Entrega Parcial 4

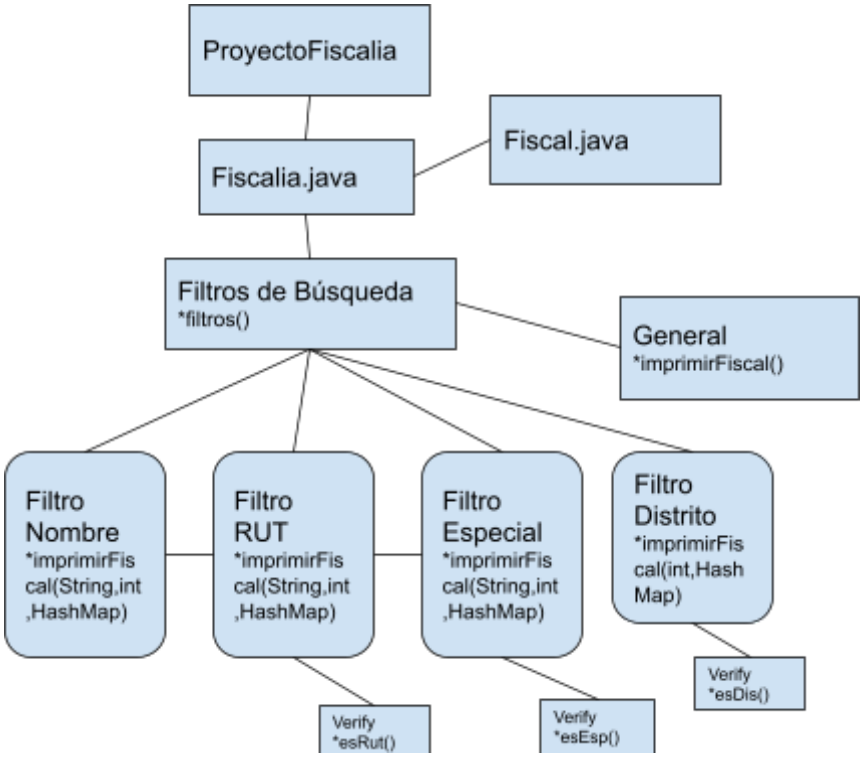
Nombres: Valentina San Martín, Marlene Lagos y Matías Mujica

Fecha de Entrega: 25 de mayo de 2021

Repositorio GitHub de Versiones de Proyecto

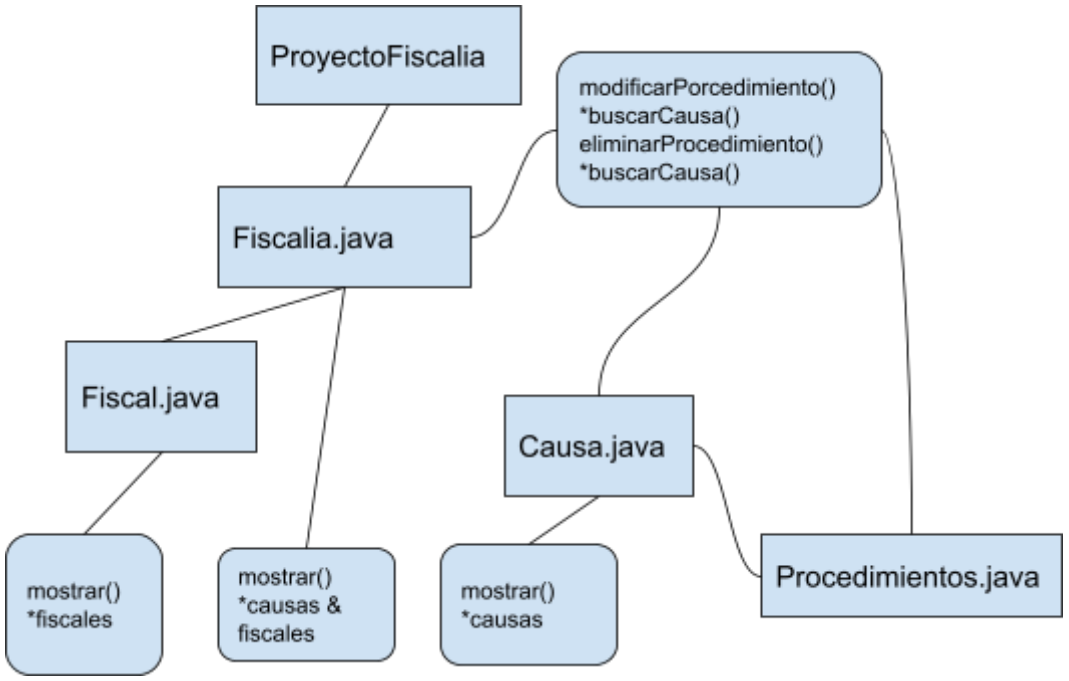
Distritos Electorales de Chile (28), Wikipedia

EP 4.1. Se deben incluir al menos 2 funcionalidades propias que sean de utilidad para el negocio (distintas de la inserción,edición, eliminación y reportes).



Como se puede apreciar en el mapa conceptual mostrado, nos enfocamos en filtros para nuestros fiscales, en el futuro será para causas y filtros de búsqueda más específicos, por el momento se filtra el fiscal por RUT, Nombre, Distrito y Especialidad junto a su métodos de verificación particular, para mayor comprensión, favor revisar Imagen diagrama UML.

EP 4.2. Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos.

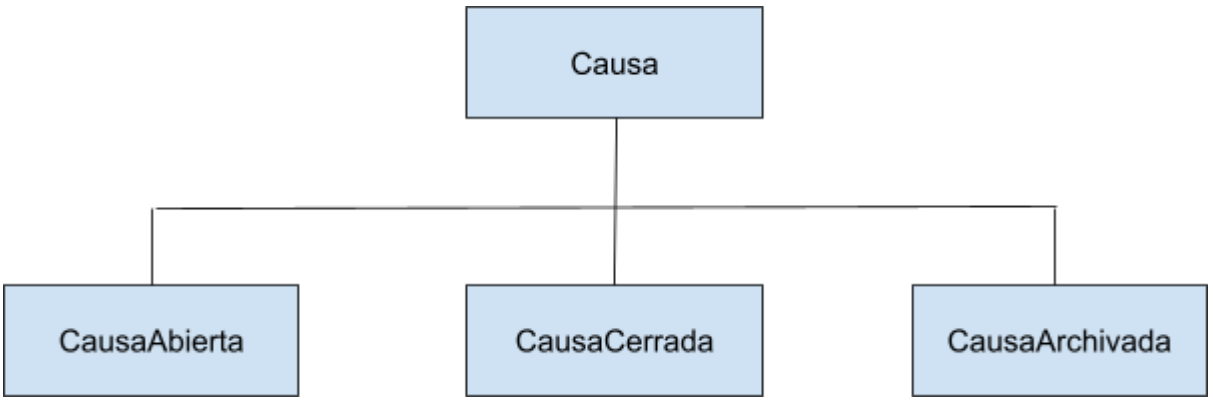


La sobre escritura se presenta en muchas partes del codigo, favor revisar imagen UML, lo mostrado en el diagrama de arriba es una representación del método mostrar(), que son diferentes dependiendo de la clase (ie. mostrar en clase Causa, mostrará atributos únicos de cada causa) lo que facilita el orden y estructura.

Otros métodos de sobre escritura incluyen:

modificarProcedimiento(), Metodo eliminarProcedimiento(), Métodos en clase Formato.java, Filtros de Búsqueda con override en Fiscal.

EP 4.3. Diseño y codificación de 1 (una) clase abstracta que sea padre de al menos 2 (dos) clases. La clase abstracta debe ser utilizada por alguna otra clase (contexto)



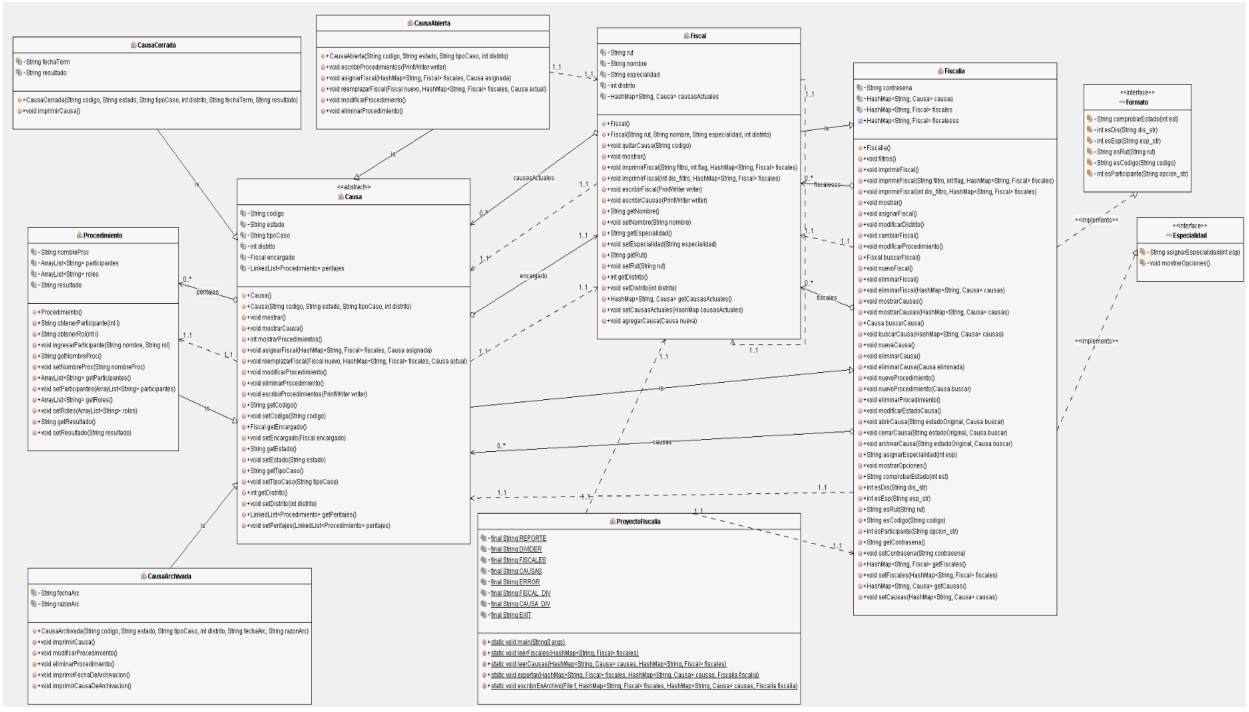
La clase Causa fue elegida como la clase abstracta y sus clases hijas son: CausaAbierta, CausaCerrada y CausaArchivada.. Al realizar este cambio, también se tuvo que modificar la función nuevaCausa(), se tuvo que agregar la opción “Modificar el estado de una causa” al menú y junto con esto se tuvieron que agregar las funcionalidades modificarEstadoCausa(), abrirCausa(String estadoOriginal, Causa buscar), cerrarCausa(String estadoOriginal, Causa buscar) y archivarCausa(String estadoOriginal, Causa buscar).

La clase abstracta fue utilizada en las clases Fiscalia, Fiscal y ProyectoFiscalia como parte del HashMap<String, Causa> y como variable de instancia en algunas de sus funciones.

EP 4.4.Diseño y codificación de 1 (una) interfaz que sea implementada por al menos 2 (dos) clases. La interfaz debe ser utilizada por alguna otra clase (contexto)

En el proyecto se implementan las interfaces Especialidad y Formato. La interfaz Especialidad ayuda a registrar todas las especialidades posibles para que causa y fiscal sólo puedan actuar bajo esos parámetros. Por otro lado la interfaz Formato guarda el formato que deberían tener los códigos de causa y los rut de fiscales para poder comprobarlos y que así los datos ingresados sean correctos.

Diagrama UML



Información General

- **Pequeña Introducción a diagrama y código:**
- Clase Main -> ProyectoFiscalia.java
- Padre -> Fiscalia.java, implementa interfaces Formato.java & Especialidad.java para correcta escritura y confirmación de datos
- Hijos -> Fiscal.java & Causa.java, la ultima es clase abstracta de las siguientes
- Hijos de Causa.java -> Procedimientos.java, CausaAbierta.java, CausaCerrada.java y CausaArchivada.java
- **Para mejor visibilidad de Archivo UML por favor revisar GitHub->Imágenes->Diagrama UML, también disponible en archivo zip.**
- **Documentación de actualizaciones se encuentra en GitHub->Documentos->changelog.txt**
- **Documentación general se puede encontrar en nuestro GitHub: https://github.com/Arakito14/Fiscalia**