Patrones de diseño Factory

Sirius

Protectora animal

Selene González Curbelo
Mariana Surós Álvarez

05/11/2023

ULPGC
Escuela de Ingeniería Informática

Contents

Introducción 2
1. Estructura de clases con patrón Factory Method 2
2. Elementos clave de la aplicación 2
Diagrama de clases 2
Código de las clases 3
Breve descripción de cada clase 4
3. Preparación para el futuro 5
¿Qué pasa si en un futuro se quisiera añadir un nuevo tipo de notas? 5
¿Qué partes de tu aplicación tendrías que modificar? 5
¿Qué nuevas clases tendrías que añadir? 5

Selene González Curbelo
Mariana Surós Álvarez

Introducción

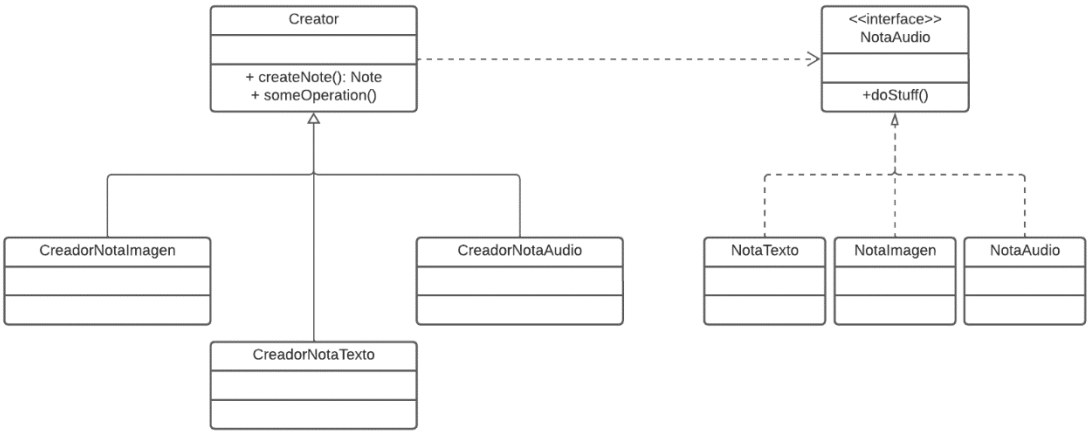
Este informe se centra en la estructura de clases de una aplicación que utiliza el patrón Factory Method en Kotlin. La aplicación está diseñada para la gestión de notas, que pueden ser de diferentes tipos, como texto, imagen y audio.

1. Estructura de clases con patrón Factory Method

- **Clase creadora**
  - Creator
    - createNote: Note
    - someOperation()
- **Producto**
  - <<interface>> Note
    - doStuff()
- **Productos concretos**
  - NotaTexto: Note
  - NotaAudio: Note
  - NotaImagen: Note
- **Creadores concretos**
  - CreadorNotaTexto
  - CreadorNotaImagen
  - CreadorNotaAudio

2. Elementos clave de la aplicación

Diagrama de clases



Selene González Curbelo
Mariana Surós Álvarez

Código de las clases

```
// Producto - Interfaz que define el comportamiento común para todos los tipos de notas
interface Note {
    fun doStuff()
}

// Productos Concretos - Implementaciones específicas de la interfaz del producto
class TextNote : Note {
    override fun doStuff() {
        println("This is a text note.")
    }
}

class ImageNote : Note {
    override fun doStuff() {
        println("This is an image note.")
    }
}

class AudioNote : Note {
    override fun doStuff() {
        println("This is an audio note.")
    }
}

// Clase creadora - Interfaz que contiene el Factory Method para crear notas
abstract class NoteCreator {
    abstract fun createNote(): Note
    fun someOperation() {
        val note = createNote()
        note.doStuff()
    }
}
```


Selene González Curbelo
Mariana Surós Álvarez

```
// Creadores Concretos - Subclases que sobrescriben el Factory Method para devolver un tipo
específico de producto

class TextNoteCreator : NoteCreator() {

    override fun createNote(): Note {

        return TextNote()

    }

}


class ImageNoteCreator : NoteCreator() {

    override fun createNote(): Note {

        return ImageNote()

    }

}


class AudioNoteCreator : NoteCreator() {

    override fun createNote(): Note {

        return AudioNote()

    }

}


fun main() {

    val textNoteCreator = TextNoteCreator()

    val imageNoteCreator = ImageNoteCreator()

    val audioNoteCreator = AudioNoteCreator()


    textNoteCreator.someOperation()

    imageNoteCreator.someOperation()

    audioNoteCreator.someOperation()

}
```

Breve descripción de cada clase

- **Producto**
  - Interfaz, común a todos los objetos que pueda producir la clase creadora y subclases. Define el comportamiento común para todos los tipos de notas.
- **Producto Concretos:**
  - Distintas implementaciones de la interfaz del producto. Representan un tipo específico de nota.
- **Creadora**

- Devuelve nuevos objetos de producto. Siendo del mismo tipo que la interfaz del producto.

- **Creadores Concretos**
  - Sobrescriben el Factory Method base, de modo que devuelva un tipo diferente de producto. El tipo específico de nota que corresponde

3. Preparación para el futuro

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

El patrón Factory Method separa el código de construcción de producto del código que hace uso del producto. Por ello, es más fácil extender el código de construcción de producto de forma independiente al resto del código.

Tiene el principio de abierto/cerrado. Lo que permite incorporar nuevos productos en el programa sin descomponer el código cliente existente.

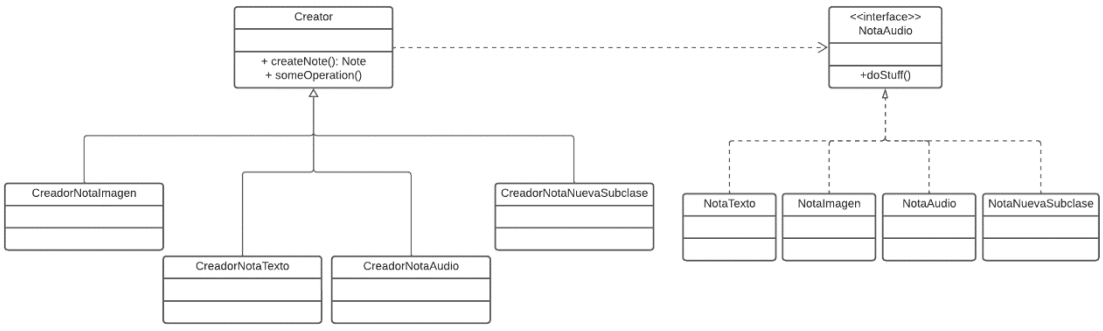
¿Qué partes de tu aplicación tendrías que modificar?

Solo se tendrá que crear una nueva subclase creadora y sobrescribir el Factory Method que contiene. Aparte del producto concreto.

¿Qué nuevas clases tendrías que añadir?

Para incorporar un nuevo tipo de nota en la aplicación, sería necesario crear una nueva subclase creadora específica para ese tipo de nota y sobreescribir el Factory Method correspondiente. Esto asegura que se puedan crear instancias del nuevo tipo de nota de manera coherente dentro del sistema existente.

En el diagrama de clases, esta adición se vería reflejada como una extensión natural de las clases existentes, mostrando la nueva subclase creadora y la clase del producto específico relacionadas con el nuevo tipo de nota:



Selene González Curbelo
Mariana Surós Álvarez