Daniel Zanon Lopez

2º DAM PMDM

Practica 6p
(Si hi ha cap dubte per a corregir, a la carpeta on es troba aquest document es troba el projecte).

1. Es descriu correctament l'estructura de fitxers i carpetes del projecte.

   Perspectiva de Fitxers:

   app: Carpeta principal de l'aplicació.
   src: Conté el codi font i recursos de l'aplicació.
   java (o kotlin): Codificació font en Java o Kotlin.
   res: Recursos com arxius XML de disseny, imatges, etc.
   manifests: Conté AndroidManifest.xml amb la configuració de l'aplicació.
   Gradle Scripts: Carpeta amb arxius de configuració de Gradle per gestionar dependencies i configuracions del projecte.
   .idea: Carpeta amb arxius de configuració per a Android Studio.
   build: Carpeta que conté arxius generats pel sistema de construcció Gradle (no editar manualment).

   Perspectiva d'Android:

   En la perspectiva d'Android, Android Studio ofereix diverses vistes per explorar components i recursos de l'aplicació:
   Vista de disseny: Per dissenyar la interfície d'usuari utilitzant l'Editor de Disseny.
   Explorador de projecte: Mostra l'estructura jeràrquica del projecte i els recursos.
   Logcat: Mostra registres i missatges de depuració.

Gradle: Per gestionar dependencies i executar tasques de construcció.
Component Tree: Per explorar i editar l'arbre de components de l'aplicació, com activitats i fragments.

2. S'explica correctament quines implicacions té crear una nova activitat.

   Interfície d'Usuari: Cada activitat té la seva pròpia interfície d'usuari, definida en un arxiu XML, permetent una organització clara de les pantalles de l'aplicació.
   Cicle de Vida: Les activitats passen per estats com crear, pausar, reprendre i destruir, que has de gestionar adequadament per mantenir el comportament de l'aplicació coherent.
   Transicions: Les activitats es llancen i tanquen mitjançant intents, permetent la navegació entre diferents pantalles de l'aplicació.
   Pila d'Activitats: Les activitats es gestionen en una pila, la darrera activitat oberta es troba a dalt i tornar enrere elimina les anteriors.
   Pas de Dades: Pots passar dades entre activitats mitjançant intents i bundles.
   Tema i Estil: Cada activitat pot tenir el seu propi estil i tema.
   Interacció de l'Usuari: Les activitats gestionen interaccions com clics de botons i introducció de text.
   Gestió d'Estats: Has de manejar la conservació d'estat en canvis d'orientació i altres situacions.
   Recursos Específics: Les activitats poden tenir recursos específics com arxius XML de disseny i assets.

3. Es creen els mètodes per capturar les diferents fases del cicle de vida.

```kotlin
override fun onStart() {
    super.onStart()
    Log.d( tag: "DEPURACION",  msg: "Al metodo onStart")
}

override fun onResume() {
    super.onResume()
    Log.d( tag: "DEPURACION",  msg: "Al metodo onResume")
}

override fun onPause() {
    super.onPause()
    Log.d( tag: "DEPURACION", msg: "Al metodo onPause")
}
override fun onStop() {
    super.onStop()
    Log.d( tag: "DEPURACION",  msg: "Al metodo onStop")
}
override fun onRestart() {
    super.onRestart()
    Log.d( tag: "DEPURACION",  msg: "Al metodo onRestart")
}
override fun onDestroy() {
    super.onDestroy()
    Log.d( tag: "DEPURACION",  msg: "Al metodo onDestroy")
}
```

4. S'analitzen les fases per les que passa l'activitat rere un canvi de configuració i es justifica la pèrdua d'estat.
5. Es manté l'estat fent ús de Bundle.

```kotlin
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    // Code to save the state
    Log.d( tag: "DEPURACION",  msg: "Estoy en onSave")
    outState.putInt("Comptador", comptador)
}

override fun onRestoreInstanceState(savedInstanceState: Bundle) {
    super.onRestoreInstanceState(savedInstanceState)
    // Code to restore the state
    Log.d( tag: "DEPURACION",  msg: "Estoy en onRestore")
    comptador = savedInstanceState.getInt( key: "Comptador")
    binding.textViewComptador.text = comptador.toString()
}
```

6. S'afig la funcionalitat per restar i ressetejar el comptador.

```
<Button
    android:id="@+id/btResta"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    android:textSize="34sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/btReset"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textViewComptador" />

<Button
    android:id="@+id/btReset"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#FF5722"
    android:backgroundTint="#FF5722"
    android:text="RESET"
    android:textSize="34sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/btAdd"
    app:layout_constraintStart_toEndOf="@+id/btResta"
    app:layout_constraintTop_toBottomOf="@+id/textViewComptador"
    app:rippleColor="#FF5722"
    app:strokeColor="#FF5722" />
```

```
// Referencia al botón
val btAdd=binding.btAdd
val btResta=binding.btResta
val btReset=binding.btReset



// Asociaciamos una expresióin lambda como
// respuesta (callback) al evento Clic sobre
// el botón
binding.btAdd.setOnClickListener { it: View!
    comptador++
    binding.textViewComptador.text = comptador.toString()
}
// Boton para restar
binding.btResta.setOnClickListener { it: View!
    comptador--
    binding.textViewComptador.text = comptador.toString()
}
// Boton para resetear
binding.btReset.setOnClickListener { it: View!
    comptador = 0
    binding.textViewComptador.text = comptador.toString()
}
```

7. S'implementa correctament el View Binding.

```kotlin
import com.ieseljust.pmdm.comptador.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    var comptador=0

    lateinit var binding:ActivityMainBinding

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

        binding = ActivityMainBinding.inflate(layoutInflater)

        val view = binding.root
        setContentView(view)

        //setContentView(R.layout.activity_main)
        setContentView(binding.root)

        // Referencia al TextView
        // val textViewContador=findViewById<TextView>(R.id.textViewComptador)

        // Inicialitzem el TextView amb el comptador a 0
        //textViewContador.setText(comptador.toString())

        binding.textViewComptador.setText(comptador.toString())

        // Referencia al botón
        val btAdd=binding.btAdd
        val btResta=binding.btResta
        val btReset=binding.btReset
```

```
// Asociaciamos una expresióin lambda como
// respuesta (callback) al evento Clic sobre
// el botón
binding.btAdd.setOnClickListener { it: View!
    comptador++
    binding.textViewComptador.text = comptador.toString()
}
// Boton para restar
binding.btResta.setOnClickListener { it: View!
    comptador--
    binding.textViewComptador.text = comptador.toString()
}
// Boton para resetear
binding.btReset.setOnClickListener { it: View!
    comptador = 0
    binding.textViewComptador.text = comptador.toString()
}
```