

Universidad Nacional
ARTURO JAURETCHE

Universidad Nacional Arturo Jauretche

Ingeniería en Informática

Metodologías de programación II

Informe de Trabajo Integrador

Sistema para librería e imprenta

Autores:

Caro, Nicolas Daniel

Claros Maldonado, Brian Alejandro

Ortiz, Maximiliano Gabriel

Docente:

Cappelletti, Claudia

Julio 2021

Índice

1. Introducción	2
1.1 Objetivos	2
1.2 Contexto del problema a resolver	2
2. Desarrollo	2
2.1 Alcance del desarrollo	2
2.2 Diagramas	3
2.2.1 Diagramas UML	3
2.2.2 Diagrama de secuencia	4
2.3 Herramientas utilizadas	4
2.4 Implementación de la aplicación desarrollada	5
2.4.1 Clases	5
Libreria	5
Producto	8
Impresion	9
TipoDeImpresion	10
AColor	10
BlancoYNegro	11
2.4.2 Ejecución en Workspace	12
2.5 Metodología de trabajo	15
2.6 Posibles implementaciones	16
2.6.1 Frameworks para el desarrollo	16
2.6.2 Casos de uso de Patrones de diseño	16
Patrón Decorador (Decorator)	16
Patrón Builder	18
2.6.3 Uso de repositorios para desarrollo colaborativo	18
3. Conclusiones	19
4. Bibliografía	20

1. Introducción

1.1 Objetivos

El objetivo del trabajo integrador es integrar contenidos de la materia resolviendo un problema ficticio de un negocio, para crear una solución mediante la programación orientada a objetos, bajo en lenguaje Smalltalk y la utilización de metodologías ágiles. Además del uso de diagramas UML y diagramas de secuencia.

1.2 Contexto del problema a resolver

Una librería busca digitalizar el servicio de impresiones y gestión de inventario para una mejor organización, Para ello, se llevará un control de todos los productos que se tienen a la venta con su correspondiente información, pudiendo filtrar por algún campo específico u obtener información del mismo; A su vez, se tendrán almacenadas las impresiones con su cliente asociado permitiendo filtrar: por cliente, estado(impreso/no impreso), entrega(pendiente/entregado). El costo de impresion depende del tipo que sea, blanco y negro o color. Además, se debe contar las ocurrencias de impresiones que se manejan, como también mostrarlas ordenadas.

2. Desarrollo

2.1 Alcance del desarrollo

Para el proyecto nos proponemos a desarrollar un prototipo de un sistema para una librería en el cual tendrá algunas de las funcionalidades básicas y simples utilizando los conocimientos aprendidos en la cursada. Las funciones a desarrollar las pensaremos en base a los conocimientos obtenidos y también en la utilidad para un sistema de una librería.

2.2 Diagramas

2.2.1 Diagramas UML

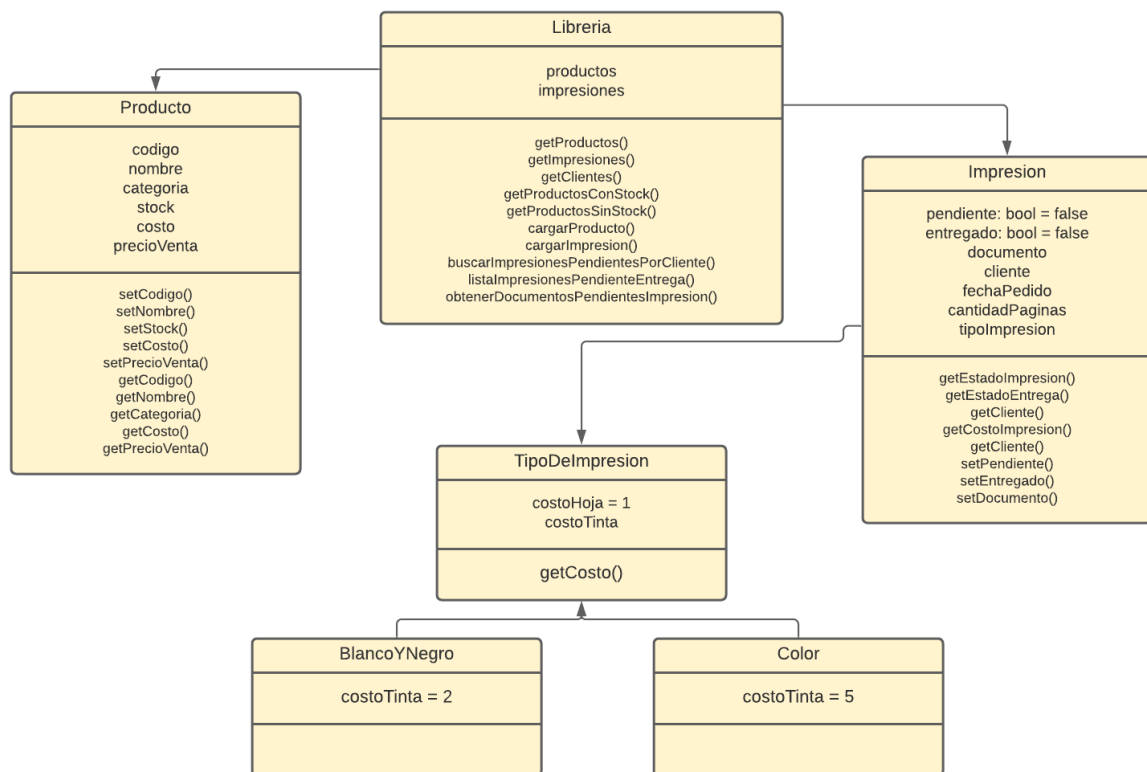


Figura 1. Diagrama UML

2.2.2 Diagrama de secuencia

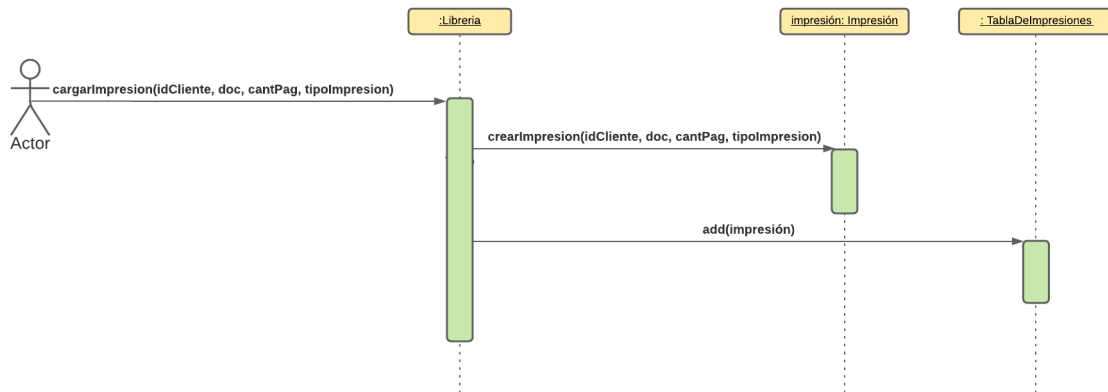


Figura 2. Diagrama de secuencia - Carga de impresión.

Para el desarrollo del proyecto, se propone diseñar el diagrama de flujo para una de las funciones del sistema, en este caso podemos ver en la figura 2 el diagrama de secuencia diseñado para la carga de una impresión.

2.3 Herramientas utilizadas

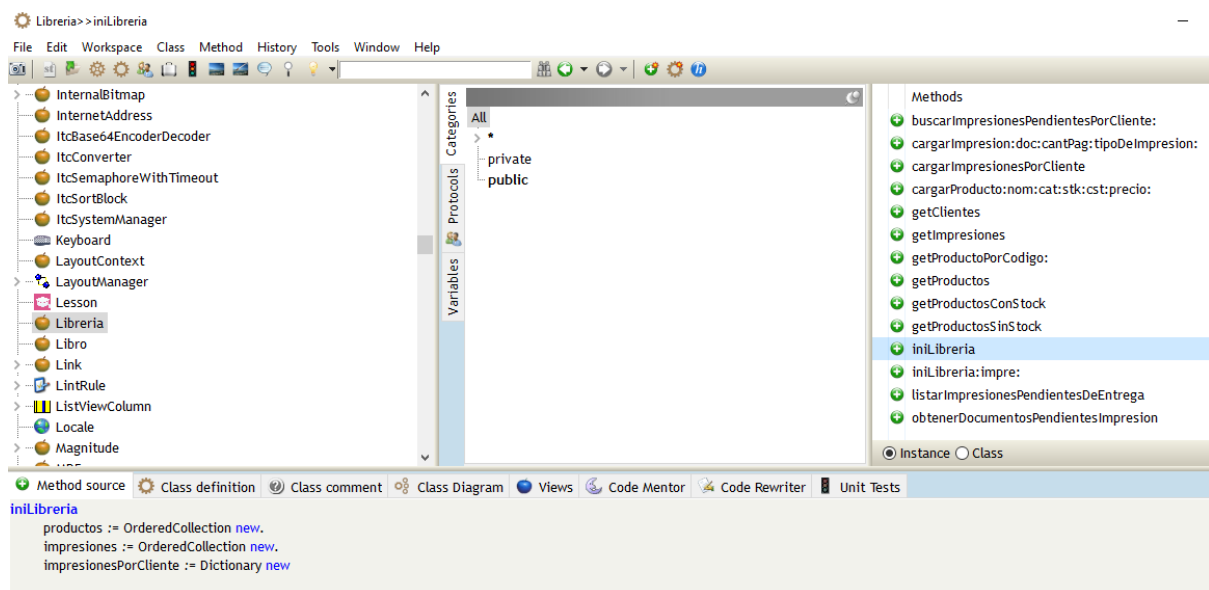
- **LucidChart.** Herramienta de diagramación basada en la web, que permite a los usuarios colaborar y trabajar juntos en tiempo real.
- **Dolphin Smalltalk.** Implementación del lenguaje de programación Smalltalk para Microsoft Windows.
- **Git.** Software de control de versiones
- **GitHub.** Forja para alojar proyectos utilizando el sistema de control de versiones Git. Permite el desarrollo colaborativo remoto.
 - **GitHub Projects.** Herramienta de sistema de tablero y tarjetas para el seguimiento de tareas de un repositorio en GitHub. Posee implementación de metodología KanBan.

2.4 Implementación de la aplicación desarrollada

2.4.1 Clases

Libreria

Clase principal del programa.



Métodos de Instancia

Método	iniLibreria
Descripción	Inicializa una librería, seteando listas ordenadas a las variables de productos, impresiones e impresionesPorCliente.
Captura	<pre>iniLibreria productos := OrderedCollection new. impresiones := OrderedCollection new. impresionesPorCliente := Dictionary new</pre>

Método	cargarImpresion
Descripción	Agrega una impresión a la tabla de impresiones (OrderedCollection).

Captura	<pre> cargarImpresion: unCliente doc: doc cantPag: cantPag tipoDeImpresion: tipoDeImpresion impresion impresion := Impresion crearImpresion: unCliente doc: doc cantPag: cantPag tipoDeImpresion: tipoDeImpresion. impresiones add: impresion </pre>
---------	--

Método	cargarProducto
Descripción	Agrega un producto a la tabla de productos (OrderedCollection).
Captura	<pre> cargarProducto: unCodigo nom: unNombre cat: unCategoria stk: unStock cst: unCosto precio: unPrecio producto producto := Producto crearProducto: unCodigo nom: unNombre cat: unCategoria stk: unStock cst: unCosto precio: unPrecio. productos add: producto </pre>

Método	buscarImpresionesPendientesPorCliente - USO DE SELECT
Descripción	Trae una colección de impresiones pendientes de entrega que coincidan con el cliente pasado por parámetro.
Captura	<pre> buscarImpresionesPendientesPorCliente: unCliente ^impresiones select[:impresion impresion getCliente=unCliente & impresion getEstadoImpresion=true]. </pre>

Método	getProductosConStock- USO DE REJECT
Descripción	Devuelve una colección de productos con stock.
Captura	<pre> getProductosConStock ^productos reject[:producto producto getStock=0]. </pre>

Método	getProductoSinStock- USO DE SELECT
Descripción	Devuelve una colección de productos sin stock.

Captura	<pre>getProductosSinStock ^productos select:[:producto producto getStock=0].</pre>
---------	--

Método	listarImpresionesPendientesDeEntrega USO DE REJECT
Descripción	lista impresiones pendientes de entrega
Captura	<pre>listarImpresionesPendientesDeEntrega ^impresiones reject: [:impresion impresion getEstadoEntrega = true]</pre>

Método	obtenerDocumentosPendientesImpresion USO DE COLLECT
Descripción	Devuelve una lista ordenada de documentos pendientes de impresión
Captura	<pre>obtenerDocumentosPendientesImpresion impresionesPendientes impresionesPendientes:=impresiones select:[:impresion impresion getEstadoImpresion=true]. ^impresionesPendientes collect:[:impresion impresion getDocumento].</pre>

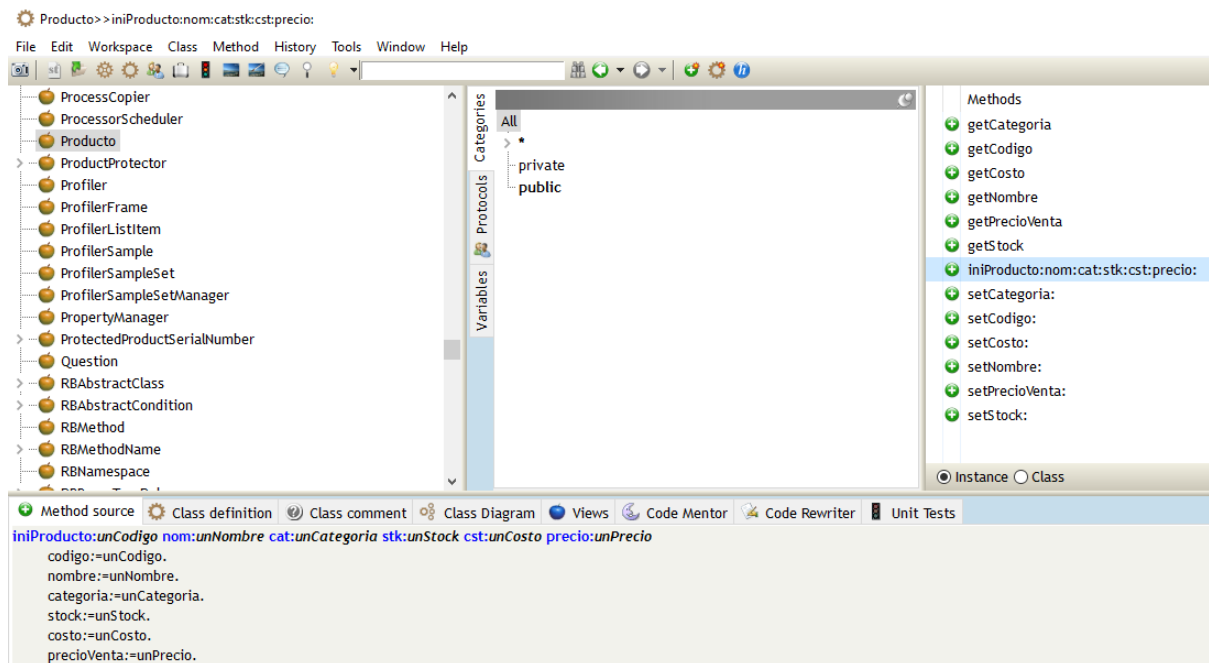
Método	getProductoPorCodigo:unCodigo USO DE DETECT
Descripción	Devuelve un producto por código, si no lo encuentra devuelve nil.
Captura	<pre>getProductoPorCodigo:unCodigo ^productos detect: [:producto producto getCodigo=unCodigo] ifNone:[^nil].</pre>

Método	cargarImpresionesPorCliente USO DE DICCIONARIO
Descripción	Carga en el diccionario “impresiopnesPorCliente” la cantidad de impresiones por cliente. El método recorre la lista de impresión y carga el diccionario con las cantidad de impresiones por cliente. La clave es el id de cliente.

Captura

```
cargarImpresionesPorCliente
  impresiones do:
    [:impresion |
      (impresionesPorCliente includesKey: impresion getCliente)
      ifTrue:
        [impresionesPorCliente at: impresion getCliente
          put: (impresionesPorCliente at: impresion getCliente) + 1]
      ifFalse: [impresionesPorCliente at: impresion getCliente put: 1]]
```

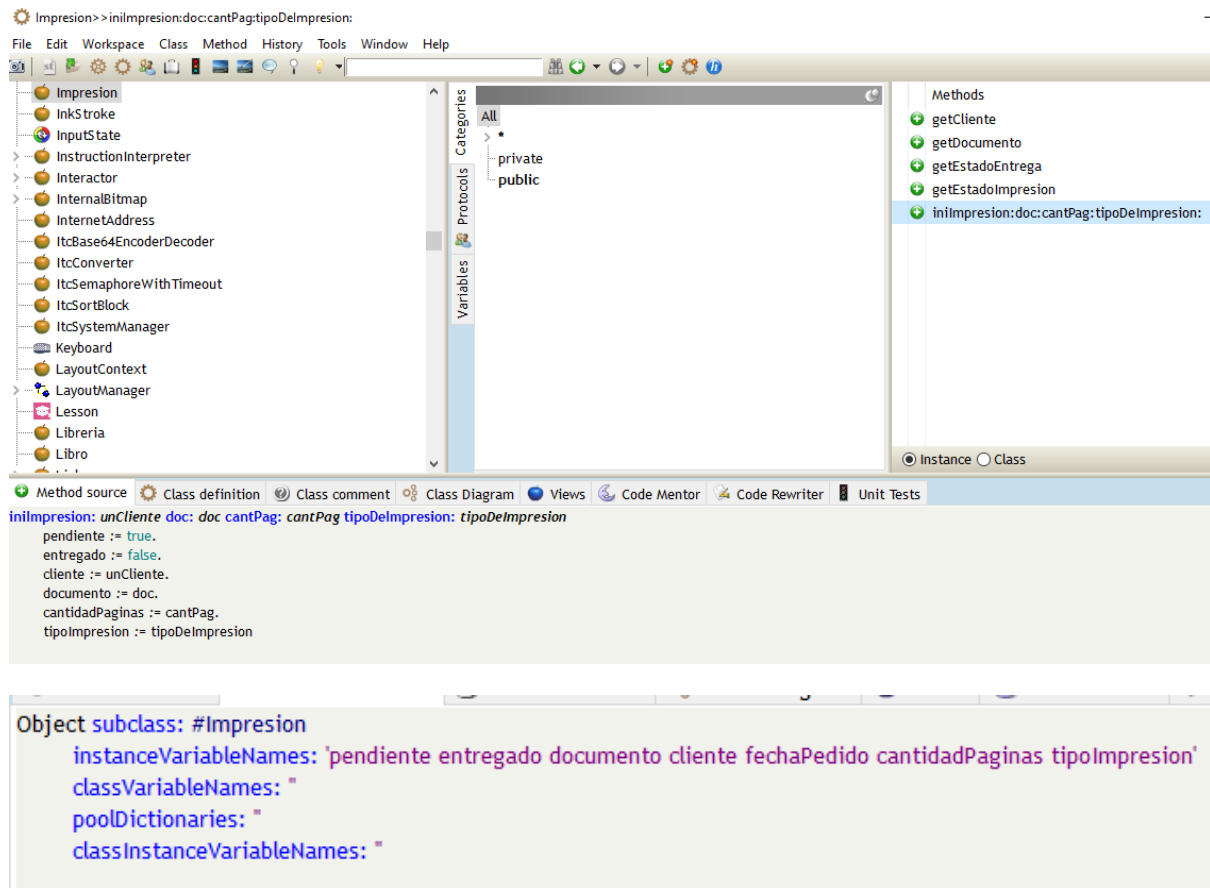
Producto



Los métodos que posee Producto son los set y get de sus propiedades.

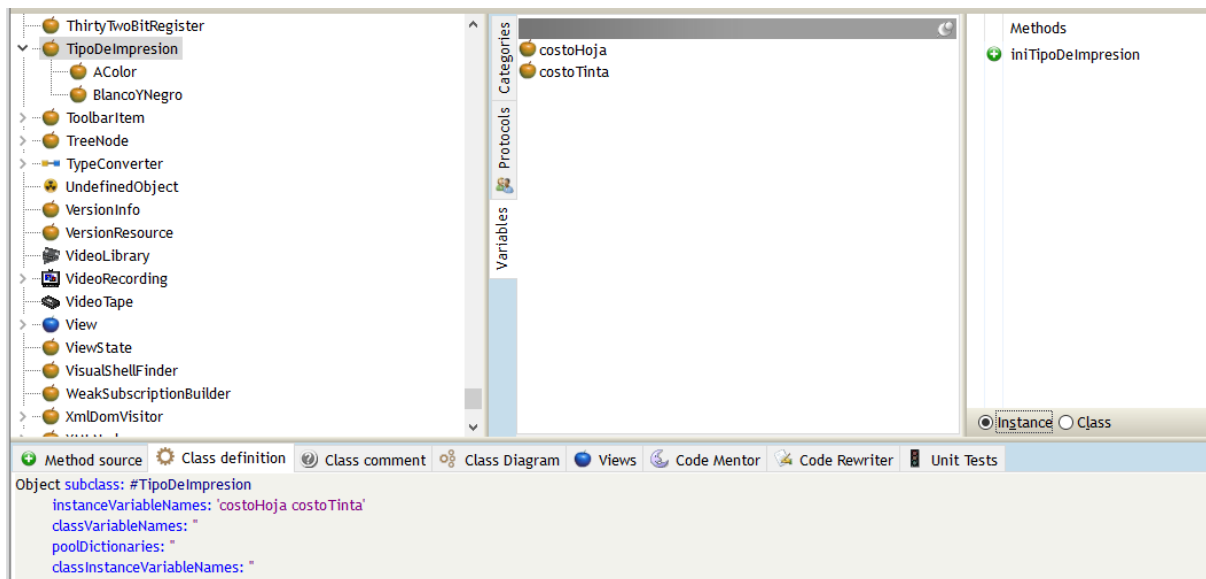
Impresion

Esta clase se encarga de manejar el estado e información de una impresión .



Los métodos que posee son gets de propiedades. Respecto al iniImpresion, este método inicializa una impresión, por defecto tiene los valores de pendiente y entregado en false.

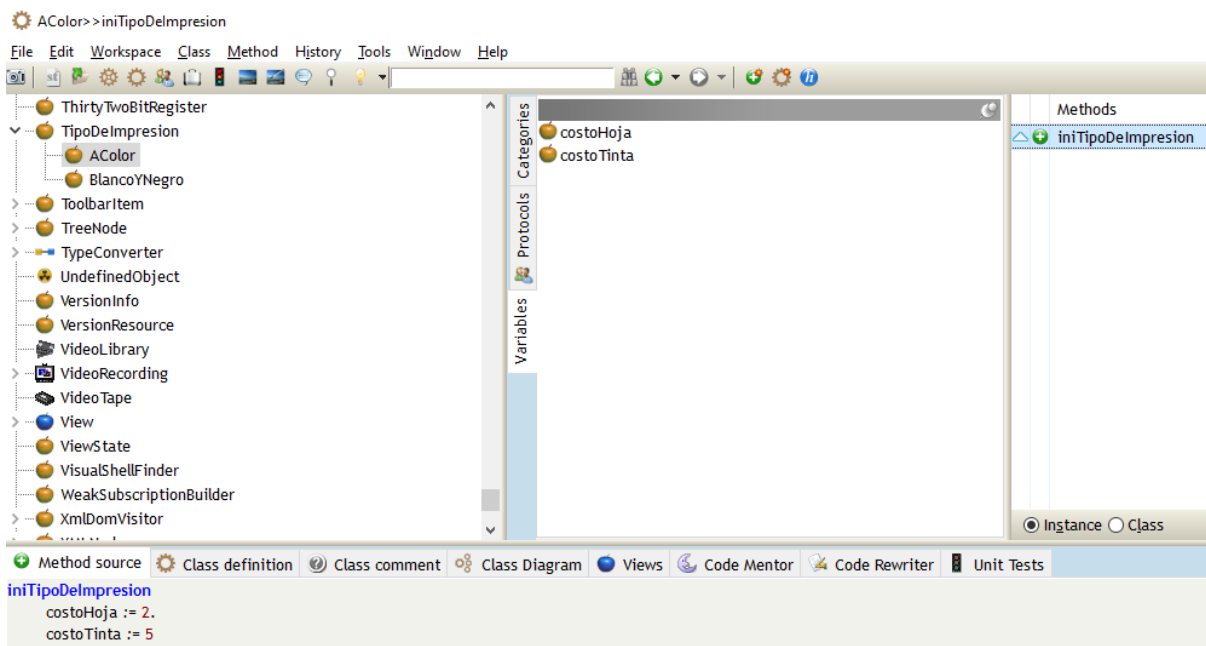
TipoDeImpresion



`iniTipoDeImpresion`
^self implemente by subclass

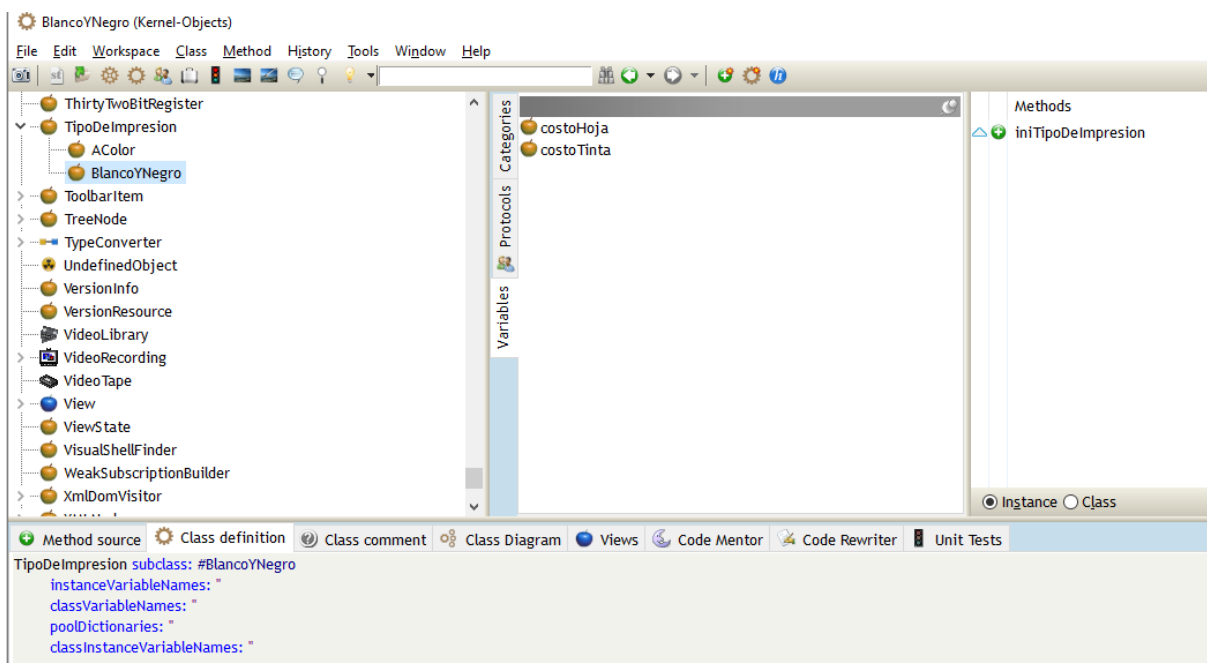
AColor

Es subclase de TipoDeImpresion.



```
TipoDeImpresion subclass: #AColor
instanceVariableNames: ""
classVariableNames: ""
poolDictionaries: ""
classInstanceVariableNames: ""
```

BlancoYNegro



```
iniTipoDeImpresion
costoHoja := 2.
costoTinta := 3
```

2.4.2 Ejecución en Workspace

```

C:\Users\ncaro\Dev\metodologias2-imprenta\demo.st*
File Edit Workspace Tools Window Help
[libreria aColor blancoYNegro]

libreria := Libreria crearLibreria.

aColor := AColor crearTipoDeImpresion.
blancoYNegro := BlancoYNegro crearTipoDeImpresion.

"cargo de Impresiones"
libreria cargarImpresion: 1 doc: 'doc1' cantPag: 15 tipoDeImpresion: blancoYNegro.
libreria cargarImpresion: 2 doc: 'doc2' cantPag: 10 tipoDeImpresion: aColor.
libreria cargarImpresion: 2 doc: 'doc3' cantPag: 20 tipoDeImpresion: blancoYNegro.
libreria cargarImpresion: 3 doc: 'doc4' cantPag: 50 tipoDeImpresion: blancoYNegro.
libreria cargarImpresion: 4 doc: 'doc5' cantPag: 21 tipoDeImpresion: aColor.
libreria cargarImpresion: 1 doc: 'doc6' cantPag: 15 tipoDeImpresion: aColor.
libreria cargarImpresion: 5 doc: 'doc7' cantPag: 6 tipoDeImpresion: blancoYNegro.

"cargo de Productos"
libreria cargarProducto: 1 nom: 'Lapicera' cat: 'Utiles' stk: 100 cst: 15 precio: 18.
libreria cargarProducto: 2 nom: 'Goma' cat: 'Utiles' stk: 20 cst: 10 precio: 13.
libreria cargarProducto: 3 nom: 'Lapiz' cat: 'Utiles' stk: 50 cst: 12 precio: 15.
libreria cargarProducto: 4 nom: 'Regla' cat: 'Utiles' stk: 0 cst: 10 precio: 12.

"uso del select"
libreria buscarImpresionesPendientesPorCliente: 2.

"uso del reject"
libreria listarImpresionesPendientesDeEntrega.

"uso del collect"
libreria obtenerDocumentosPendientesImpresion.

"uso de reject"
libreria getProductosConStock.

"uso de select"
libreria getProductosSinStock.

"uso de detect"
libreria getProductoPorCodigo: 2.

"uso del diccionario"
libreria cargarImpresionesPorCliente.

```

**método
a
inspeccionar**


"uso del select"







libreria buscarImpresionesPendientesPorCliente: 2.

Resultado Inspect	Published Aspect		Value
	self		an OrderedCollection
			an Impresion
	getClient	123	2
	getDocumento	'abc'	'doc2'
	getEstadoEntrega	✗	false
	getEstadoImpresion	✓	true
			an Impresion
	getClient	123	2
	getDocumento	'abc'	'doc3'
	getEstadoEntrega	✗	false
	getEstadoImpresion	✓	true

método a inspeccionar	"uso del reject" libreria listarImpresionesPendientesDeEntrega.		
Resultado Inspect	Published Aspect		Value
	self		an OrderedCollection
			an Impresion
	getClient	123	1
	getDocumento	'abc'	'doc1'
	getEstadoEntrega	✗	false
	getEstadoImpresion	✓	true
			an Impresion
			an Impresion
			an Impresion
			an Impresion
	getClient	123	4
	getDocumento	'abc'	'doc5'
	getEstadoEntrega	✗	false
	getEstadoImpresion	✓	true
			an Impresion
			an Impresion
	getClient	123	5
	getDocumento	'abc'	'doc7'
	getEstadoEntrega	✗	false
	getEstadoImpresion	✓	true

método a inspeccionar	"uso del collect" libreria obtenerDocumentosPendientesImpresion.		
-----------------------------	--	--	--

Resultado Inspect	Published Aspect		Value
	[-] self	 an OrderedCollection	
		abc	'doc1'
		abc	'doc2'
		abc	'doc3'
		abc	'doc4'
		abc	'doc5'
		abc	'doc6'
		abc	'doc7'

método a inspeccionar	"uso de detect" libreria getProductoPorCodigo: 2.																
Resultado Inspect	<table><tr><th>Published Aspect</th><th>Value</th></tr><tr><td> self</td><td> a Producto</td></tr><tr><td>getCategoria</td><td>abc 'Utiles'</td></tr><tr><td>getCodigo</td><td>123 2</td></tr><tr><td>getCosto</td><td>123 10</td></tr><tr><td>getNombre</td><td>abc 'Goma'</td></tr><tr><td>getPrecioVenta</td><td>123 13</td></tr><tr><td>getStock</td><td>123 20</td></tr></table>	Published Aspect	Value	 self	 a Producto	getCategoria	abc 'Utiles'	getCodigo	123 2	getCosto	123 10	getNombre	abc 'Goma'	getPrecioVenta	123 13	getStock	123 20
Published Aspect	Value																
 self	 a Producto																
getCategoria	abc 'Utiles'																
getCodigo	123 2																
getCosto	123 10																
getNombre	abc 'Goma'																
getPrecioVenta	123 13																
getStock	123 20																

método a inspeccionar	"uso del diccionario" libreria cargarImpresionesPorCliente.		
------------------------------	---	--	--

Resultado Inspect	impresionesPorCliente		a Dictionary	
	1	☢ nil		
	+ 2	🍏 1 -> 2		
	+ 3	🍏 2 -> 2		
	+ 4	🍏 3 -> 1		
	+ 5	🍏 4 -> 1		
	+ 6	🍏 5 -> 1		
	7	☢ nil		

2.5 Metodología de trabajo

Para realizar el trabajo se utilizó la **metodología ágil KanBan**. Esta metodología se adapta correctamente para el trabajo, ya que es un equipo reducido y, el software es acotado y se fue incrementando de manera incremental.

- Es muy fácil de implementar, se realiza el seguimiento del estado del proyecto de forma muy intuitiva mediante las tarjetas.
- No existen roles.
- Posee WIP (Work In Progress), una forma de poner un límite a las tareas en curso, para que todo el equipo trabaje equitativamente. La implementación con herramientas en tiempo real online es muy rápida y fácil. En el desarrollo se implementó GitHub Project.

Por otra parte, la implementación se basó en la siguiente forma:

- Sistema de 4 tableros: Backlog, To do, In Progress, Done
- En el tablero “In Progress” se estableció un WIP de 3, para asegurar que el equipo trabaje equitativamente.
- Como herramienta, se utilizó GitHub Projects

- Se crearon tarjetas con las tareas a realizar, a medida que eran propuestas, en caso que el equipo la aprueba (mediante debate), hacía su paso a To Do, y luego continúa el ciclo hasta llegar a “Done”.

2.6 Posibles implementaciones

2.6.1 Frameworks para el desarrollo

→ .NET Framework

- ◆ .NET Framework es un entorno de ejecución administrado para Windows que proporciona una variedad de servicios a sus aplicaciones en ejecución. Consta de dos componentes principales: Common Language Runtime (CLR), que es el motor de ejecución que maneja las aplicaciones en ejecución, y .NET Framework Class Library, que proporciona una biblioteca de código probado y reutilizable que los desarrolladores pueden llamar desde sus propias aplicaciones.

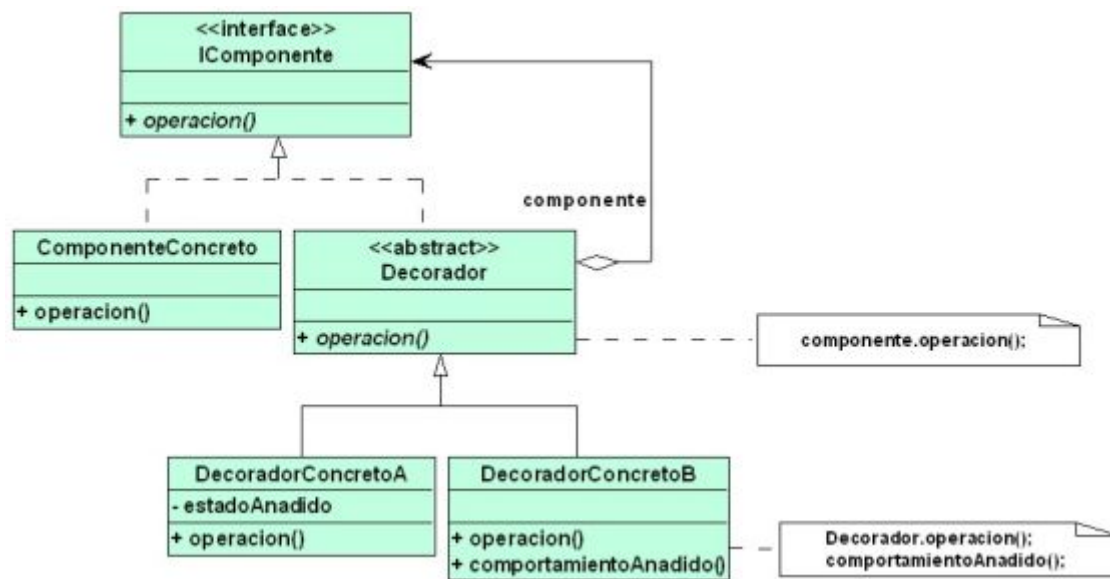
→ Entity Framework

- ◆ Entity Framework Core es un mapeador de bases de datos de objetos moderno para .NET. Admite consultas LINQ, seguimiento de cambios, actualizaciones y migraciones de esquemas. EF Core funciona con muchas bases de datos, incluidas SQL Database (local y Azure), SQLite, MySQL, PostgreSQL y Azure Cosmos DB.

2.6.2 Casos de uso de Patrones de diseño

Patrón Decorador (Decorator)

Decorator es un patrón de diseño estructural que te permite añadir funcionalidades a objetos colocando estos objetos dentro de objetos encapsuladores especiales que contienen estas funcionalidades.



Aplicabilidad:

- Utiliza el patrón Decorator cuando necesites asignar funcionalidades adicionales a objetos durante el tiempo de ejecución sin descomponer el código que utiliza esos objetos.
- El patrón Decorator te permite estructurar tu lógica de negocio en capas, crear un decorador para cada capa y componer objetos con varias combinaciones de esta lógica, durante el tiempo de ejecución. El código cliente puede tratar a todos estos objetos de la misma forma, ya que todos siguen una interfaz común.
- Utiliza el patrón cuando resulte extraño o no sea posible extender el comportamiento de un objeto utilizando la herencia.

Uso en el programa:

- Se podría implementar este patrón para asignarle decorados adicionales a una impresión, modificando el “tipo de impresión”, tales como “Anillado”, “Tapa blanda”, “Tapa Dura”. Estos van decorando la impresión y modificando el valor del precio, la función operación representada en el diagrama UML del patrón, en nuestra implementación sería **getCosto()**.

Patrón Builder

Builder es un patrón de diseño creacional que nos permite construir objetos complejos paso a paso. El patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción.

Aplicabilidad:

- Utiliza el patrón Builder cuando quieras que el código sea capaz de crear distintas representaciones de ciertos productos (por ejemplo, casas de piedra y madera).
- El patrón Builder se puede aplicar cuando la construcción de varias representaciones de un producto requiera de pasos similares que sólo varían en los detalles.
- La interfaz constructora base define todos los pasos de construcción posibles, mientras que los constructores concretos implementan estos pasos para construir representaciones particulares del producto. Entre tanto, la clase directora guía el orden de la construcción.

Uso en el programa:

- Se podría implementar este patrón para la construcción de Impresiones. Ya que las existen distintos tipos, y si se agrega junto al patrón decorador. Se puede tener distintos paquetes de impresiones. Por ejemplo: blanco y negro y anillado, color, anillado y tapa dura, portada a color, distinto tipo de anillado, etc.

2.6.3 Uso de repositorios para desarrollo colaborativo

El equipo de desarrollo utilizó Git y GitHub para poder estar sincronizados y poder obtener constantemente los cambios en el código. La descripción sobre las herramientas se encuentra en el apartado 2.3 Herramientas.

Para ello, un integrante utilizó Git de forma local para almacenar la imagen y archivos de Workspace, que luego estos fueron subidos al sistema de repositorios colaborativos GitHub, para que el resto pueda descargar y trabajar en el mismo código.

Cuando un integrante realiza cambios en el código o imagen, debería realizar un ***commit*** y subir los cambios al repositorio remoto mediante el comando “*git push origin master*”. Los demás integrantes debían actualizar al último cambio mediante “*git pull*”.

3. Conclusiones

Pudimos realizar un desarrollo utilizando un lenguaje no conocido por el equipo, el cual está orientado a objetos, pudimos repasar y poder clarificar aquellos conceptos de los cuales ya teníamos conocimiento sobre el paradigma.

En el trabajo, nos concentramos en utilizar algunos conocimientos aprendidos en la cursada sobre colecciones y herencia. Pudimos utilizar algunas sentencias proporcionadas por SmallTalk para mapear, filtrar, buscar objetos de las colecciones y además utilizar herencia para algunas clases con propiedades y métodos en común.

4. Bibliografía

- ❖ Gamma, Erich (2002), “Patrones de diseño: elementos de software orientado a objetos reutilizables”. En: Pearson Educación.
- ❖ Refactoring Guru. “Decorator”. Disponible en:
<https://refactoring.guru/es/design-patterns/decorator> [Consulta: 6/07/2021]
- ❖ Refactoring Guru. “Builder”. Disponible en:
<https://refactoring.guru/es/design-patterns/builder> [Consulta: 6/07/2021]
- ❖ GitHub. “About”. Disponible en: <https://github.com/about> [Consulta: 6/07/2021]
- ❖ Microsoft. “What is .NET? An open source developer platform”. Disponible en:
<https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet> [Consulta: 6/07/2021]
- ❖ Microsoft. “Entity Framework documentation”. Disponible en:
<https://docs.microsoft.com/en-us/ef/> [Consulta: 6/07/2021]