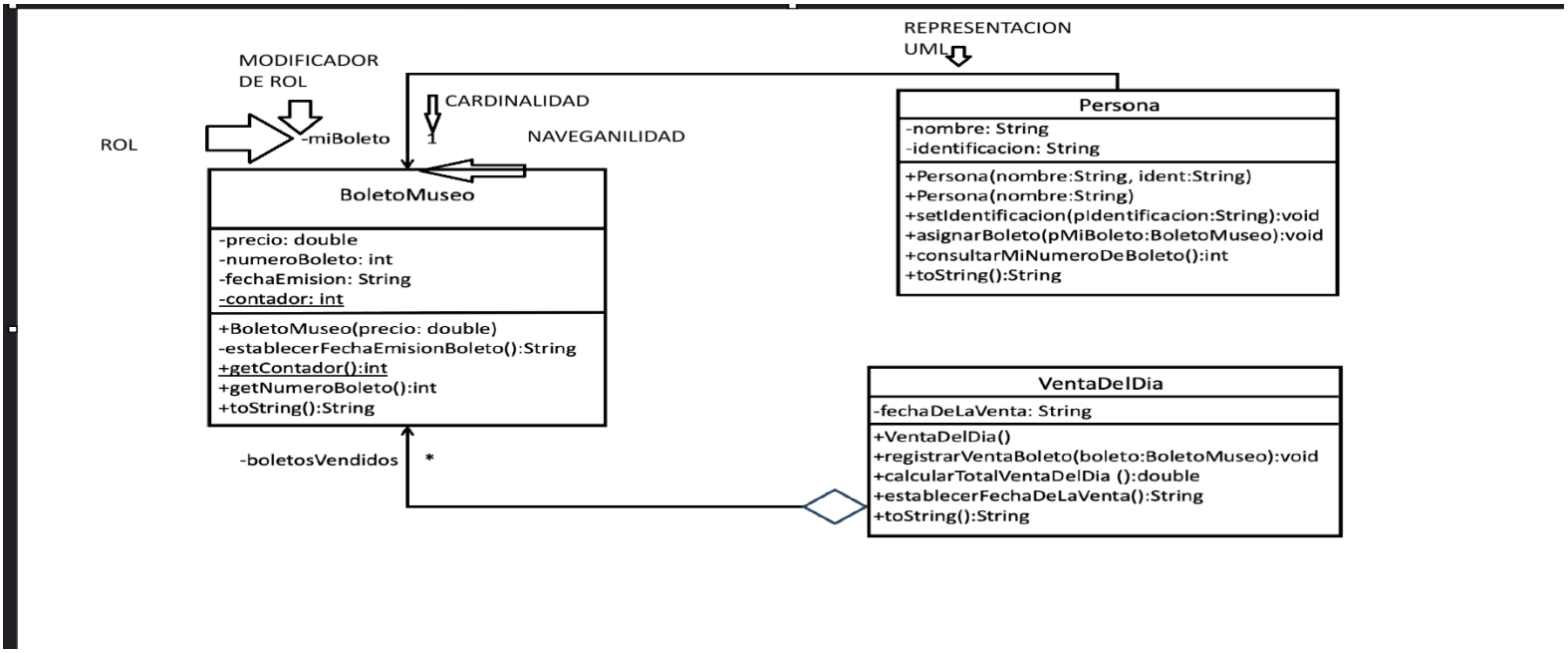
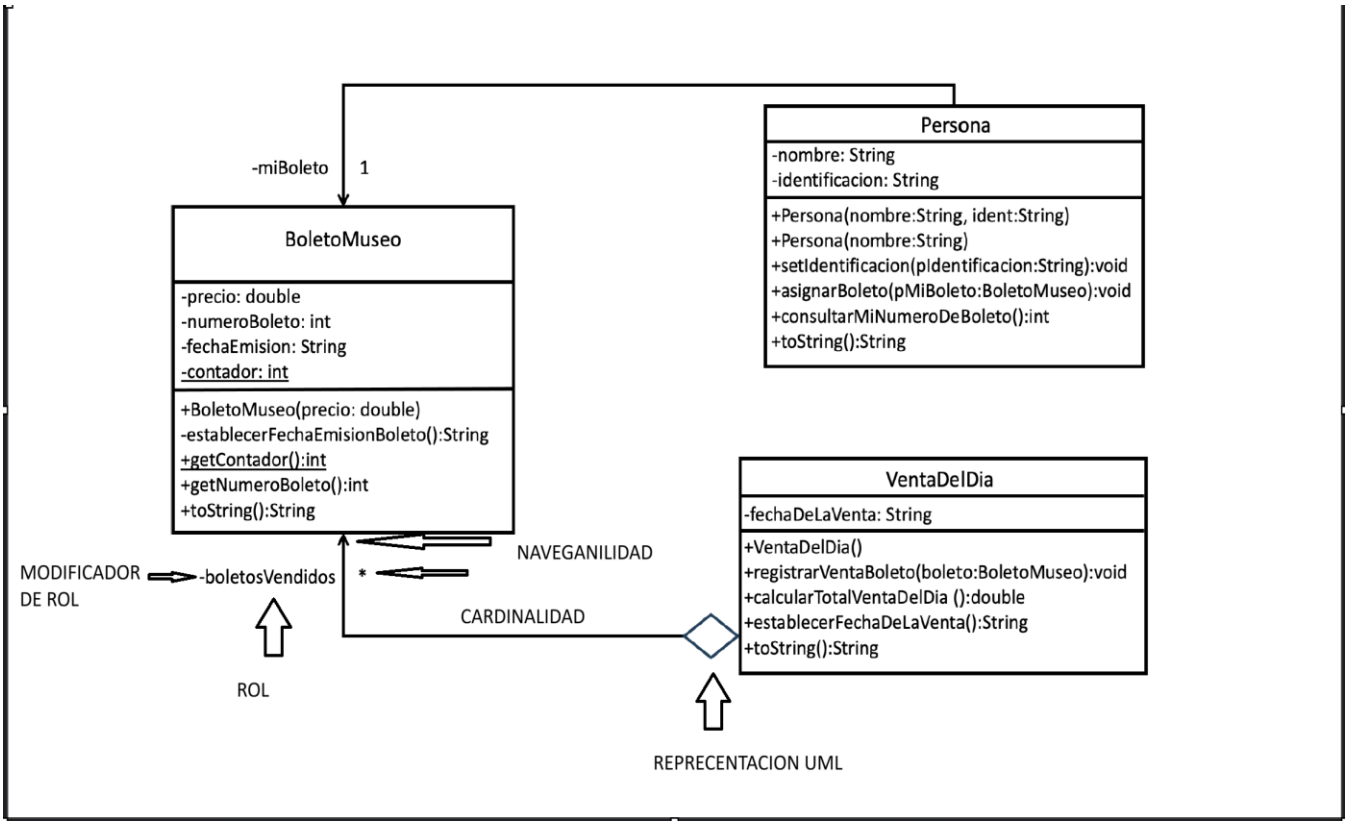
```
Detalle del primer objeto Persona: Persona
 Nombre: Nicolás Maduro
 Identificacion: 666-6
 Boleto asignado: #1

Detalle del segundo objeto Persona: Persona
 Nombre: Donald Trump
 Identificacion: 333-3
 Boleto asignado: #2

Detalle del tercer objeto Persona: Persona
 Nombre: Claudia Sheinbaum
 Identificacion: 777-7
 Boleto asignado: #3

Contador global de boletos creados: 3
Detalle de la Venta Del Día: VentaDelDia
  Fecha: 2025-09-26
  Cantidad de boletos: 3
  Detalle:
    - Boleto #1 | 4500.0
    - Boleto #2 | 6000.0
    - Boleto #3 | 5800.0
  Total: 16300.0
```

ASOCIACIÓN



AGREGACIÓN



Preguntas

A. Si la clase A está vinculada con la clase B mediante una relación de asociación. ¿La estructura de la clase B se ve impactada? Explique con detalle.

Respuesta:

No necesariamente. Una asociación implica que la clase A conoce a la clase B, pero eso no obliga a que la clase B conozca a la clase A. Por lo tanto, la estructura de B no se modifica, salvo que el diseño requiera una relación bidireccional. En otras palabras, solo A almacena una referencia a B, pero B no se ve afectada en su definición interna.

B. Si la clase P está vinculada con la clase Q mediante una relación de agregación. ¿La estructura de la clase Q se ve impactada? Explique con detalle.

Respuesta:

No, la clase Q (la parte) no se ve impactada por la agregación. La agregación establece que la clase P (el todo) contiene o agrupa objetos de Q, pero Q puede existir por sí misma de manera independiente. Su estructura no depende de P, lo que diferencia la agregación de la composición.

C. Si la clase A está vinculada con la clase B mediante una relación de asociación y la clase B está vinculada con la clase A mediante una relación de asociación. ¿La estructura de ambas clases se ve impactada? Explique con detalle.

Respuesta:

Sí. En este caso, hablamos de una asociación bidireccional, lo cual significa que ambas clases deben tener referencias mutuas. Eso obliga a modificar la estructura de las dos clases para incluir atributos que representen el vínculo con la otra.

D. ¿Un objeto de tipo Z podría enviar mensajes a otro objeto de tipo W, aun cuando no exista un vínculo (de asociación o agregación) entre la clase Z y la clase W? Explique con detalle.

Respuesta:

En principio, no. Para que un objeto Z pueda enviar mensajes a un objeto W, debe existir algún mecanismo de acceso: una relación estructural directa (asociación/agregación), herencia o bien la participación de otra clase intermediaria que provea la referencia. Sin un vínculo en el diseño, Z no tendría forma de conocer la existencia de W.

E. En un diagrama de clase con detalles de implementación, suponga que existe una relación de asociación entre la clase P y la clase Q. Suponga también que esa relación tiene los cinco elementos respectivos en el diagrama. Es decir, la relación presenta todo el detalle de implementación posible. ¿Eso es suficiente para establecer de forma completa el vínculo de asociación entre P y Q? Explique con detalle.

Respuesta:

No, no es suficiente solo con el diagrama. Aunque el diagrama UML con sus cinco elementos (representación, navegabilidad, rol, modificador de acceso y cardinalidad) describe el vínculo, este debe materializarse en el código. Es decir, debe existir un atributo en la clase correspondiente que guarde la referencia, y métodos que permitan interactuar. El diagrama es una guía, pero el verdadero vínculo se concreta al implementarlo en el programa.

Reflexión

Durante el desarrollo de esta actividad asincrónica pude comprender mejor la importancia de las relaciones entre objetos en la Programación Orientada a Objetos. Al inicio, me parecía que trabajar con clases aisladas era suficiente, pero al aplicar asociación y agregación entendí que la verdadera potencia del paradigma está en cómo los objetos colaboran entre sí para formar soluciones más completas.

También aprendí que un buen diseño no se limita al código: el diagrama UML con detalles de implementación es fundamental porque me permite visualizar la estructura, la cardinalidad y la navegabilidad de las relaciones antes de programar. Esto me ayudó a comprender mejor cómo implementar las conexiones entre las clases Persona, BoletoMuseo y VentaDelDia.

Otro aspecto valioso fue el uso de Javadoc. Documentar cada clase, atributo y método me obligó a pensar en la claridad de mi código y a escribirlo de forma que pueda ser entendido por otras personas en el futuro. Además, ver cómo se genera la documentación en HTML me dio una experiencia práctica cercana a cómo se documentan librerías reales en Java.

En conclusión, esta actividad me permitió reforzar no solo la parte técnica de POO y UML, sino también la importancia de la documentación y las buenas prácticas de programación. Siento que ahora tengo una base más sólida para enfrentar proyectos más grandes y para desarrollar software de manera organizada y profesional.