Camilo Linares 202122820
Daniel Pedroza 202123283

Información general del proyecto, deben incluir la URL para consultar el proyecto
Patron- Template
Proyecto elegido- BankingSystem https://github.com/ssoad/BankingSystem

El proyecto BankingSystem tiene como objetivo desarrollar un método de automatizar el proceso de creación de cuentas en los sistemas de bancos ya que actualmente el proceso se hace de forma manual. Sus funcionalidades principales es que a través de la aplicación se pueda guardar la información bancaria con eficacia, ver la información 100% precisa de los usuarios, hacer transacciones a otros bancos. Es una aplicación para avanzar en el desarrollo de los bancos.

Información y estructura del fragmento del proyecto donde aparece el patrón. No se limiten únicamente a los elementos que hacen parte del patrón: para que tenga sentido su uso, probablemente van a tener que incluir elementos cercanos que sirvan para contextualizar.
El proyecto se divide en 5 clases esenciales, además de 1 clase para la serialización y 4 clases para manejar las excepciones. Las partes del proyecto que utilizan el patrón son las 5 clases esenciales y las 4 clases que se utilizan para manejar las excepciones. De las 5 clases en la que es utilizada como template es la clase BankAccount la cual busca simular una cuenta de banco y se construye con un nombre, balance y balance mínimo. Las demás clases implementan a BankAccount como clase padre para demostrar instancias diferentes de cuentas de bancos tales como SavingsAccount, StudentAccount y CurrentAccount.

Información general sobre el patrón: qué patrón es y para qué se usa usualmente
El método de plantilla es un patrón de diseño de comportamiento que define el esqueleto de un algoritmo en la superclase pero permite que las subclases anulen pasos específicos del algoritmo sin cambiar su estructura. El patrón de método de plantilla sugiere dividir un algoritmo en una serie de pasos, convertir esos pasos en métodos y colocar una serie de llamadas a esos métodos en un único método de plantilla. Los pasos pueden ser abstractos o tener una implementación predeterminada. Para usar el algoritmo, se espera que el cliente proporcione su propia subclase, implemente todos los pasos abstractos y anule algunos de los pasos opcionales si es necesario (pero no el método del modelo en sí).

Información del patrón aplicado al proyecto: explicar cómo se está utilizando el patrón dentro del proyecto
El patrón de método de plantilla está siendo utilizado como una forma de generalizar el comportamiento de la clase BankAccount para poder ser utilizado por las demás clases hijas de BankAccount, dejando los métodos withdraw() y deposit() supuestos en la herencia de clases. También se utiliza el template de excepción para dejar puestas ciertas excepciones

como AccNotFound y InvalidAmount las cuales heredan de la clase Exception para permitir un mejor manejo de error frente a ciertos errores con mensajes personalizados.

¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?

El patrón tiene sentido en ese proyecto ya que se necesita crear dos cosas diferentes tipos de cuentas de bancos por lo que hereden de una cuenta que sirva como base hace que el proceso de creación sea más eficiente, además de la facilitar la adaptación a diferentes cambios que se quieran hacer a las cuentas. El proceso que se lleva a cabo en este proyecto para las cuentas es corto por lo que mantener la template del método no es complicado. También tenemos que una de las clases que se genera con la plantilla también está sirviendo como plantilla para otras clases de cuentas de banco que puedan derivar como subclases de los otros tipos de cuenta de banco como cuenta de estudiante deriva de cuenta de ahorros.

¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?

Una de las desventajas que tiene es que si se manejan de forma muy genérica los métodos de la plantilla cuando se quieran implementar otros tipos de cuentas que tengan variaciones y métodos diferentes a las cuentas principales tocaría hacer una pirámide de clases y subclases que puedan dividir con eficacias las características de todas los tipos de cuentas bancarias. Esto genera que el código se tenga que cambiar mucho por lo que pierde la eficiencia del método. También si llega a largar mucho los templates se puede complicar el entendimiento del código a la hora de implementar todos los templates.

¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?

Este proyecto hubiera podido ser resuelto con la utilización de interfaces en un método más acorde con el Factory Method. Esto hubiera podido dejar las clases como StudentAccount y SavingsAccount con más libertad de implementar sus propias excepciones en cuanto al método withdraw(). Además hubiera facilitado la creación de otros tipos de cuentas las cuales no tengan que implementar los dos métodos withdraw() y deposit() dado que la implementación de los métodos se define por las interfaces que implementan y no por la superclase de la que heredan.