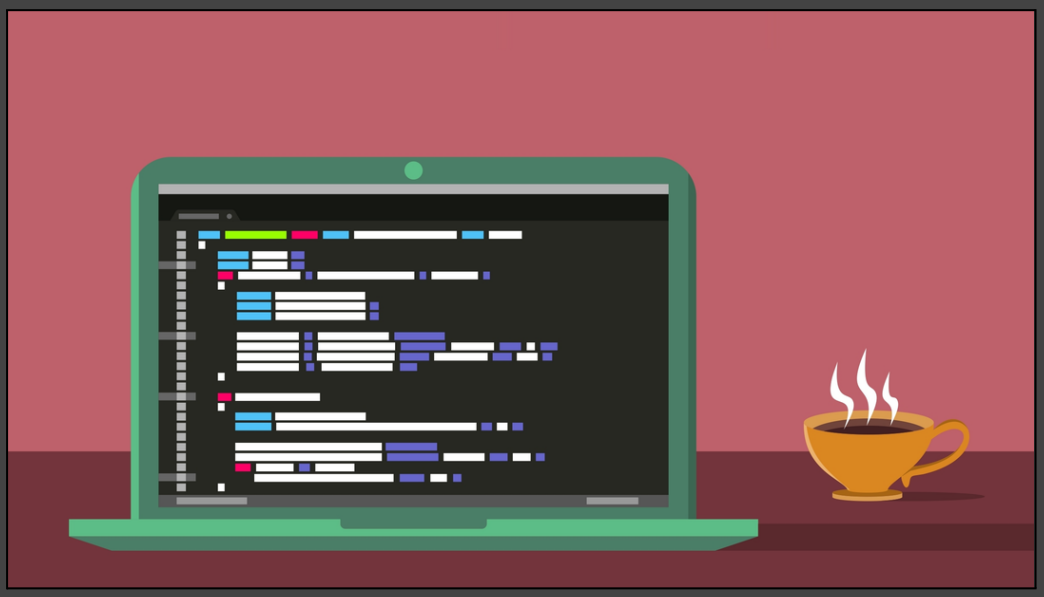
Universidad Nacional
ARTURO JAURETCHE

METODOLOGÍAS DE PROGRAMACIÓN II

1er Cuatrimestre 2022



TRABAJO PRÁCTICO
INTEGRADOR SMALLTALK

Alumnos: **Agustin Ojeda, Leandro Solano**

Profesora: **Claudia Cappelletti**

Índice:

Objetivos

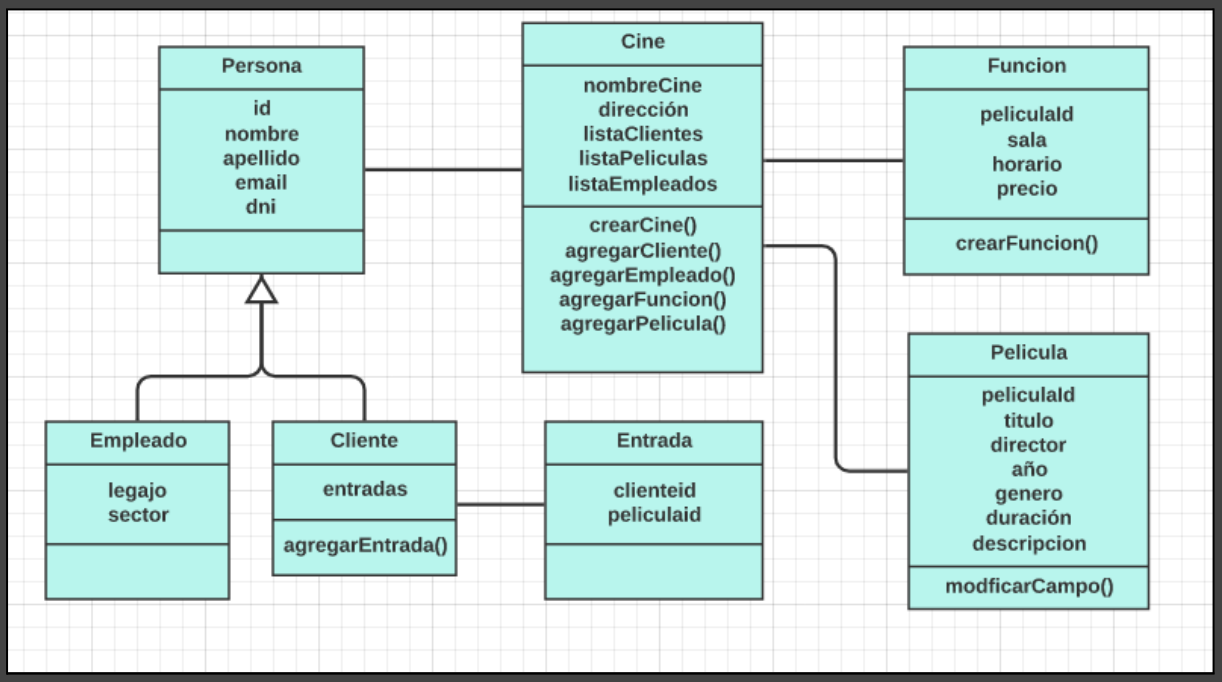
El presente trabajo es una aplicación de software, que tiene su foco en poder brindar y administrar los diferentes servicios dados por un cine. Esto incluye mostrar las diferentes películas disponibles a sus clientes, como también da la posibilidad de comprar entrada/s de una película seleccionada.

El software desarrollado puede ser usado tanto para este negocio (elegido por nosotros y aprobado por la docente de la materia) como para cualquier que posea una actividad similar. En donde sea se requiera la administración de entradas a un servicio, como podría ser un teatro, recitales, eventos, entre otros.

Fue implementado con el lenguaje de programación Smalltalk, con el aprendizaje adquirido durante la cursada de Metodologías de Programación II a cargo de la profesora Claudia Cappelletti. También se hizo uso de una metodología de trabajo para organizar y gestionar el proyecto desarrollado en equipo.

Especificación

Diagrama UML

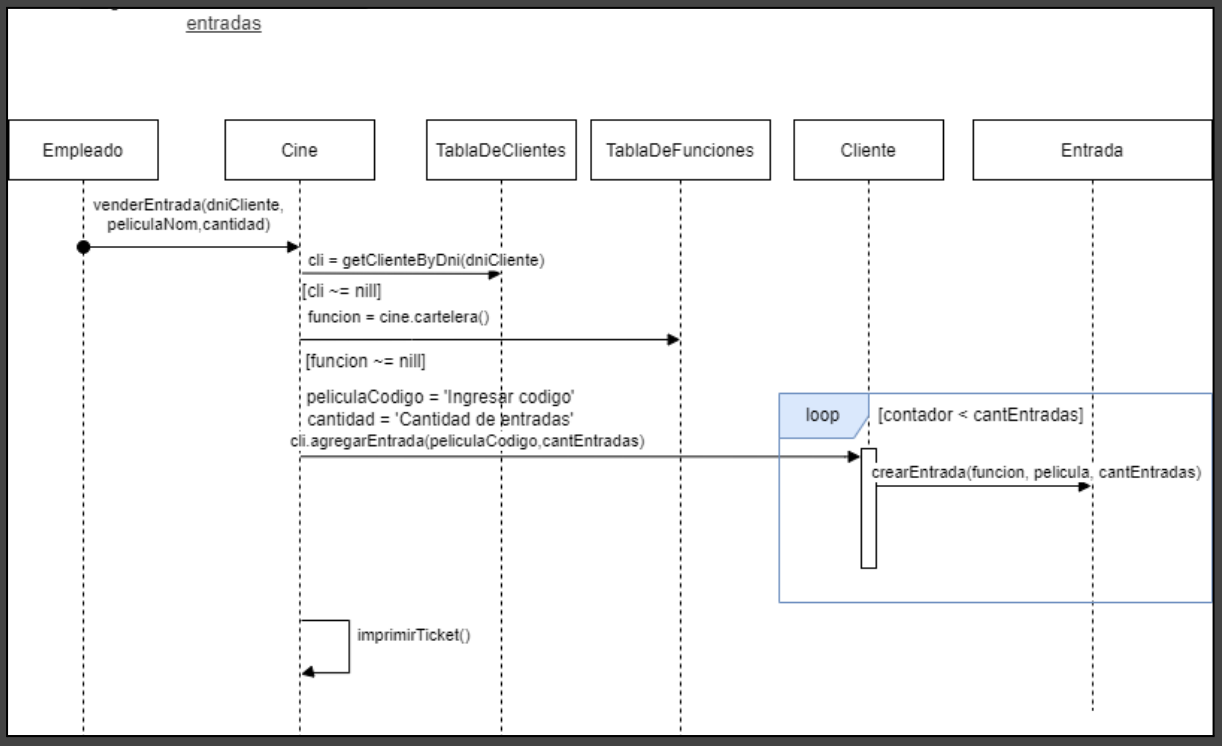


El cine es la clase principal del proyecto. Esta puede llamar a la clase Función y a la clase Película. Dentro almacena los diferentes clientes, películas, empleados como también otros datos de la misma entidad.

La clase Cine también puede llamar a la clase Persona que tiene como subclases a Empleado y Cliente, para la creación de las mencionadas entidades.

Las clases Empleado y Cliente conservan atributos de la clase superior (Persona) pero cada uno posee luego diferentes propiedades características. Cliente guarda las entradas creadas y puede crear nuevas llamando a la clase Entrada.

Diagrama de Secuencia



*El diagrama muestra las diferentes interacciones entre objetos del sistema de Cine para concluir con la venta de entrada/s.

1) Un empleado busca un cliente por su DNI (que es un identificador único y en caso de no encontrarlo se le permite registrarse)

2) Se obtiene la cartelera desde el cine.

3) Se pide que se ingrese el ID de la película elegida y la cantidad de entradas.

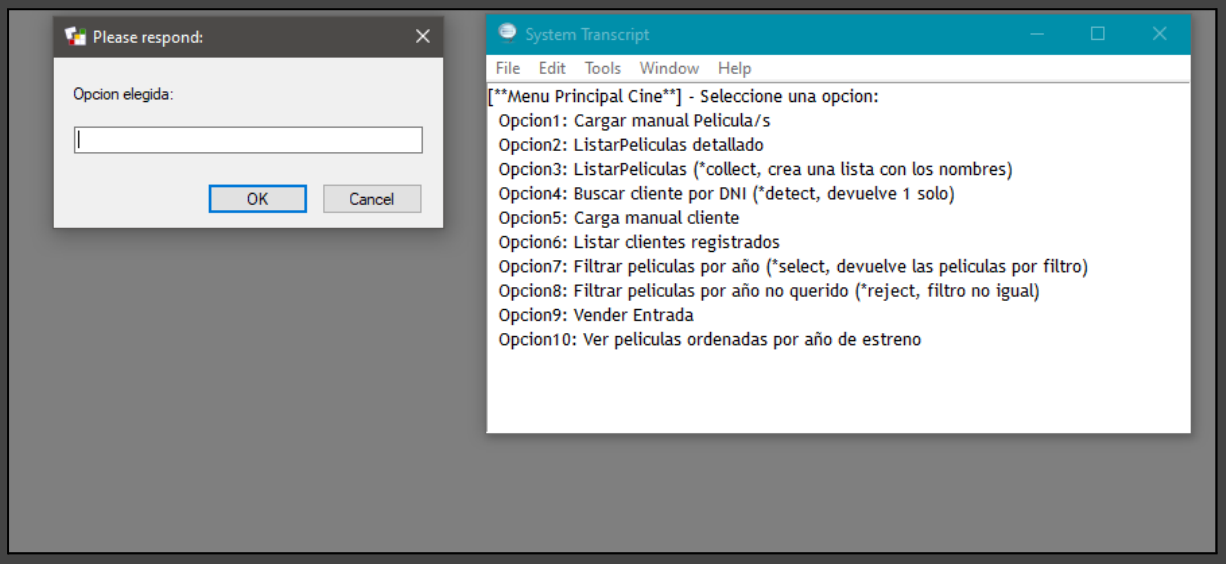
4) Se agrega la/s entrada/s a la lista de entradas del cliente. Se usa un loop interno durante la compra porque el cliente puede comprar una o varias entradas.

5) Se imprime un ticket para demostrar que se realizó la compra exitosamente.

Implementación de la aplicación:

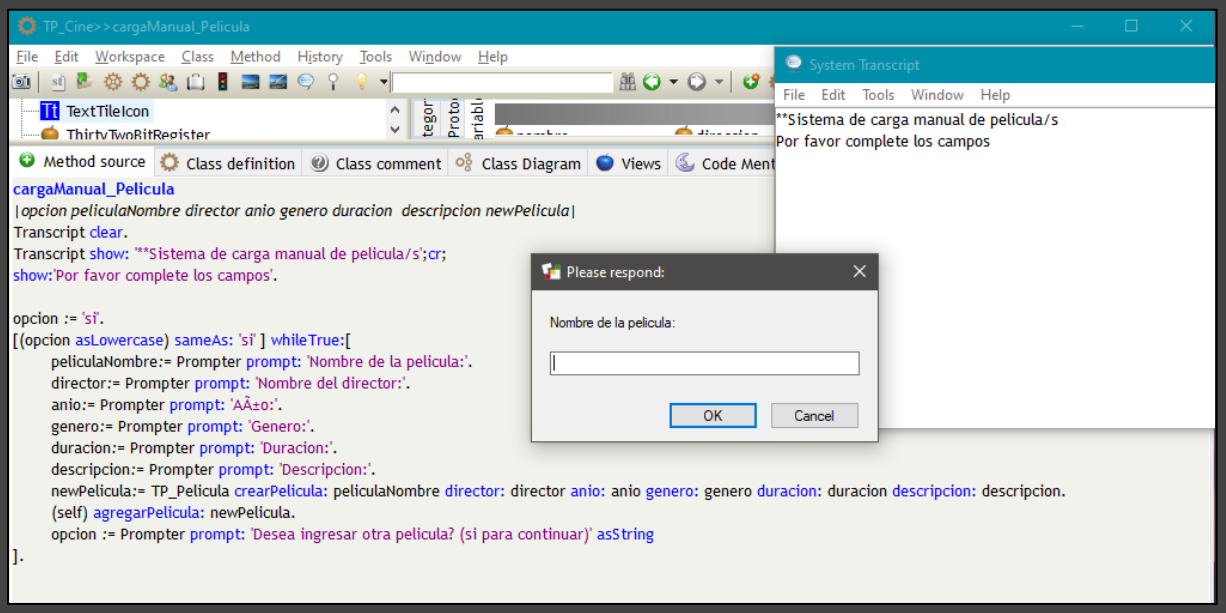
Se mostrará cómo se interactúa con el sistema y cómo se realiza el diagrama de secuencia demostrado antes de forma visual.

- En una primera instancia se muestra un menú con 10 opciones en donde el empleado puede realizar diferentes acciones. En todas las opciones se da una opción para volver al menú principal.

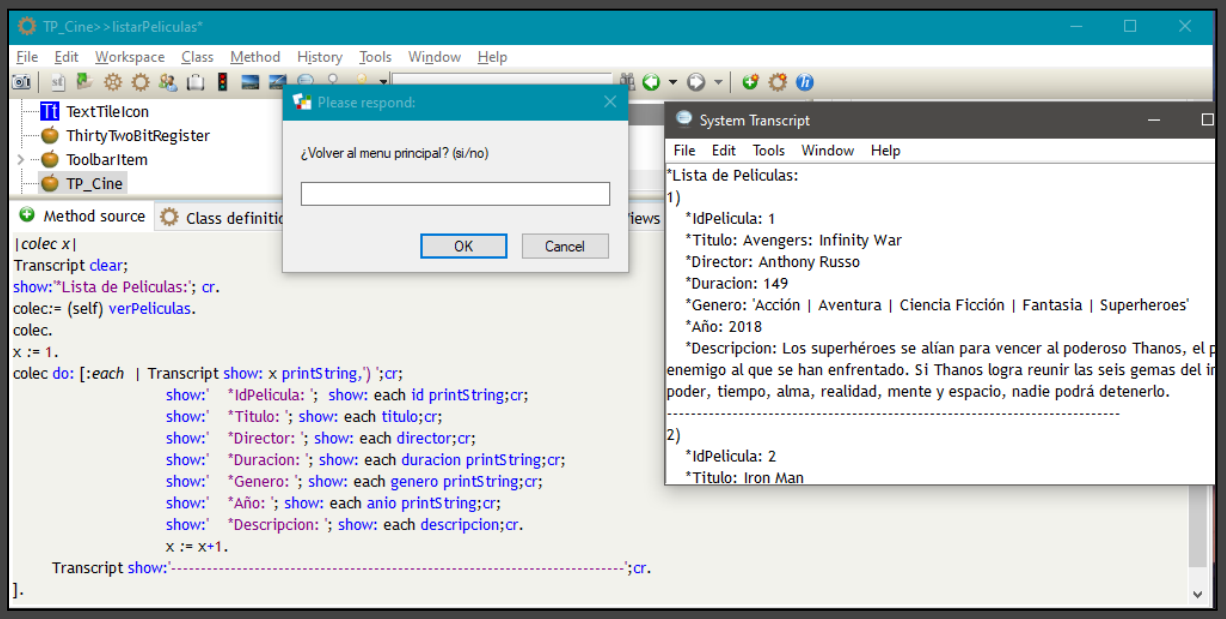


Opciones:

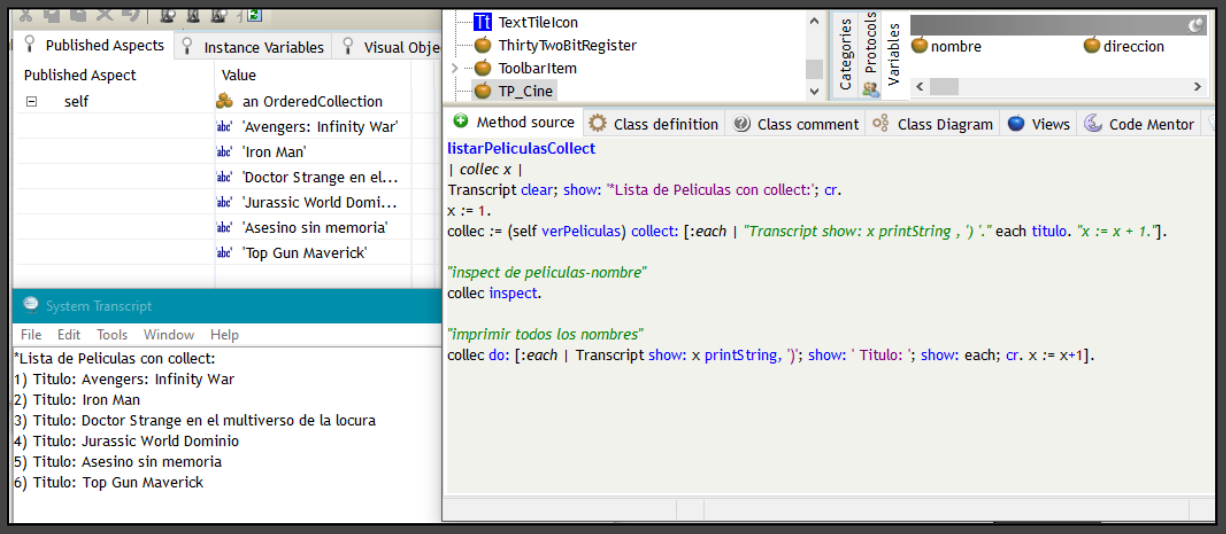
1) Carga manual de película: En donde se puede cargar una o varias películas.



2) Listar Películas detallado: Muestra las películas registradas con todos los datos de las mismas.

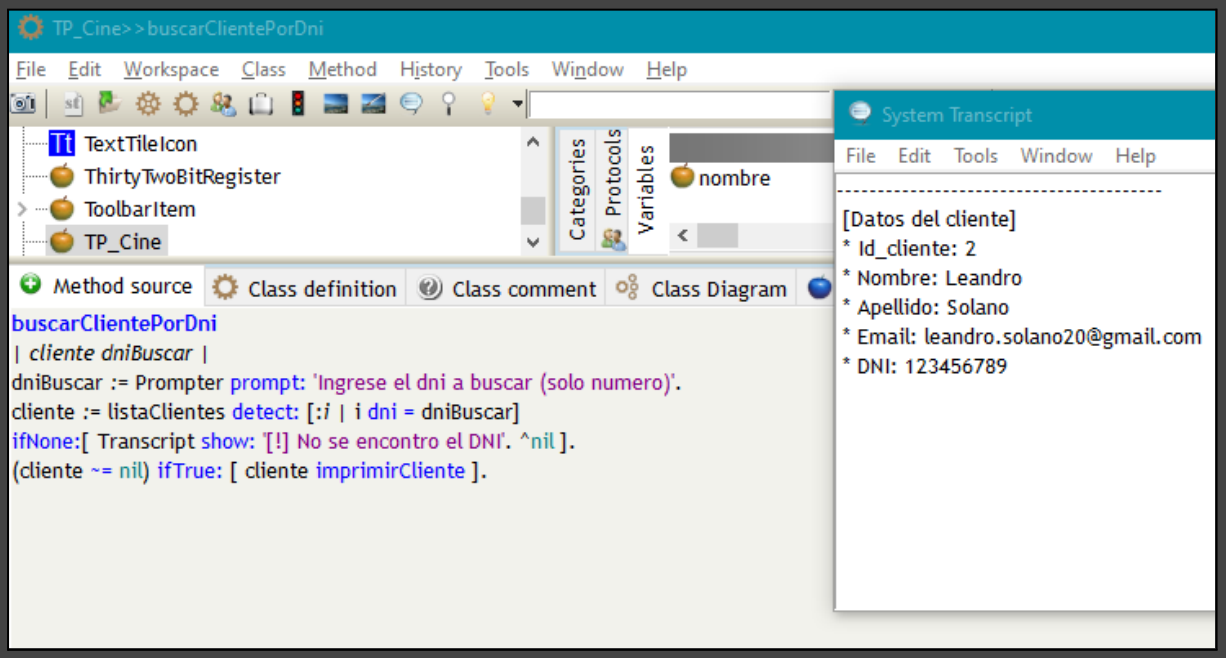


3) Lista de películas: Muestra SOLO los nombres de las películas registradas (se utiliza un *collect* para esta acción).



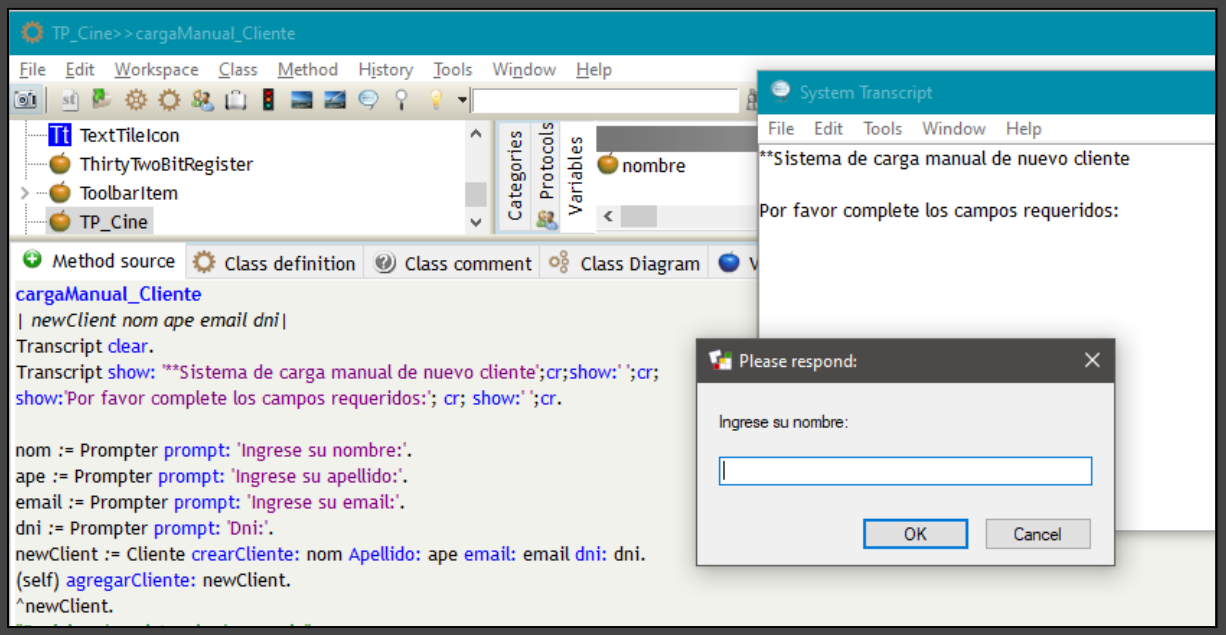
*En la parte derecha hay una lista con los nombres de las películas, como también se muestran en el Transcript.

4) Buscar cliente por DNI: Busca en la lista interna de clientes por DNI, si lo encuentra muestra los datos del mismo.



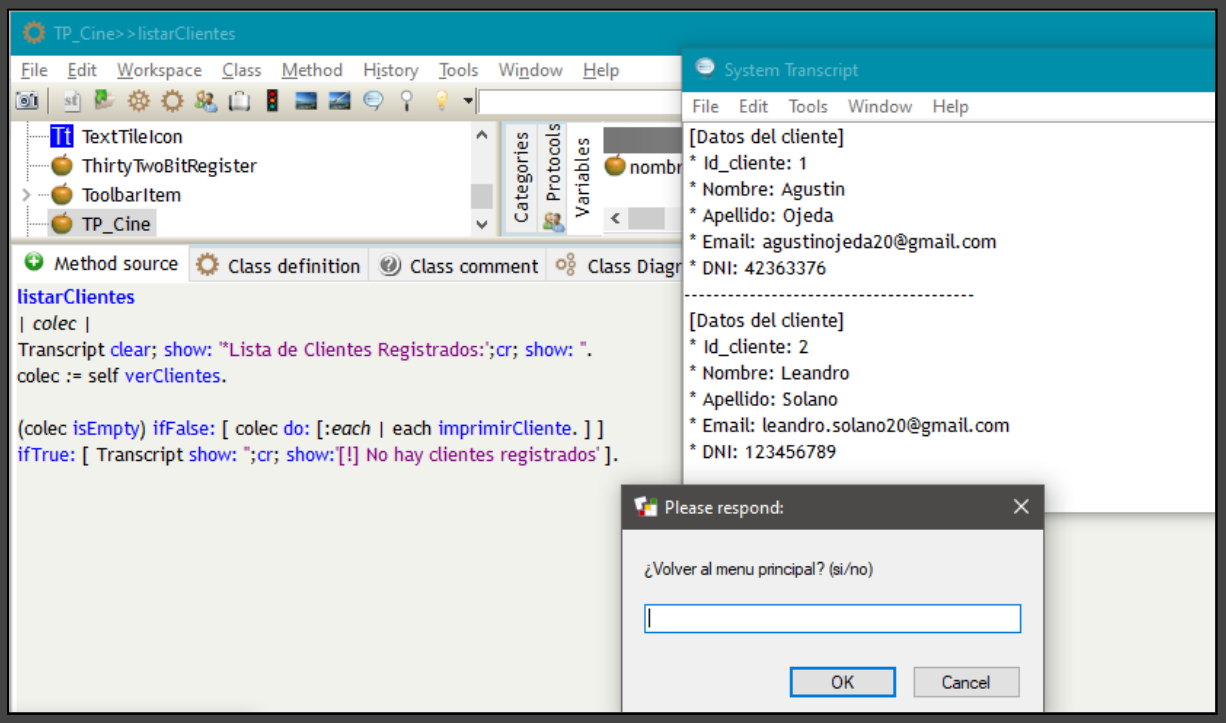
*En este caso sí encontró un cliente con el DNI 123456789 e imprimió sus datos llamando a una función ImprimirCliente.

5) Carga manual cliente: Se pide completar diferentes campos para poder realizar el registro.



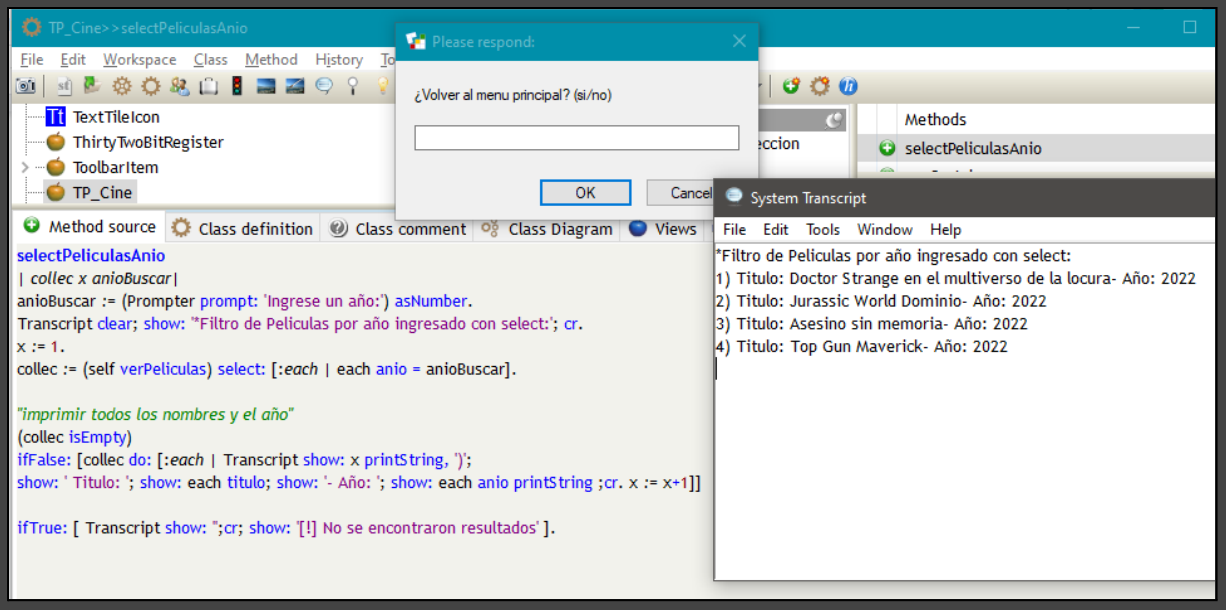
*Una vez ingresados los datos, se guarda al cliente recien registrado en la lista de clientes del cine.

6) Listar clientes registrados: Muestra los clientes registrados con sus datos guardados.



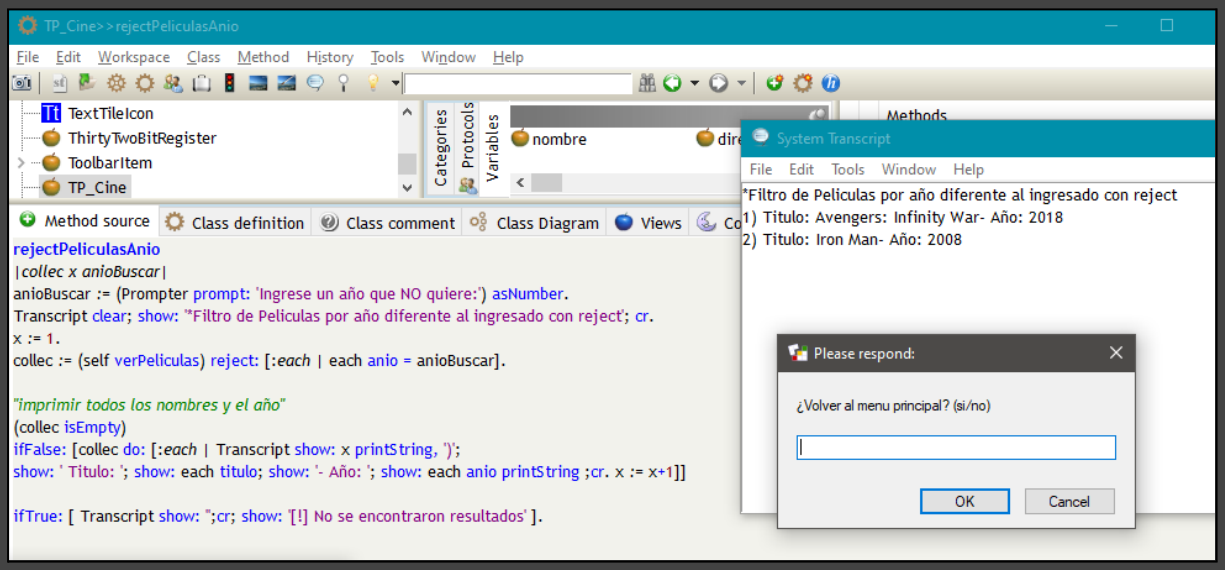
*En caso de que no hubieran clientes registrados se mostraría un mensaje “No hay clientes registrados”.

7) Filtrar película por año: Se debe ingresar un año y se muestra todas las películas que tengan como año de estreno este año. (Esta acción es realizada con un *select*).



*Si no hubieron resultados se muestra el correspondiente mensaje.

8) Filtrar película por año no querido: Muestra las películas que NO pertenezcan al año ingresado (Todo lo contrario a la opción anterior, se usa un *reject*).

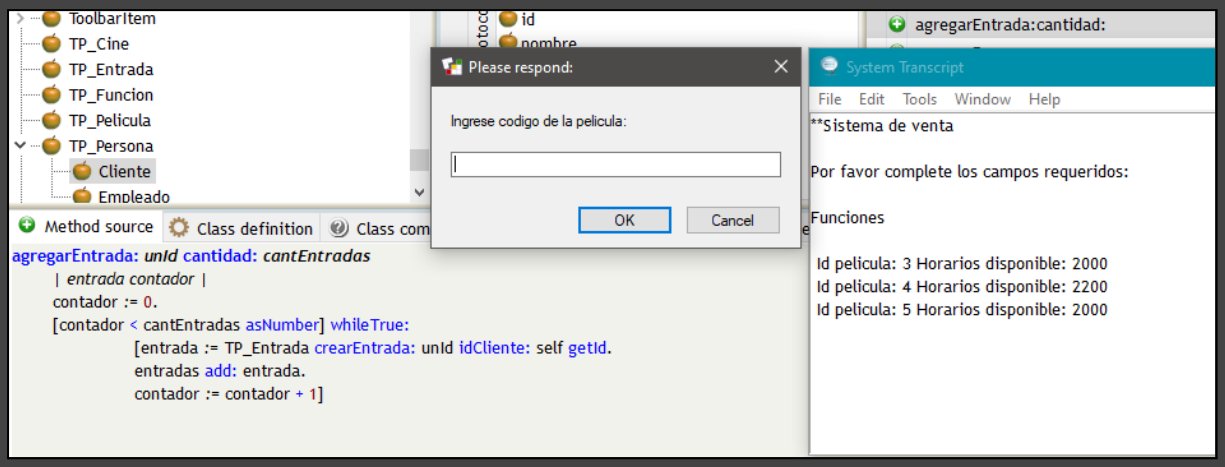


*Si no hubieron resultados se muestra el correspondiente mensaje.

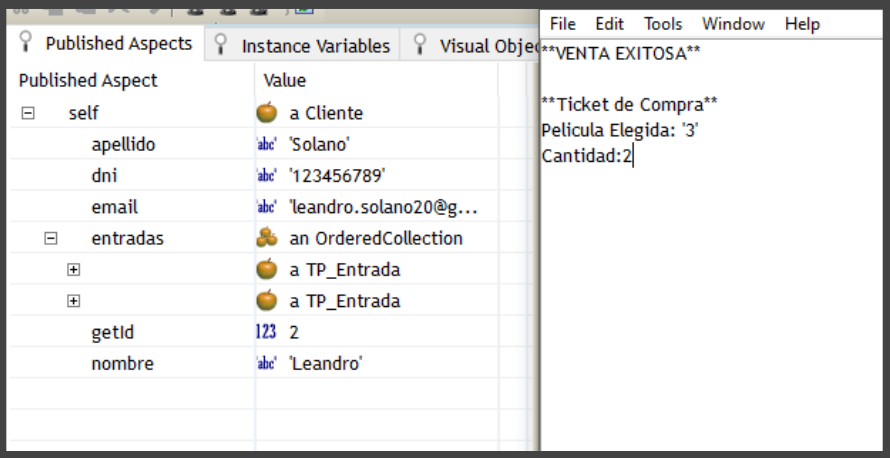
9) Vender Entrada: El usuario se identifica, en caso de que no esté registrado se le permite registrarse. Elige una película por el ID y la cantidad de entradas para la película.

```
(opcionElegida = '9') ifTrue: [
                    Transcript clear.
                    Transcript show: '**Sistema de venta';cr;show:' ';cr;
                    show:'Por favor complete los campos requeridos:'; cr; show:' ';cr.
                    listaClientes := cine verClientes.
                    x := 1.
                    "BUSQUEDA CLIENTE POR DNI, SI ES NULL CREA UNO NUEVO"
                    dniBuscar := Prompter prompt: 'Ingrese el dni a buscar (solo numero)'.
                    cliente := listaClientes detect: [:i | i dni = dniBuscar]
                    ifNone:[ Transcript show: '[!] No se encontro el DNI'. cine cargaManual_Cliente].

                    "Seleccionar una pelicula"
                    Transcript show:'Funciones';cr.
                    funciones := cine cartelera.
                    dct := Dictionary new.
                    (funciones isEmpty) ifFalse: [ funciones do: [:each | dct at: each peliculaid put: each horario ] ].

                    dct keysDo:[:key| Transcript cr.
                                      Transcript nextPutAll: ' Id pelicula: '.
                                      key printOn: Transcript.
                                      Transcript nextPutAll: ' Horarios disponible: '.
                                      (dct at:key) printOn: Transcript].

                    peliculaCodigo := Prompter prompt: 'Ingrese codigo de la pelicula:'.
                    cantidad := Prompter prompt: 'Cantidad de entradas:'.
                    cliente agregarEntrada: peliculaCodigo cantidad:cantidad.
                    Transcript clear.
                    Transcript show: '**VENTA EXITOSA**'.
                    cliente inspect.
```

*Se verifica la existencia del cliente desde el menú (Paso 1 Diagrama de Secuencia).

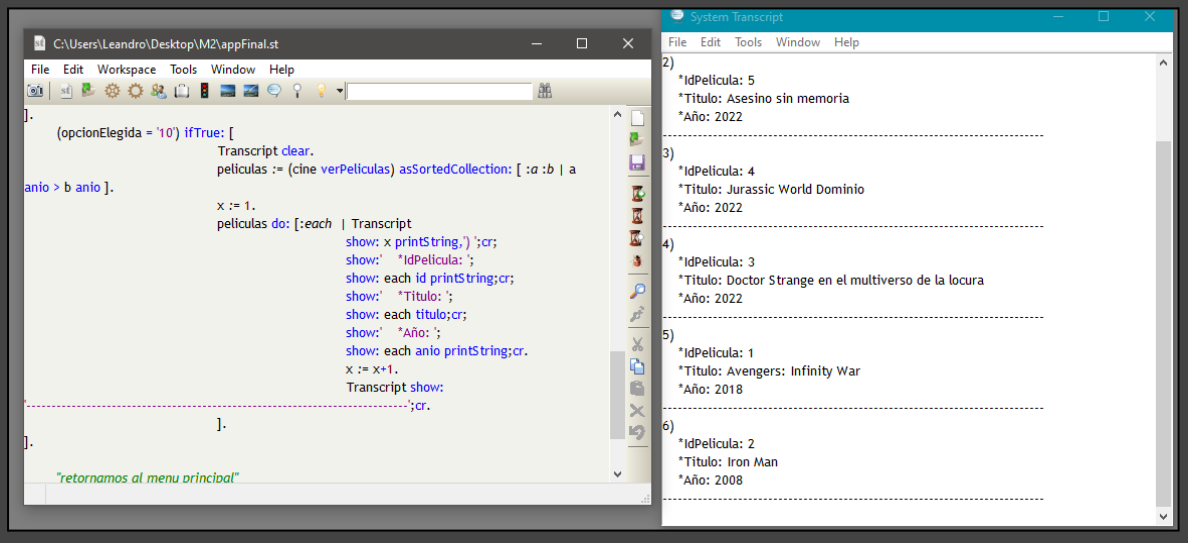


*(Paso 2 y 3 Diagrama de Secuencia).



*Inspeccionando al Cliente podemos ver que las 2 entradas elegidas fueron creadas y están presentes en la lista de entradas (Paso 4 y 5 Diagrama de Secuencia).

10) Ver películas por año de estreno: Muestra las películas ordenándolas por su año de estreno.



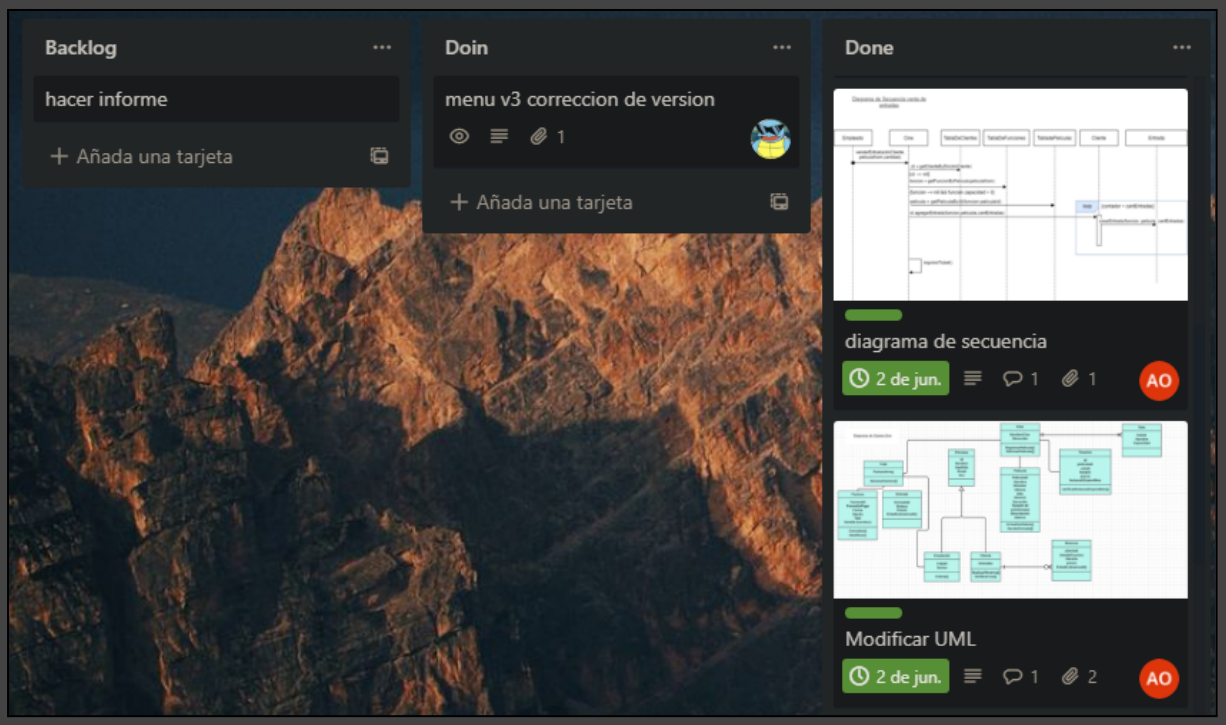
Metodología de trabajo

Se utilizó Trello (Software de administración de proyectos con interfaz web, https://trello.com/es) para la asignación de tareas y ver el estado de las actividades. Se tenía un tablero con 3 secciones:

- Backlog: Tareas por hacer.
- Doing: Tareas que se están en progreso.
- Done: Tareas ya finalizadas.

Cada vez que se terminaba una actividad se mandaba un mensaje al grupo aclarando que se hizo, y se subían los cambios a Github (Repositorio remoto de desarrollo, https://github.com/).

- Repositorio del Proyecto: https://github.com/leizaLS/Metodologia2/tree/main



*Tablero parecido al utilizado en Metodologías Ágiles Scrumban y Kanban.

Metodología Ágil aplicada

Se implementó la metodología *Scrum* durante el desarrollo, las daily con el cliente (Claudia Cappelletti) eran cada martes regularmente a las 18:00 para mostrar los avances y requisitos completados del proyecto. En estos encuentros también se determinaron los requisitos próximos que debian ser presentados para la semana siguiente. No hubo un *Scrum Master*, ambos miembros del equipo trabajamos de igual manera cada uno aportando en un sector del proyecto.

Se tenía un *Producto Backlog* con los objetivos del cliente (Pautas del trabajo a entregar.pdf). Además se usó una tabla muy parecida a la usada en Kanban y Scrumban, con el estado de tareas.

Posibles mejoras

Como en todo software es siempre posible mejorar. Cambiar de entorno de desarrollo es importante, de esta manera podemos brindar un mejor servicio y estar actualizado a nivel software.

Algunos ejemplos de frameworks:

- .Net
  Es Framework de Microsoft y uno de los más utilizados. Permite el desarrollo de diferentes tipos de proyectos de software. Ya hemos cursado materias durante el transcurso de la carrera que usan este framework, sería fácil de adaptar.
  https://dotnet.microsoft.com/en-us/

- Express
  Es el framework web más popular de Node Js usado para crear Aplicaciones Web. Teniendo ya desarrollado el “backend” es posible crear el “frontend” del proyecto. Express y Node Js se utilizan en la materia “Proyecto Software”.
  https://expressjs.com/es/

Patrones de diseño:

Algunos patrones que podrían utilizarse en el proyecto son:

- Factory Method: Para crear objeto como Clientes, Empleados, Películas, entre otros, a pedido
- Singleton: Para solo tener una instancia de Cine.
- Iterator: Para acceder a los diferentes datos, como la lista de Películas o lista de Clientes registrados.
- Decorator: En el sistema de ventas, dependiendo de los extras del servicio como combo de pochoclos o otros productos, se incremente el precio del servicio brindado.

Conclusión:

Se puso en práctica el aprendizaje adquirido durante la cursada, se pudo llegar a un entregable funcional desarrollado en Smalltalk que cumple con los requisitos dados por el cliente y profesora de Metodologías de Programación II (Claudia Cappelletti).

El uso de una metodología ágil de trabajo fue bastante útil para organizar el desarrollo en equipo, como se había mostrado en clase y fue una primera experiencia a la forma de trabajo real de un programador.

Concluyendo, el resultado del trabajo finalizado es fruto de la puesta en práctica de la combinación de diferentes aspectos vistos durante la materia misma.