Primera práctica de kotlin en Android Studio.
PMDM 24/25
Santiago Rodenas Herráiz

Descripción de la práctica

Queremos simular un crud de clientes de dos modos diferentes. En una primera versión, utilizando una interfaz y como segunda versión, utilizando **lambdas**. En nuestro crud, la idea es utilizar un repositorio de clientes de sólo lectura y una vez inicializada la pantalla, utilizaremos una lista mutable para insertar todos los clientes del repositorio inicial y de ahí, **insertar**, **actualizar** y **eliminar**. Lo haremos **simulando** un listado, ya que de momento, acabamos de empezar y no sabemos cómo hacerlo. Cada vez que pulsemos el botón de insertar, llamaremos a un objeto cuya clase simulará un DialogFragment (Diálogo). Dentro del diálogo, deberemos de simular la captura de datos, para posteriormente insertarla o modificarla en nuestra lista.

Tener en cuenta, que para que podamos simular los listener, al diálogo hay que pasarle la referencia de nuestra clase principal, que es la que simula la pantalla y para ello tenemos las interfaces. Hago una pregunta, ¿Qué sucede si tenemos diferentes pantallas y quisiéramos reutilizar el mismo diálogo?, o incluso diferentes formas de acceso a los datos. Al pasar la referencia de la pantalla al diálogo, queremos utilizar una instancia que implemente de la interfaz, con los métodos del crud. Dichos métodos los implementaremos en la pantalla principal, pero utilizaremos un controller para que sea él, quien lleve a cabo la inserción, edición y eliminación.

Os aconsejo que utilices git para un control de versiones y así recordar los comandos.

En el ordenador de clase:

- Te creas el repositorio de tu cuenta de git.
- **git clone https://github.com/srodenas/comparaci-n_lambda_inter.git** (Clonamos el repositorio. Este es el mío, tendrá que ser el vuestro)
- **(opcional) git remote add origin <url del repositorio>** (configuramos repo remoto, pero esto lo podemos obviar por el git clone de antes)
- **git add .** (añadimos todos los ficheros)
- **git commit -m " bla bla bla"** (comiteamos)
- **git push origin main** (subimos nuestra rama al remoto)
- **git checkout -b lambda** (creamos la nueva rama y nos cambiamos)
- **git branch** (listamos las ramas y verificamos donde estamos)
- **git commit -m "bla bla bla"** (comiteamos cambios en la rama lambda)
- **git push origin lambda** (subimos los cambios a la rama remota)

En el ordenador de tu casa:

- **git clone <repositorio>** (Clonamos el repositorio)-> trae la main
- **git pull origin main** (buena práctica después de clonar. Aseguramos)
- **git fetch origin** (traemos información de los commit a local)
- **git checkout -b lambda origin/lambda** (creamos rama y conectamos con la remota)
- **git pull origin lambda** (traemos los commit de la remota a la local)

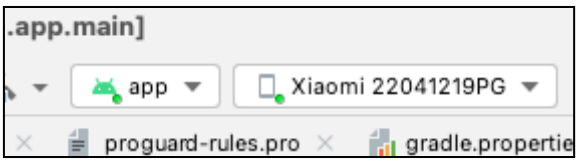
1.- Rama main:

Los métodos de la interfaz serán:

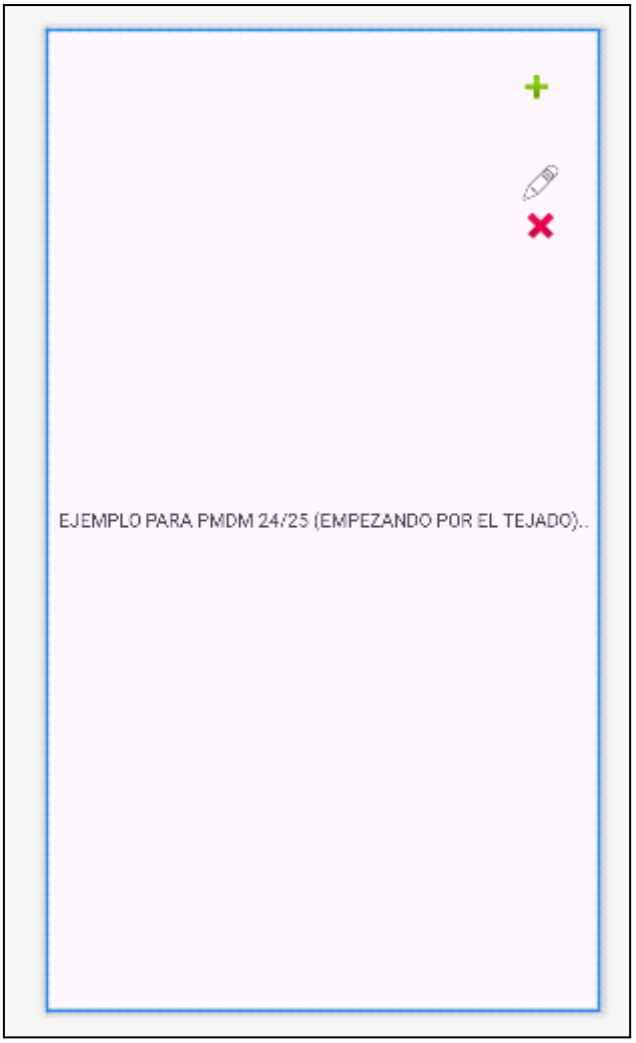
```
fun ClientAdd ( id: Int, name: String )
fun ClientDel ( id: Int)
fun ClientUpdate ( id: Int, name: String)
```

Como consejo:

1. Utilizar ramas en git (con interfaz y con lambda)
2. Crear tres paquetes principales (views, logic, data)
3. Dentro de views, crear la pantalla y el diálogo. El diálogo, simulará un dialogFragment, por tanto poner un método setListener que acepte como referencia la pantalla de tipo Interfaz.
4. Crear un método show dentro del diálogo que emule al show real.
5. Utilizar un repositorio con un listOf dentro de data.
6. Utilizar un MutableList para los datos dinámicos
7. Utilizar un paquete dentro de logic llamado interface, con dicha interfaz.
8. Implementar la clase principal de la interfaz y sobreescribir los métodos.
9. Implementar tres botones dentro de la pantalla principal.
10. Capturar el evento de los tres botones y llamar a un método del dialogo que emule la captura de datos.
11. Para cada uno de los casos insertar, editar o eliminar, aplicar alguna lógica para que de manera aleatoria, en el caso de editar o borrar, seleccionemos alguno y sea modificado con cualquier cambio o eliminado. Para la inserción, podemos insertar el mismo cliente pero con diferente id.
12. Utilizar alguna lógica, para crear un id diferente a cada uno de los clientes.
13. Utilizar Log, para mostrar por pantalla los datos y las interacciones con el usuario.
14. Utilizar un terminal físico.
15. Ejecutar y realizar trazas de ejecución.



Dispositivo físico



Ejemplo de pantalla.

```
Esto es un ejemplo de log
El cliente con id = 104, ha sido insertado correctamente
[Client(id=100, name=Santi), Client(id=101, name=Sonia), Client(id=102, name=Guille), Client(id=103, nam
El cliente con id = 105, ha sido insertado correctamente
[Client(id=100, name=Santi), Client(id=101, name=Sonia), Client(id=102, name=Guille), Client(id=103, nam
El cliente con id = 100, ha sido actualizado correctamente
[Client(id=100, name=CAMBIADO), Client(id=101, name=Sonia), Client(id=102, name=Guille), Client(id=103,
El cliente con id = 105, ha sido eliminado correctamente
[Client(id=100, name=CAMBIADO), Client(id=101, name=Sonia), Client(id=102, name=Guille), Client(id=103,
El cliente con id = 103, ha sido eliminado correctamente
[Client(id=100, name=CAMBIADO), Client(id=101, name=Sonia), Client(id=102, name=Guille), Client(id=104,
El cliente con id = 106, ha sido insertado correctamente
[Client(id=100, name=CAMBIADO), Client(id=101, name=Sonia), Client(id=102, name=Guille), Client(id=104,
```

Ejemplo de traza de ejecución.

2.- Rama lambda:
En una nueva rama, aplicar los cambios necesarios para sustituir la interfaz por lambda.

Como Consejo:

1. Eliminar la interfaz y todo lo que dependa de ella.
2. Pasar al Diálogo, tres lambda correspondientes a los tres métodos que deberá ejecutarse dentro del diálogo.
3. Volver a ejecutar y realizar trazas de ejecución.

“Como Tarea de clase, tendréis que simular el proyecto completo y subir un pdf con las capturas necesarias de la simulación de cada una de las ramas”.