**ALTERNATIVAS**

**Clase DocumentoComercial**

Crear una clase para representar un documento comercial. Que tenga de atributos “tipoDocumentoComercial” y “numeroDocumento”

Ventajas

Delegamos el conocimiento de esos 2 atributos a otra clase y no a la clase “OperaciónDeEgreso”

Desventajas

Estaríamos creando objetos sin funcionalidades que solo almacenan datos, algo que se puede hacer en una base de datos.

**Proveedores de una operación de egreso**

Crear una lista para representar un conjunto de Proveedores para una misma operación de egreso. Cambiando en clase “OperacionDeEgreso” el atributo “Proveedor proveedor” por “List<EntidadBase> proveedores”.

Ventajas

Nos da mas flexibilidad a la hora de generar una OperacionDeEgreso. Podemos relacionar uno o varios proveedores con una misma operación.

Desventajas

Podríamos estar modelando algo que no se valla a usar.

**DNI en vez de CUIL**

Usar numero de DNI en vez de CUIL como atributo para un Proveedor.

Ventajas

Podemos usar una variable long en vez de int

Desventajas

Puede haber colisiones entre DNIs. Es mas confiable usar un CUIL

**Mas tipos de documentos comerciales**

Añadir al enum “TipoDocumentoComercial” más tipos como:

remito, nota de débito, nota de crédito

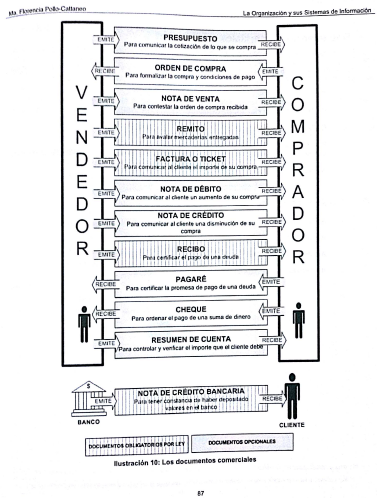
Según lo referenciado en el enunciado del TP:

<http://www.mundoit.com.ar/documentos-comerciales-de-una-empresa/>

Y:

Presupuesto, nota de venta y resumen de cuenta.

Tal como se puede apreciar en la siguiente captura del libro “La organización y sus sistemas de información” de la profesora Pollo Cattaneo. Ya que, en una operación de egreso de una organización, se pueden recibir esos tipos de documentos.



Ventajas

Nos da mas flexibilidad para adaptar el proyecto a varias organizaciones

Desventajas

Podríamos estar generando tipos que no vallamos a usar. Podría estar fuera del alcance del proyecto.

**Método de pago con Cheque**

Agregar un método de pago por cheque, tal como se ejemplifica en el punto 5 del enunciado

Ventajas

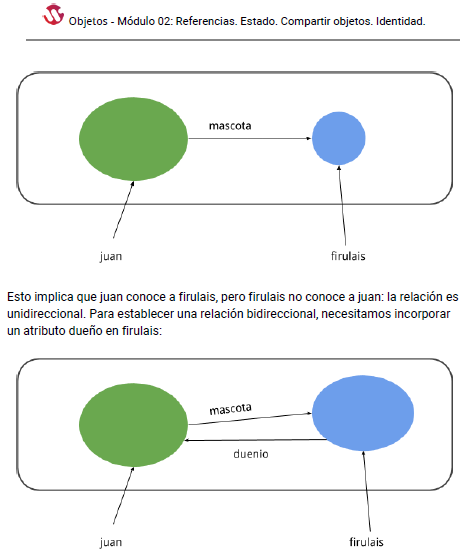
Se agrega flexibilidad a la hora de generar operaciones de egreso.

Desventajas

No respetan los medios de pago de mercado libre.

**Relación bidireccional entre Entidad Jurídica y Entidad Base**

Generar una relación bidireccional entre ambas entidades. Se podría inicializar una referencia de Entidad base a Entidad Jurídica cuando las primeras son agregadas a la lista de entidades asociadas a la Entidad Jurídica. Así una Entidad base conoce a la única entidad jurídica a la que está asociada; a la vez que una entidad jurídica conoce a todas las entidades base que tiene asociadas (en su lista). Un ejemplo grafico de lo que se trata de decir extraído de “Módulo 02: Referencias. Estado. Compartir objetos. Identidad.” del prof. Dodino:



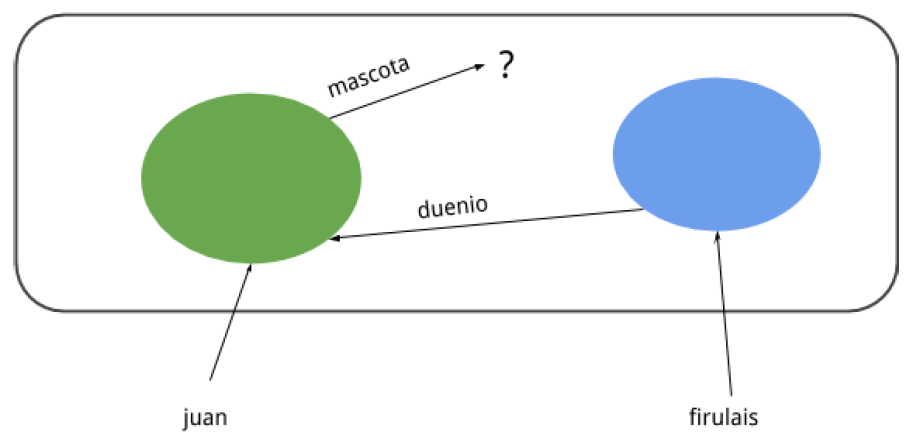
Ventajas

Permite que se puedan mandar mensajes entre las instancias de las 2 clases

Desventajas

Puede que por el momento no sea muy útil, ya que no conocemos el comportamiento de ambas entidades.

Puede ocurrir que, por una mala sincronización, quede una referencia libre o vacía.



**Clasificación de Empresas en tiempo de ejecución**

Calcular el tipo de empresa según lo estipulado por la AFIP:

<https://pymes.afip.gob.ar/estiloAFIP/pymes/ayuda/default.asp>

ventajas

no hace falta actualizar el tipo de organización, se genera calculando el valor de las ventas totales anuales

desventajas

todavía no tenemos la información suficiente para poder calcular ese valor

**Doble validación de contraseña**

Agregar una validación extra al constructor de la clase “Usuario” además del que esta el “BuilderUsuario”

Ventajas

Permite que se valide la contraseña si se genera un usuario directamente con el constructor en vez de con el Builder. Algo que puede pasar no intencionalmente.

Desventajas

El mismo código se ejecuta 2 veces en caso de que se creé un usuario con el Builder.

En validadorPasswords utilizamos el patrón Singleton, ya que, al no tener un estado mutable y solo realizar operaciones de chequeo, no trae mayores complicaciones y ahorra tiempo computacional, al evitar tener que releer el archivo de contraseñas cada vez que se genera una nueva instancia.

Sin embargo, no habría problema en generar una nueva instancia de validadorPasswords cada vez que se desee crear una nueva contraseña.

En builderOperacionDeEgreso y builderUsuario utilizamos el patrón builder, ya que vamos a tener diferentes representaciones del objeto a construir y dicho patrón nos permite tener un mayor control sobre el proceso de construcción.

De todas formas, no es algo imprescindible en este momento, por lo que podríamos optar por usar otras formas de construir estos objetos.