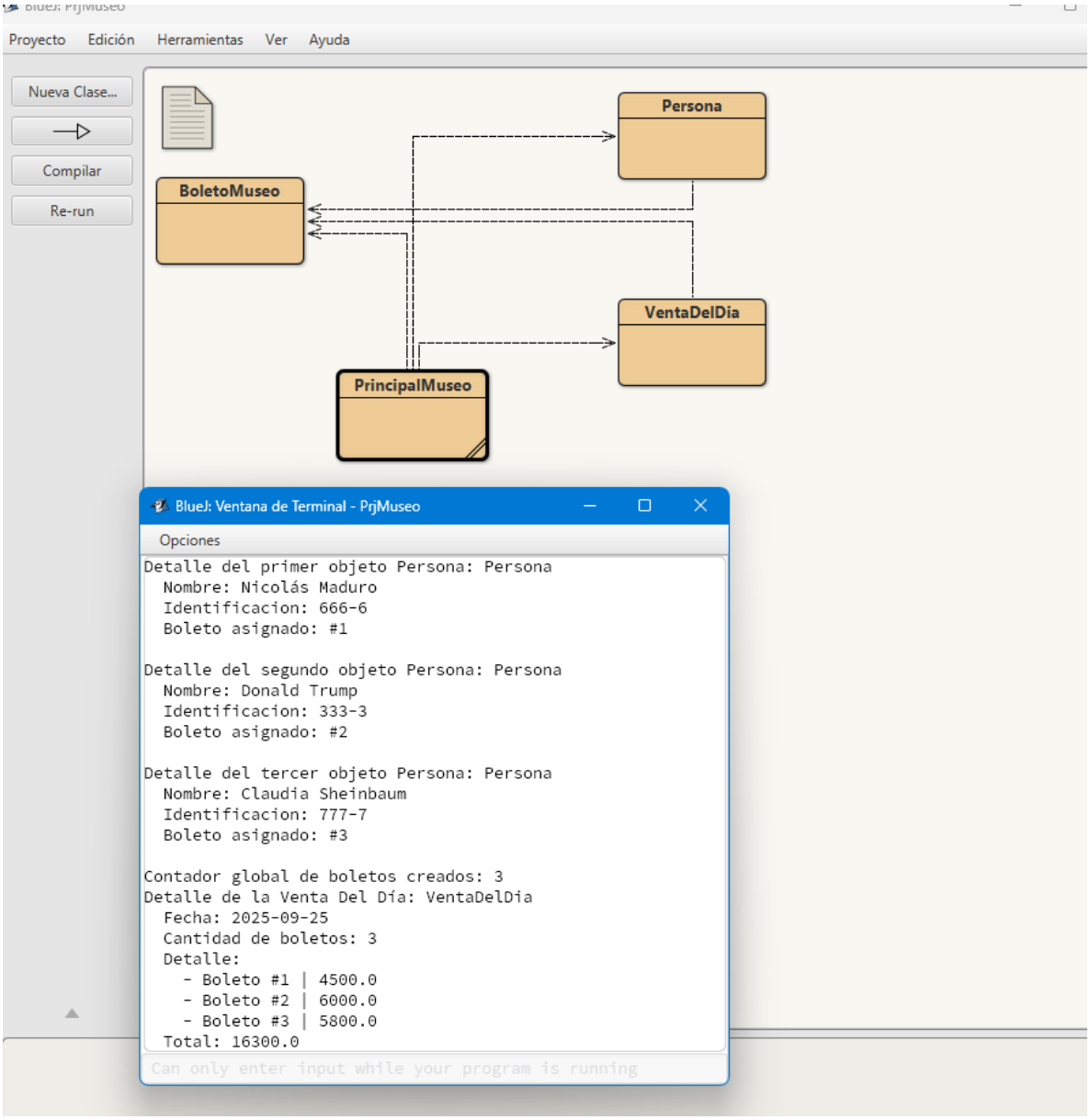
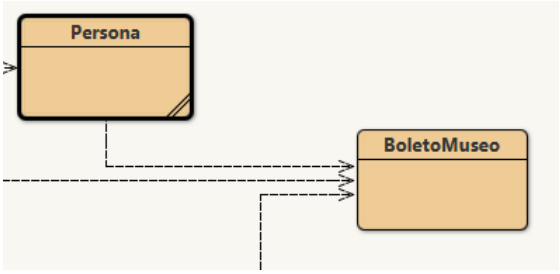
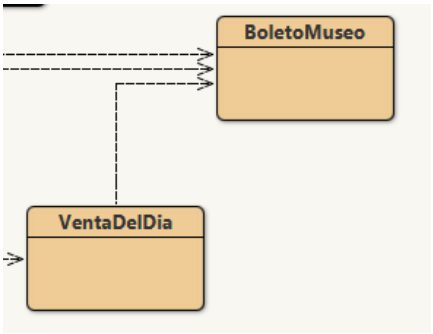
E.


Proyecto
Edición
Herramientas
Ver
Ayuda
Nueva Clase...
Compilar
Re-run
Persona
BoletoMuseo
VentaDelDia
PrincipalMuseo
BlueJ: Ventana de Terminal - PrjMuseo
Opciones
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 Dia: VentaDelDia
Fecha: 2025-09-25
Cantidad de boletos: 3
Detalle:
- Boleto #1 | 4500.0
- Boleto #2 | 6000.0
- Boleto #3 | 5800.0
Total: 16300.0
Can only enter input while your program is running

H.



1. Representación UML: línea.
2. Navegabilidad: flecha de Persona hacia BoletoMuseo.
3. Rol: miBoleto.
4. Modificador: privado (-).
5. Cardinalidad: 1.



1. Representación UML: rombo vacío en VentaDelDia
2. Navegabilidad: VentaDelDia hacia BoletoMuseo.
3. Rol: boletosVendidos.
4. Modificador: privado (-).
5. Cardinalidad: * (muchos).

J.

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.
No, la clase B no se ve afectada en su estructura. En una asociación, lo que ocurre es que la clase A mantiene una referencia de tipo B, lo cual le permite usar sus atributos o llamar a sus métodos. Sin embargo, la definición de B no cambia, sigue teniendo los mismos atributos y métodos sin alterarse. Lo único que sucede es que A “conoce” a B y puede interactuar con ella, mientras que B se mantiene totalmente independiente. En la práctica, esto significa que puedo tener muchas clases que se asocien a B sin necesidad de modificar la clase B cada vez.

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.
No, la clase Q no cambia su estructura porque la agregación lo que describe es la relación de “todo–parte”. P contiene o agrupa varios objetos de Q, pero Q sigue siendo autónoma. Es decir, Q se puede crear y usar de manera independiente, sin depender de P. Lo que cambia es que la clase P debe tener una colección o lista de objetos de tipo Q, con un rol y una cardinalidad definida. En mi proyecto, por ejemplo, VentaDelDia agrega varios BoletoMuseo, pero la clase BoletoMuseo no necesita modificarse para que esto funcione.

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.
Sí, en este caso sí se impactan las dos estructuras. Cada clase debe tener un atributo que apunte a la otra, lo cual genera una relación bidireccional. Eso significa que A guarda una referencia de B y, al mismo tiempo, B guarda una referencia de A. Esto modifica las dos clases porque ya no son independientes del todo: cada una necesita conocer a la otra. Este tipo de relación puede ser útil cuando se requiere comunicación constante en ambos sentidos, pero también se debe usar con cuidado porque puede generar acoplamiento fuerte entre las clases.

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.
No, un objeto de Z no puede comunicarse con un objeto de W si no existe una relación definida entre sus clases. Para que Z pueda enviar un mensaje, necesita tener una referencia de W, y esa referencia solo existe si en el diseño hay una asociación o agregación que lo permita. De lo contrario, Z no tendría forma de “ver” o “acceder” a W en memoria. Esto refleja un principio importante: la comunicación entre objetos en programación orientada a objetos siempre depende de las relaciones establecidas en el diseño UML.

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. ¿Eso es suficiente para establecer de forma completa el vínculo de asociación entre P y Q? Explique con detalle.
Sí, cuando la asociación incluye los cinco elementos (representación UML, navegabilidad, rol, modificador y cardinalidad), ya se considera completamente definida. Esto significa que cualquier persona que lea el diagrama sabe exactamente cómo se implementará la relación en el código. Por ejemplo, con esos datos se puede identificar qué clase conoce a la otra, cómo se llama el atributo que almacena la referencia, qué visibilidad tendrá ese atributo y cuántos objetos estarán involucrados. Con esta información ya no quedan dudas ni ambigüedades a la hora de escribir el programa, lo cual es el objetivo de trabajar con diagramas detallados.

L.

Con esta actividad aprendí de una forma más clara qué son las relaciones de asociación y agregación, algo que antes solo había visto de manera teórica. Al programar las clases y ver cómo se relacionaban entre sí, entendí mejor cómo se aplican en la práctica. También descubrí lo útil que es Javadoc para documentar el código, porque permite generar explicaciones automáticas que facilitan la lectura. Aunque todavía me falta experiencia, siento que este ejercicio me ayudó a comprender mejor cómo se conectan los objetos en un programa y por qué es importante representarlo en un diagrama UML.