UNIVERSIDAD LAS PALMAS DE GRAN CANARIA

PROGRAMACIÓN DE APLICACIONES MÓVILES NATIVAS

INFORME SEMANA 7:

PATRONES DE DISEÑO: FACTORY METHOD

PRESENTADO POR: IVONNE YULIETH OJEDA CUERVO

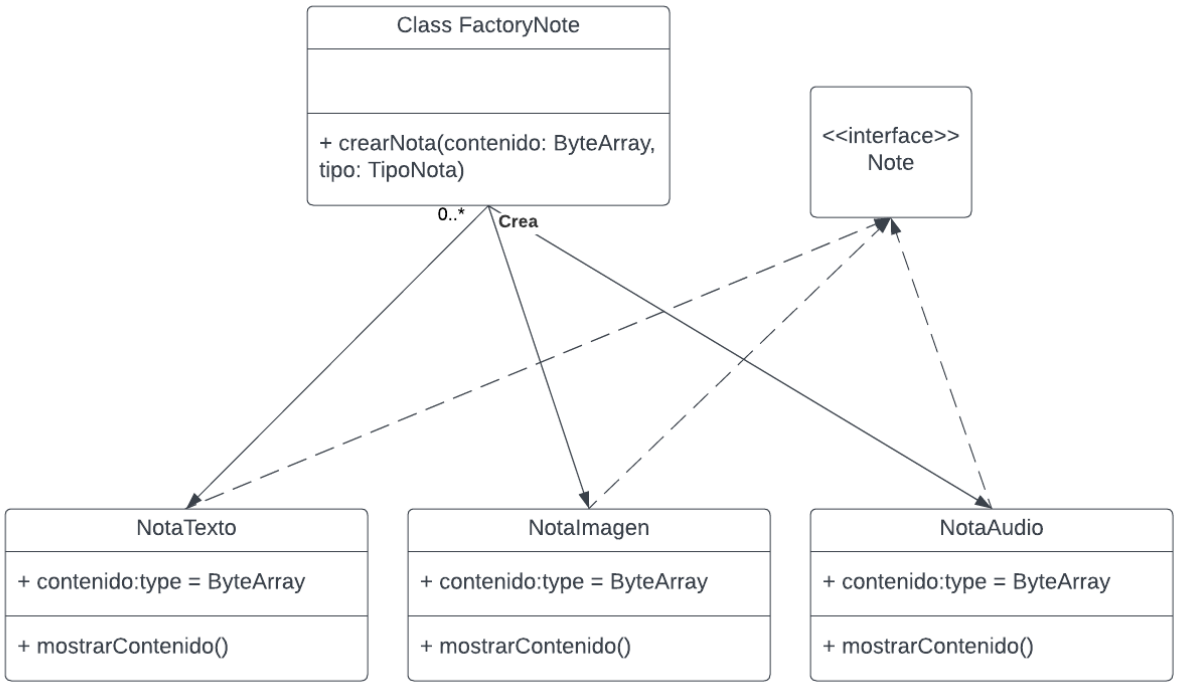
AZ930245

LAS PALMAS DE GRAN CANARIA – 2023

PATRONES DE DISEÑO: FACTORY METHOD

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.

1. Diagrama de Clases



2. Código de Clases con su descripción

```
// La interface que genera el modelo que toda nota debe tener
interface Nota {
fun mostrarContenido()
}

// La clase que maneja notas tipo texto
class NotaTexto(private val contenido: ByteArray) : Nota {
override fun mostrarContenido() {
println("Nota de Texto: ${String(contenido)}")
}
}
```

```
// La clase que maneja notas tipo Imagen
class NotaImagen(private val contenido: ByteArray) : Nota {
override fun mostrarContenido() {
println("Nota sobre la imagen: Mostrando la imagen...")// Código específico para
imagen
}
}
// La clase que maneja notas tipo Audio
class NotaAudio(private val contenido: ByteArray) : Nota {
override fun mostrarContenido() {
println("Nota de audio: Reproduciendo el audio...")// Código específico para Audio
}
}

// Factory class que crea las diferentes notas
class FactoryNote {
fun crearNota(contenido: ByteArray, tipo: TipoNota): Nota {
return when (tipo) {
TipoNota.Texto -> NotaTexto(contenido)
TipoNota.Imagen -> NotaImagen(contenido)
TipoNota.Audio -> NotaAudio(contenido)
}
}
}
// Programa principal
fun main() {
val factory = FactoryNote()

val textContent = "Esta es una nota de texto"
val imageContent = byteArrayOf(0x12, 0x34, 0x56, 0x78) // Acá va la imagen
val audioContent = byteArrayOf(0x7F, 0x80, 0x81, 0x82) // Acá va el audio

val notaTexto = factory.crearNota(textContent.toByteArray(), TipoNota.Texto)
val notaImagen = factory.crearNota(imageContent, TipoNota.Imagen)
val notaAudio = factory.crearNota(audioContent, TipoNota.Audio)

notaTexto.mostrarContenido()
notaImagen.mostrarContenido()
notaAudio.mostrarContenido()
}
```

Reflexiona sobre cambios futuros: ¿Qué pasa si en un futuro se quisiera añadir un nuevo tipo de notas? ¿Qué partes de tu aplicación tendrías que modificar? ¿Qué nuevas clases tendrías que añadir?

Si en el futuro se quisiera añadir un nuevo tipo de notas, como por ejemplo "NotaVideo", tendría que realizar las siguientes modificaciones:

Crear una nueva clase para el nuevo tipo de nota que implemente la interfaz Nota y defina la lógica específica para las notas de video. Esta clase tendría su propia implementación de mostrarContenido.

Actualizar la fábrica (Factory): En la clase FactoryNote, debería modificar el método crearNota para que pueda crear instancias de NotaVideo además de los tipos existentes. La fábrica tendría un nuevo caso en la sentencia when para manejar las solicitudes de creación de notas de video.