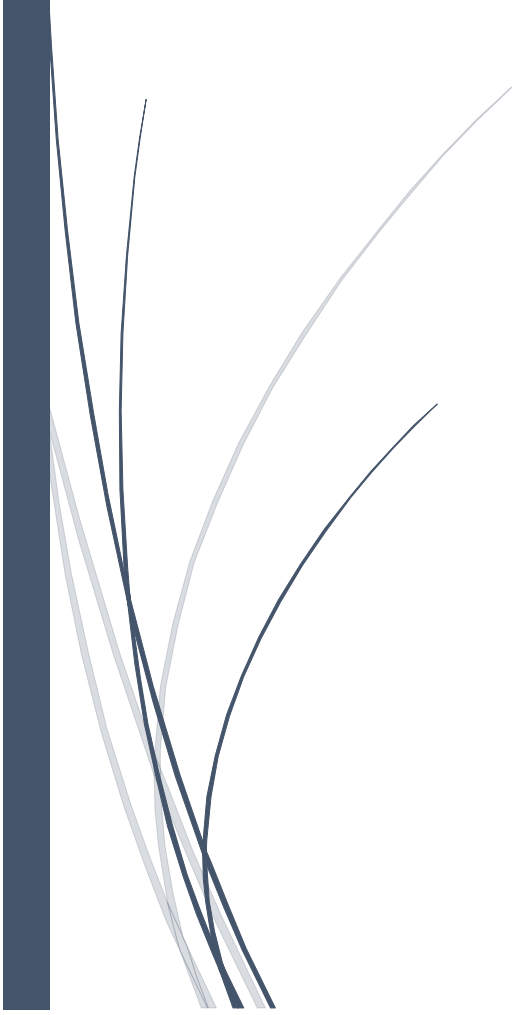
4-11-2023

Práctica 7: Factory Method

Acorán González Moray



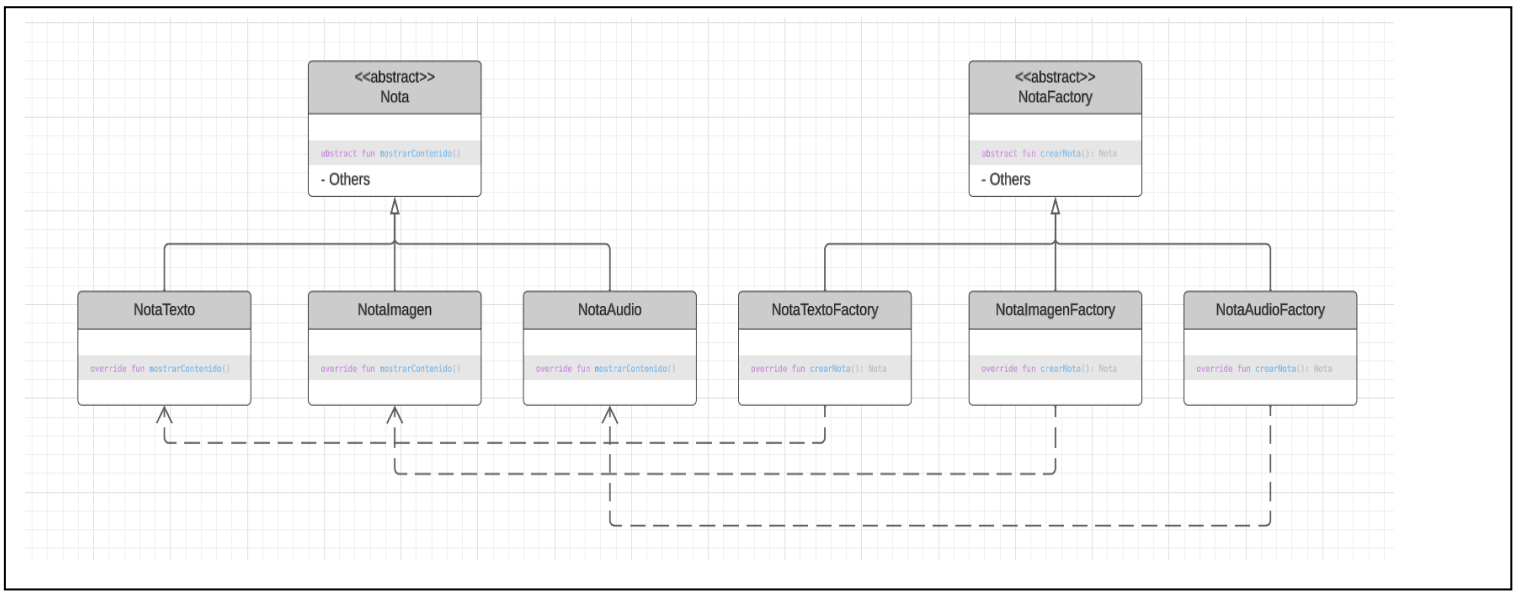
Universidad de Las Palmas de Gran Canaria

PROGRAMACIÓN DE APLICACIONES MÓVILES NATIVAS

Contenido

1.- Diseñar una estructura de clases para una aplicación de notas en Kotlin utilizando el patrón Factory Method. Las notas pueden ser de tipo texto, imagen y audio.

- Diagrama de clases con la estructura del patrón de diseño aplicada a la aplicación de notas.



- Código de las clases sin entrar en detalles de implementación, en la presentación de clases puedes seguir el nivel de detalle del ejemplo del validador de documentos

```kotlin
abstract class Nota {
    abstract fun mostrarContenido()
}
```

```kotlin
class NotaTexto: Nota() {
    override fun mostrarContenido() {
        println("Desplega Texto")
    }
}
```

```kotlin
class NotaImagen: Nota() {
    override fun mostrarContenido() {
        println("Desplega Imagen")
    }
}
```

```kotlin
class NotaAudio: Nota() {
    override fun mostrarContenido () {
        println("Desplega Audio")
    }
}
```

```
abstract class NotaFactory {
    abstract fun crearNota(): Nota
}
```

```
class NotaTextoFactory: NotaFactory() {
    override fun crearNota(): Nota {
        return NotaTexto()
    }
}
```

```
class NotaImagenFactory: NotaFactory() {
    override fun crearNota(): Nota {
        return NotaImagen()
    }
}
```

```
class NotaAudioFactory: NotaFactory() {
    override fun crearNota(): Nota {
        return NotaAudio()
    }
}
```

Posible Codigo para su uso:

```
fun main() {
    val notaFactory: NotaFactory = NotaTextoFactory()
    val nota: Nota = notaFactory.crearNota()
    nota.mostrarContenido()
}
```

- Breve descripción de cada clase

Nota, esta clase es la clase base abstracta que representa una nota. Contiene métodos y propiedades comunes a todas las notas, como podría ser el título y la fecha de creación y en este caso una funcion para desplegar su contenido

NotaTexto, esta clase hereda de la clase Nota y representa una nota de tipo texto

NotaImagen, esta clase hereda de la clase Nota y representa una nota de tipo imagen

NotaAudio, esta clase hereda de la clase Nota y representa una nota de tipo audio

NotaFactory, esta clase es la clase abstracta que actúa como la fábrica para crear diferentes tipos de notas. Contiene el método abstracto "crearNota" que debe ser implementado por las subclases.

NotaTextoFactory, esta clase hereda de la clase NotaFactory y se encarga de crear notas de tipo texto. Implementa el método "crearNota" para crear una instancia de la clase NotaTexto.

NotaImagenFactory, esta clase hereda de la clase NotaFactory y se encarga de crear notas de tipo imagen. Implementa el método "crearNota" para crear una instancia de la clase NotaImagen.

NotaAudioFactory, esta clase hereda de la clase NotaFactory y se encarga de crear notas de tipo audio. Implementa el método "crearNota" para crear una instancia de la clase NotaAudio.

2.- - Reflexiona sobre cambios futuros:

- ¿Qué pasa si en un futuro se quisiera añadir un nuevo tipo de notas?

Vamos a suponer que se quiere añadir una Nota de Video, entonces habria que crear una clase que implente a la clase **Notas**, y una clase concreta que implemente a **NotaFactory**

- ¿Qué partes de tu aplicación tendrías que modificar?

La clase abstracta **NotaFactory** podría ser actualizada/modificada para incluir un nuevo método que cree instancias de la nueva clase de nota si esta lo requiere claro esta.

- ¿Qué nuevas clases tendrías que añadir?

Se requeriría extender/añadir a la jerarquía de clases para incluir la nueva clase concreta y la clase que implemente a **Nota** que represente la nota de video.