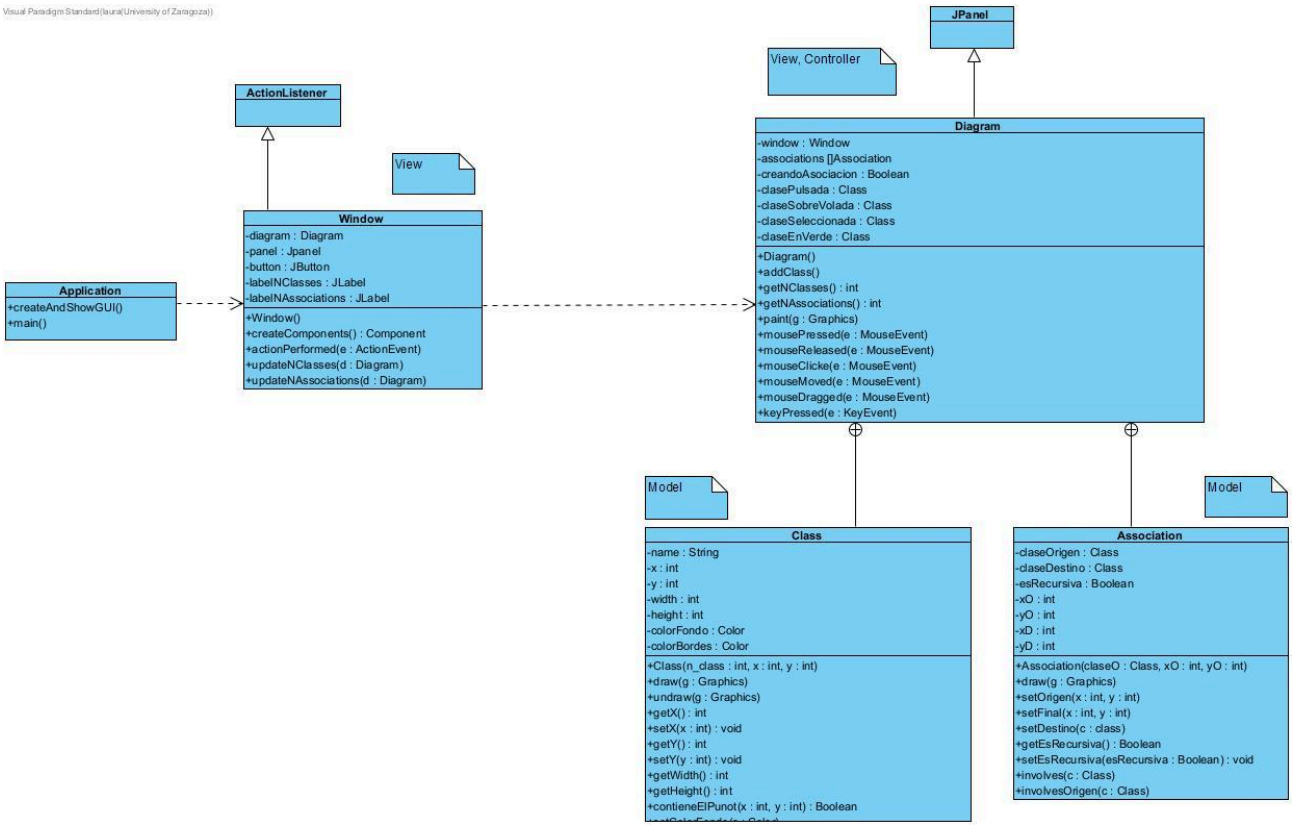
PRÁCTICA 2

ARQUITECTURA SOFTWARE:

MODELO, VISTA Y CONTROLADOR

Jorge Hernández Aznar 872838
Laura Hernández Sierra 872203
Arquitectura Software 2024/2025
17/03/2025

1. Diagrama de Clases



El patrón Modelo-Vista-Controlador en nuestra implementación tiene variaciones con respecto a la propuesta clásica. En el enfoque estudiado en clase, el Modelo se encarga exclusivamente de la lógica de negocio y la gestión de datos, la Vista es responsable de la representación gráfica sin lógica interna, y el Controlador actúa como intermediario, capturando la entrada del usuario y comunicándose con el Modelo y la Vista para actualizar el estado de la aplicación.

Sin embargo, en nuestra implementación, esta división no es tan concreta, ya que ciertas responsabilidades se encuentran más entremezcladas.

Por un lado, el Modelo está representado por las clases Class y Association, que se encargan de manejar los datos. La clase Class guarda la información de cada clase del diagrama, como su nombre, atributos y métodos, además de su posición y tamaño en la interfaz, mientras que Association gestiona las relaciones entre las clases, conectando distintas instancias de Class. No obstante, estas clases no actúan como lo haría un Modelo clásico, ya que no emplean un patrón Observer para notificar a la Vista los cambios. En su lugar, es el Controlador el que se encarga de consultar el estado del Modelo al actualizar la vista cuando suceden cambios.

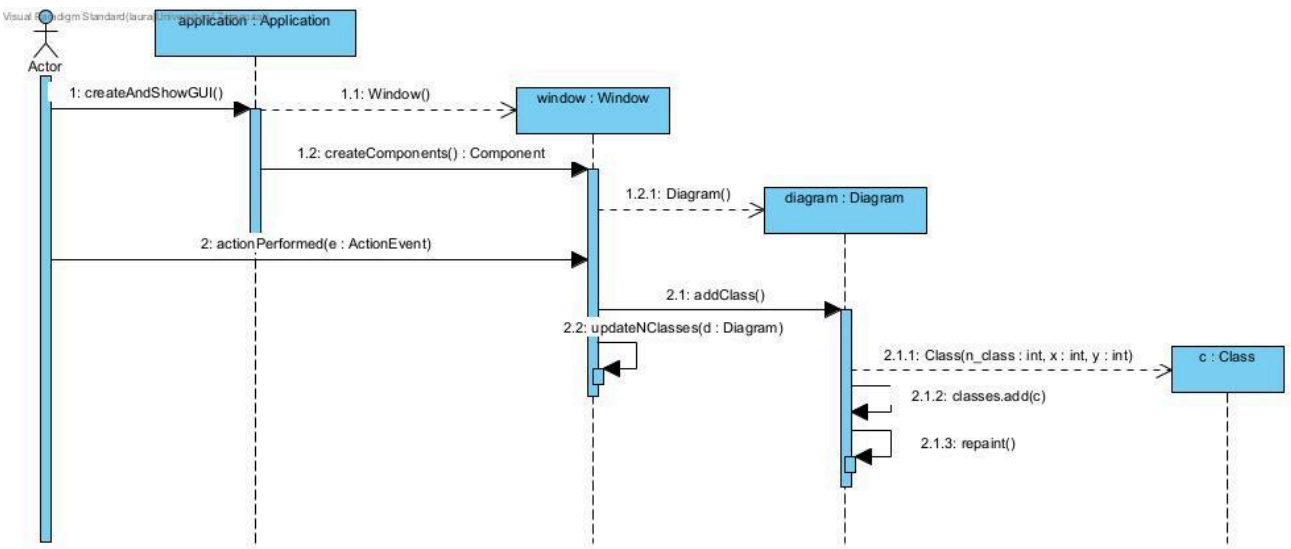
La función de Controlador, en nuestro caso, es asumida por la clase Diagram. Esta clase recibe los eventos del usuario, como clics y teclas presionadas, y transforma esas acciones en cambios en el Modelo. Por ejemplo, cuando el usuario selecciona una clase o agrega una nueva, Diagram se encarga de gestionar esa acción y luego actualizar la Vista para reflejar el cambio. En este enfoque, evitamos la necesidad de una clase específica de Controlador, ya que Diagram cumple con ese papel de manera integrada.

En cuanto a la Vista, las clases Window y Diagram son las responsables de mostrar la información en pantalla. Window se ocupa de la interfaz principal y de actualizar algunos elementos cuando el número de clases o asociaciones cambia. Diagram, además de gestionar eventos, se encarga de dibujar los elementos visuales, mostrando los datos almacenados en el Modelo. Aunque en el MVC clásico la Vista solo debería mostrar información sin modificarla, en nuestra versión cumple un papel más dinámico, ya que también responde a las acciones del usuario.

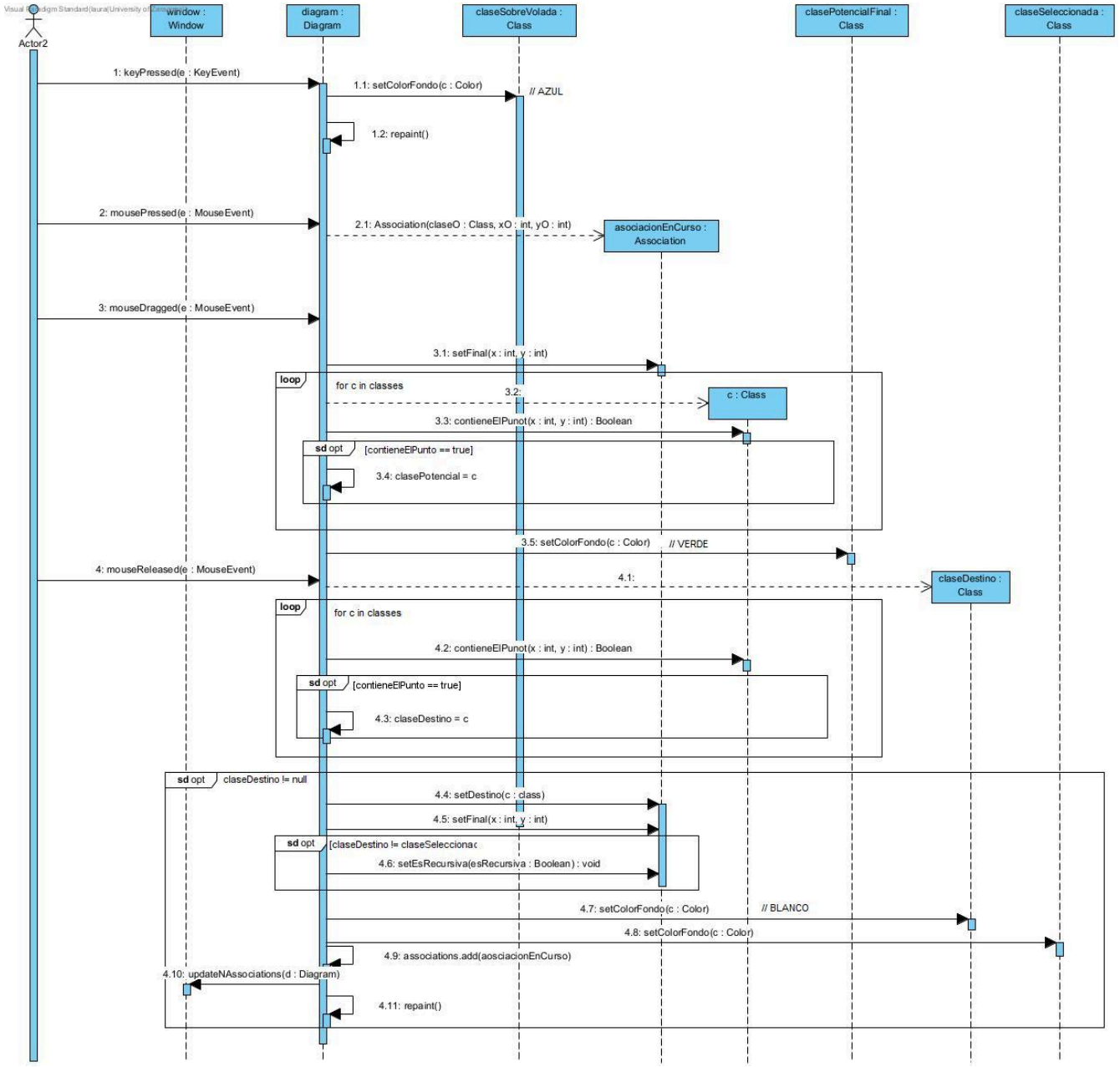
En conclusión, nuestro enfoque del patrón MVC no sigue una separación rígida de funciones, sino que adapta los roles según las necesidades del programa. Class y Association gestionan los datos como parte del Modelo, Diagram cumple un doble rol como Vista y Controlador, procesando las acciones del usuario y actualizando la representación gráfica de la información. Window actúa también como vista al encargarse de mostrar la interfaz, y Application simplemente inicia el programa. Gracias a esta organización, conseguimos que nuestra aplicación funcione de manera fluida sin necesidad de una separación estricta entre los componentes.

2. Diagramas de Secuencia

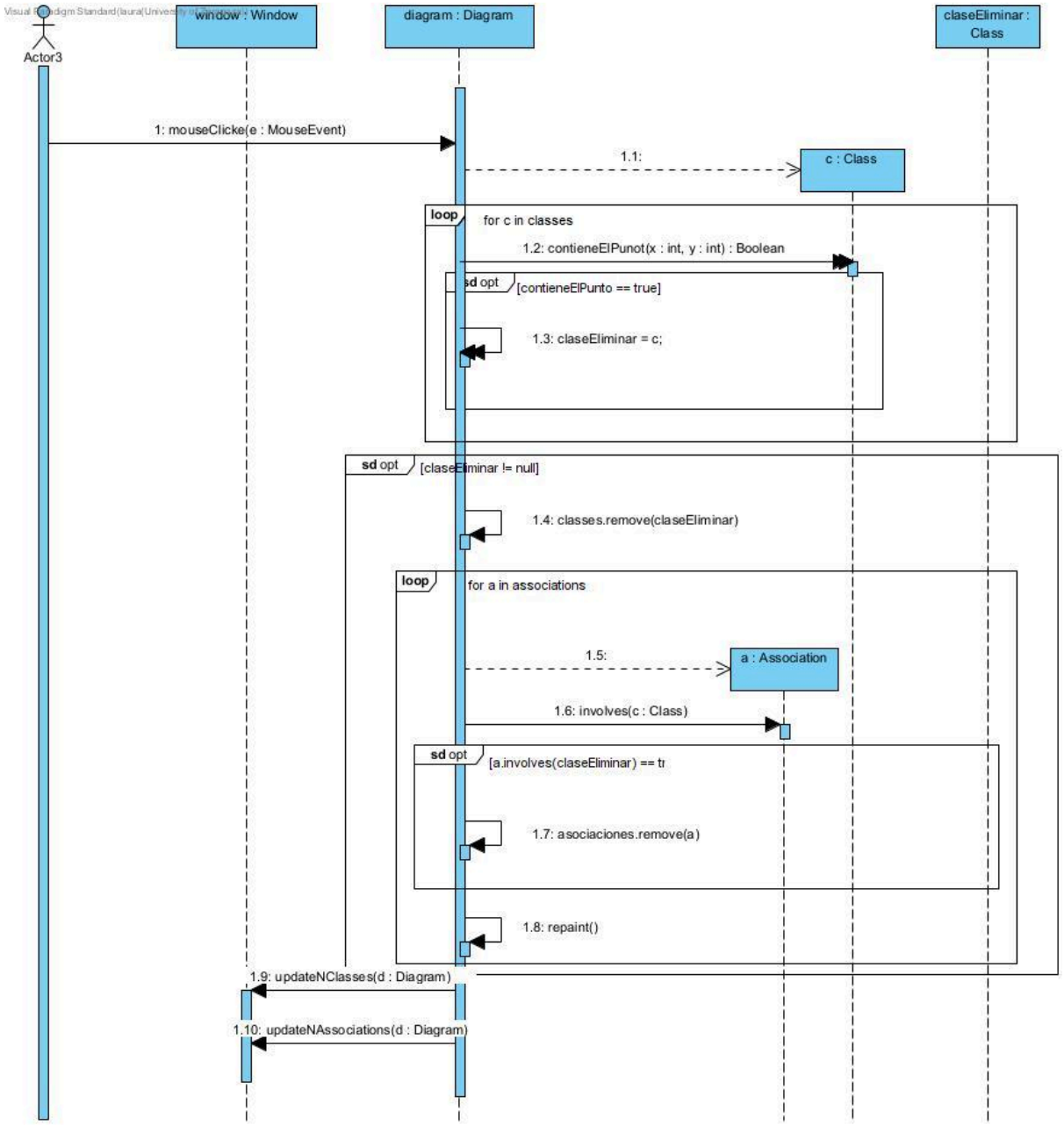
2.1. Añadir Clase



2.2. Añadir Asociación



2.3. Borrar Clase



2.3. Mover Clase