Programación de Aplicaciones Móviles Nativas

Patrones de diseño: Factory Method

Raúl Cruz Ortega
Laura González Suárez
Ismael Ramos Alonso

Introducción

El patrón Factory Method ofrece una solución probada para la creación de objetos en un sistema de manera flexible y extensible. Al comprender y aplicar este patrón, los desarrolladores pueden simplificar la creación de objetos, permitiendo la incorporación de nuevos tipos de objetos sin modificar el código existente.

En el contexto de una aplicación de notas, el uso de este patrón de diseño significa que podemos agregar nuevos tipos de notas, como notas de texto, imágenes o audio de forma sencilla sin hacer cambios en todo el código.

Implementación del patrón Factory Method en nuestra aplicación

Con el objetivo de implementar el patrón Factory Method en nuestra aplicación de notas, debemos seguir una serie de pasos clave.

Primero, definiremos una interfaz que establecerá el comportamiento fundamental de todas las notas en nuestra aplicación.

A continuación, crearemos clases concretas para cada tipo de nota, como "TextNote," "ImageNote," y "AudioNote." Estas clases representarán las diferentes variantes de notas disponibles en la aplicación y proporcionarán implementaciones específicas para mostrar su contenido.

Por último, desarrollaremos una clase fábrica que proporcionará un método para la creación de instancias de notas basadas en el tipo de nota que se desee crear. Así, el patrón Factory Method nos permitirá agregar nuevas funcionalidades a nuestra aplicación de notas de manera eficiente y sin afectar el código existente.

Diagrama de clases de la aplicación

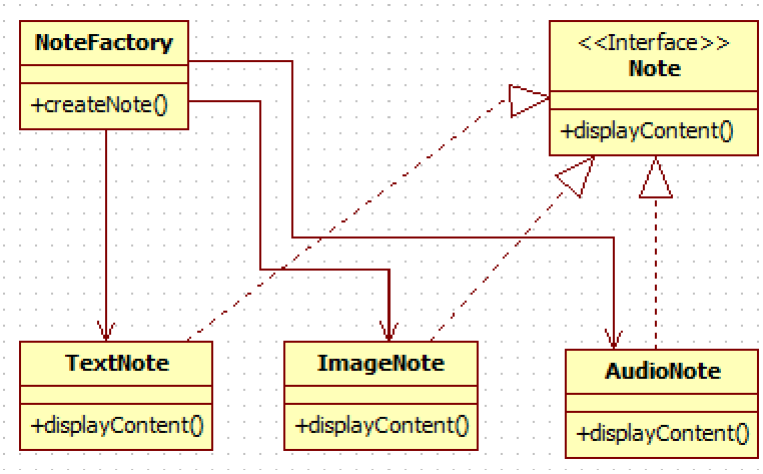

Ilustración 1: Diagrama de clases de la aplicación

Código de la aplicación

A continuación, se muestra el código de las clases sin entrar en detalles de implementación:

```kotlin
// Interfaz Note
interface Note {
    fun displayContent()
}

// ** Clases concretas para cada tipo de nota
// 1. Nota de Texto
class TextNote : Note {
    override fun displayContent() {
        // Implementar la visualización de una nota de texto
    }
}

// 2. Nota de Imagen
class ImageNote : Note {
    override fun displayContent() {
        // Implementar la visualización de una nota de imagen
    }
}

// 3. Nota de Audio
class AudioNote : Note {
    override fun displayContent() {
        // Implementar la visualización de una nota de audio
    }
}
```

```kotlin
// Clase que actúa como fábrica para crear notas
class NoteFactory {
    fun createNote(type: String): Note? {
        return when (type) {
            "text" -> TextNote()
            "image" ->  ImageNote()
            "audio" -> AudioNote()
            else -> null
        }
    }
}
```

Ilustración 2: código de las clases de la aplicación usando Factory Method

Explicación de las clases de la aplicación

Interfaz Note: La interfaz Note define el comportamiento básico que todas las notas deben tener, en este caso, el método displayContent() que permite mostrar el contenido de la nota. Esta interfaz sirve como un contrato para garantizar que todas las notas implementen este método.

Clases concretas para cada tipo de nota:

- **TextNote:** Esta clase concreta implementa la interfaz Note y se encarga de la visualización de una nota de texto. Aquí se definirá cómo mostrar el contenido de una nota de texto.
- **ImageNote:** Similar a TextNote, esta clase también implementa la interfaz Note, pero se encarga de la visualización de una nota de imagen.
- **AudioNote:** Al igual que las otras clases, AudioNote también implementa la interfaz Note, pero se encarga de la visualización de una nota de audio.

Clase NoteFactory: La clase NoteFactory actúa como una fábrica para crear instancias de diferentes tipos de notas. El método createNote(type: String) toma un argumento que representa el tipo de nota ("text", "image", "audio") y devuelve una instancia de la nota correspondiente.

Reflexión sobre cambios futuros

Si en el futuro se quisiera añadir un nuevo tipo de nota, como por ejemplo una "HandwrittenNote," el proceso sería sencillo. Simplemente crearíamos una nueva clase concreta que implemente la interfaz Note. Esta nueva clase se encargaría de definir cómo se crea y muestra una nota escrita a mano.

Sin embargo, es importante señalar que para que esta nueva clase funcione correctamente en la aplicación, tendríamos que asegurarnos de que la clase encargada de la creación de notas sea consciente de esta adición. Debemos ajustar las secciones del código que manejan las notas para incluir la nueva clase "HandwrittenNote."

Una vez que hayamos agregado la nueva clase concreta y ajustado las secciones relevantes del código, la aplicación podrá crear y mostrar notas escritas a mano sin problemas. Esto destaca el poder del patrón Factory Method, ya que nos permite introducir nuevas clases sin necesidad de modificar el código existente en su totalidad. Esto hace que la aplicación sea más modular y escalable, facilitando la incorporación de funcionalidades adicionales sin perturbar la base ya establecida.