHolder
Se define una clase interna ViewHolder que extiende RecyclerView.ViewHolder. Esta clase almacena referencias a las vistas individuales dentro del elemento de la lista.

Método onCreateViewHolder
Este método se llama cuando el RecyclerView necesita un nuevo ViewHolder. Infla la vista del elemento de la lista (item_rv_carro_compras) y realiza cálculos del total de la compra para mostrarlo en el TextView tvTotal.

```
    MACH2409
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val producto = carroCompras[position]

        holder.tvNomProducto.text = producto.nomProducto
        holder.tvDescripcion.text = producto.descripcion
        holder.tvPrecio.text = "${producto.precio}"
    }

    MACH2409
    override fun getItemCount(): Int {
        return carroCompras.size
    }
}
```

Método onBindViewHolder

Este método se llama para vincular datos a las vistas en una posición específica. Aquí, se establece el contenido de las vistas utilizando los datos del producto en la posición dada.

Método getItemCount

Devuelve el tamaño de la lista de productos en el carrito de compras, indicando cuántos elementos debe gestionar el RecyclerView.

FacturaActivity.kt

```
class FacturaActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_factura)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }

        val carroCompras = intent.getSerializableExtra("carro_compras") as ArrayList<Producto>

        val rvListaProductosFactura = findViewById<RecyclerView>(R.id.rvListaProductosFactura)
        rvListaProductosFactura.layoutManager = LinearLayoutManager(this)
        rvListaProductosFactura.adapter = AdaptadorFactura(carroCompras)

        calcularYMostrarTotal(carroCompras)
    }

    private fun calcularYMostrarTotal(productos: List<Producto>) {
        val total = productos.sumOf { it.precio }

        val tvSubTotalFactura = findViewById<TextView>(R.id.tvSubtotal)
        tvSubTotalFactura.text = "Total: $${total}"


        val tvTotalFactura = findViewById<TextView>(R.id.tvTotalFactura)
        tvTotalFactura.text = "Total: $${total}"



    }
}
```

Clase FacturaActivity
Se declara una clase llamada FacturaActivity que extiende AppCompatActivity. Esto sugiere que es una actividad en Android.

Método onCreate
Se sobrescribe el método onCreate que se llama cuando la actividad se inicia. enableEdgeToEdge(): parece ser una función personalizada que habilita el diseño sin bordes, aprovechando toda la pantalla.

Diseño de Actividades
Establece el diseño de la actividad a partir del archivo XML llamado activity_factura.

Obtención de Datos de la Actividad Anterior
Recupera el objeto serializable "carro_compras" enviado desde la actividad anterior.

Configuración del RecyclerView
Configura un RecyclerView para mostrar la lista de productos utilizando un adaptador personalizado llamado AdaptadorFactura.

Función para Calcular y Mostrar el Total
Calcula el total de los productos y actualiza dos TextViews en el diseño con los valores calculados.

AdaptadorFactura.kt

```kotlin
package com.example.shopping_cart

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class AdaptadorFactura(
    private val productos: List<Producto>
) : RecyclerView.Adapter<AdaptadorFactura.ProductoViewHolder>() {

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

    override fun onBindViewHolder(holder: ProductoViewHolder, position: Int) {
        val producto = productos[position]
        holder.bind(producto)
    }

    override fun getItemCount(): Int = productos.size

    class ProductoViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        private val tvNombreProducto: TextView = itemView.findViewById(R.id.tvNombreProducto)
        private val tvDescripcionProducto: TextView = itemView.findViewById(R.id.tvDescripcionProducto)
        private val tvPrecioProducto: TextView = itemView.findViewById(R.id.tvPrecioProducto)

        fun bind(producto: Producto) {
            tvNombreProducto.text = producto.nomProducto
            tvDescripcionProducto.text = producto.descripcion
            tvPrecioProducto.text = "Precio: ${producto.precio}"
        }
    }
}
```

Clase AdaptadorFactura
Declara una clase llamada AdaptadorFactura que extiende RecyclerView.Adapter y utiliza ProductoViewHolder como ViewHolder.

Método onCreateViewHolder
Crea y devuelve una instancia de ProductoViewHolder inflando el diseño de un elemento de producto (item_producto_factura) utilizando un LayoutInflater.

Método onBindViewHolder
Vincula los datos de un producto específico en la posición dada con el ProductoViewHolder correspondiente.

Método getItemCount
Devuelve el número total de elementos en la lista de productos, que es la longitud de la lista.

Clase Interna ProductoViewHolder
Define una clase interna ProductoViewHolder que extiende RecyclerView.ViewHolder.
Esta clase representa la vista individual de un elemento en el RecyclerView.
Almacena referencias a las vistas (TextViews) dentro de la vista del elemento.
La función bind actualiza estas vistas con la información del producto correspondiente.