UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Informática

LABORATORIO 3

INFORME PARADIGMA ORIENTADO A OBJETOS

Paradigmas de la Programación

Vladimir Vidal Luieza

Profesor: Victor Flores

Fecha: 19 de Agosto del 2024

Tabla de contenidos

I. Introducción

El reto que se aborda en el proyecto de este semestre consiste en el diseño y la implantación de un sistema de gestión para una red de metro simplificada, que afronte el problema de ofrecer soluciones de transporte eficientes y sostenibles en el ámbito de la movilidad urbana en un contexto de creciente demanda. A esto se le confiere la necesidad de emplear diversos paradigmas de programación para construir un sistema que sea capaz de afrontar operaciones críticas encaradas en una red de metro, como puede ser la creación y gestión de líneas y estaciones, la planificación de rutas y conexiones de líneas o la gestión de trenes y tarifas.

Dicho sistema debe hacer frente a los desafíos de la movilidad urbana garantizando el desplazamiento eficiente de pasajeros dentro de un entorno urbano metropolitano amplio y diverso. Se pone el acento en la disminución de la congestión del tráfico en la superficie y en la mejora de la accesibilidad y eficiencia del transporte público subterráneo, así como en la contribución a la reducción de las emisiones contaminantes durante toda la movilización. Los alumnos, a través del proyecto, han de demostrar su capacidad de integrar tecnología de señalización y control, monitorizar el comportamiento del sistema en tiempo real y garantizar la seguridad operativa y estructural del sistema de metro.

Dicho reto implica el hecho de que se han de cumplir unos mínimos requerimientos obligatorios y se ha de jugar con oportunidades para la innovación y la creatividad individual en la ejecución de soluciones específicas, a la vez que se prepara al alumno de cara a la resolución de problemas complejos en la infraestructura y el funcionamiento de sistemas de transporte público reales.

La estructura de este informe consta de la descripción del problema donde se dará a conocer cuál es la problemática para tratar, luego se hablará brevemente acerca del paradigma orientado a objetos para luego continuar con el diseño de solución, posteriormente se hará un análisis del problema, resultados y conclusión.

II. Descripción del problema

El problema descrito se centra en el desarrollo de un sistema de gestión simplificado para una red de metro utilizando el paradigma de la orientación a objetos. Este enfoque es particularmente adecuado para modelar sistemas complejos con múltiples entidades interrelacionadas, como una red de metro.

La red de metro está compuesta por varios elementos clave: estaciones, tramos, líneas, trenes y conductores. Cada uno de estos elementos tiene sus propias características y comportamientos, lo que los convierte en candidatos naturales para ser modelados como objetos en un sistema orientado a objetos.

Por ejemplo, las estaciones se pueden clasificar en diferentes tipos (estación normal, estación de transferencia, estación terminal, estación de mantenimiento), cada una con características propias. Esto sugiere una estructura de herencia con una clase base

Estación y subclases específicas para cada tipo. Los tramos representan las secciones entre estaciones y tienen atributos como la distancia y el costo. Estos podrían modelarse como objetos que conectan dos objetos de tipo estación.

Las líneas agrupan múltiples tramos, lo que sugiere una relación de composición entre los objetos de línea y tramo. Asimismo, los trenes, que están compuestos por diferentes tipos de vehículos, también sugieren una estructura de composición. Asignar trenes a líneas y conductores a trenes indica relaciones entre estos objetos, las cuales pueden modelarse a través de referencias entre objetos relacionados.

El sistema debe realizar varias operaciones, como calcular distancias y costos, y estas pueden expresarse como métodos de los objetos correspondientes. La creación de nuevos elementos (estaciones, tramos, líneas, trenes) coincide con el concepto de constructores en la programación orientada a objetos.

La necesidad de un sistema robusto y flexible que permita extensiones y modificaciones se alinea bien con los principios de encapsulamiento y abstracción del paradigma orientado a objetos. Estos principios permiten modificar la implementación interna de los objetos sin afectar al sistema en general, siempre que se mantenga la interfaz pública.

Este problema presenta un escenario complejo con múltiples entidades interrelacionadas y diversas operaciones, lo que lo hace naturalmente adecuado para una solución basada en el paradigma de orientación a objetos. Este enfoque permite modelar el sistema de una manera intuitiva que refleja la estructura del mundo real, creando un sistema modular, extensible y fácil de mantener.

III. El paradigma de programación orientada a objetos

El paradigma de programación orientada a objetos (POO) organiza el software en torno al concepto de objetos, que son representaciones de entidades reales o abstractas. Estos objetos contienen datos (atributos) y código (métodos) que trabajan con esos datos.

La POO se basa en varios conceptos claves. Las clases actúan como plantillas para crear objetos. La encapsulación oculta los detalles internos de un objeto, revelando solo lo necesario. La herencia permite crear nuevas clases basadas en las existentes, facilitando la reutilización de código. El polimorfismo permite tratar objetos de diferentes clases de la misma manera si tienen una interfaz común. La abstracción simplifica sistemas complejos, convirtiéndolos en modelos más manejables.

En un proyecto de metro, estos conceptos se aplican en la práctica. Entidades como estaciones, líneas, trenes y conductores se modelan como clases. Cada clase encapsula los datos correspondientes y las operaciones asociadas. Por ejemplo, la clase Tren puede tener métodos para agregar vagones o calcular su capacidad.

Este enfoque orientado a objetos facilita la creación de un sistema modular y flexible. Permite que el sistema sea fácilmente escalable y mantenible, ya que refleja intuitivamente

la estructura del sistema real que se está modelando. Así, el código se vuelve más comprensible y adaptable a futuros cambios o adiciones de nuevas funcionalidades.

IV. Análisis del problema

En el centro de la ciudad, que crece constantemente, surge la necesidad de diseñar y gestionar un sistema de transporte eficiente y escalable: una red de metro. Este desafío representa una oportunidad ideal para aplicar los principios de la programación orientada a objetos, modelando un mundo complejo en una estructura de programación cohesiva y flexible.

La red de metro, aunque simplificada para este ejercicio, representa un ecosistema complejo de componentes interconectados. Imagina una red de estaciones, cada una con su "personalidad" única: algunas son paradas comunes, otras son bulliciosos centros de transbordo donde convergen varias líneas, algunas son estaciones terminales, y unas pocas sirven como tranquilos refugios para el mantenimiento de trenes.

Estas estaciones no existen en el vacío, están conectadas entre sí por tramos de vías, cada uno con su longitud y costos asociados. Estos tramos se pueden comparar con hilos que tejen la tela de la red, conectando los nodos importantes del sistema. A su vez, estos tramos se unen para formar líneas completas: rutas que serpentean por la ciudad, cada una con su propia identidad y propósito.

A lo largo de estas líneas se mueven los verdaderos héroes del sistema: los trenes. Cada uno de ellos representa una combinación de vagones, algunos de los cuales están destinados a ser el vagón delantero o trasero del convoy, mientras que otros forman su cuerpo central. Y estos trenes son operados por maquinistas, cada uno con sus propias habilidades y horarios.

El desafío no consiste solo en representar estos elementos en un software, sino en lograr que interactúen de manera armoniosa. ¿Cómo calcular la longitud total de una línea? ¿Cómo determinar el costo de un viaje entre dos estaciones específicas? ¿Cómo asignar trenes a las líneas y maquinistas a los trenes de manera eficiente? Y quizás la pregunta más interesante: ¿cómo rastrear la ubicación de un tren en cualquier momento?

Además, el sistema debe ser lo suficientemente flexible para adaptarse a los inevitables cambios. La ciudad crece, se requieren nuevas estaciones, las líneas se expanden, se adquieren nuevos trenes. Nuestro software debe ser capaz de evolucionar con estos requisitos sin colapsar bajo su propio peso.

Este escenario representa un microcosmos fascinante donde los principios de la programación orientada a objetos pueden brillar. La encapsulación nos permitirá ocultar los detalles complejos de cada componente, proporcionando interfaces limpias y fáciles de usar. La herencia nos ayudará a modelar las jerarquías naturales que existen en el sistema, como los diferentes tipos de estaciones o vagones. El polimorfismo nos dará la flexibilidad para tratar de manera uniforme con diferentes objetos cuando sea necesario.

En esencia, esta tarea plantea el desafío de tomar un sistema real con toda su complejidad e interrelaciones y traducirlo en un modelo de software que sea tanto preciso en su representación como flexible en su implementación. Es una oportunidad para demostrar cómo la programación orientada a objetos puede ayudarnos a crear sistemas que no solo funcionen, sino que también sean fáciles de entender, mantener y expandir a medida que las necesidades de la ciudad y sus habitantes evolucionan con el tiempo.

V. Diseño de la solución

El proyecto propuesto se implementa utilizando la metodología orientada a objetos, aprovechando las propiedades inherentes a esta, como la encapsulación, la herencia y el polimorfismo. El sistema está dividido en diferentes clases, cada una de las cuales representa una entidad fundamental en la red del metro:

Estación: Representa las estaciones individuales en la red. Cada estación es una instancia de la clase Station, que incluye atributos como un identificador único, el nombre de la estación, el tipo de estación y el tiempo de espera en la estación. Los métodos de la clase gestionan la modificación y consulta de estos atributos.

Sección: Define los segmentos o rutas entre estaciones. La clase Section incluye propiedades como la distancia y los costos asociados. Sus instancias contienen referencias a la estación de inicio y la estación final (ambas del tipo `Estación`) y métodos para gestionar y calcular valores relacionados con cada sección.

Línea: Combina varias secciones en una ruta de metro y gestiona su coherencia y conexiones. La clase line incluye atributos como un identificador único de línea, el nombre de la línea, el tipo de rieles y una colección de instancias de section que la componen. Esta clase proporciona métodos para agregar, modificar y consultar las secciones de la línea.

Tren: Representa los trenes que circulan por las líneas. La clase Train está compuesta por varios vagones, definidos en la clase PassengerCar. Los trenes tienen un identificador único, el nombre del fabricante, el tipo de rieles, la velocidad máxima y la configuración de los vagones de pasajeros. Los métodos permiten gestionar su composición y características operativas.

Metro: Encapsula toda la red y permite realizar operaciones de alto nivel, como la adición de líneas, trenes y la simulación de operaciones. La clase subway contiene listas de líneas, trenes, secciones y estaciones. Proporciona métodos para gestionar estas entidades y ejecutar simulaciones de las operaciones del sistema.

Cada clase es responsable de gestionar su propia lógica y estado, interactuando con otras clases para formar un sistema completo y coherente. La estructura de cada clase incluye constructores, métodos de acceso (getters y setters), así como otros métodos específicos que reflejan las operaciones necesarias en el sistema.

Para resolver tareas específicas de gestión de la red de metro se utilizarán las siguientes técnicas de la metodología orientada a objetos:

Encapsulación: Cada clase oculta los detalles internos de su implementación, proporcionando solo la funcionalidad necesaria a través de métodos públicos. Esto hace que el sistema sea más modular y fácil de mantener.

Herencia: Se utilizará para crear jerarquías de clases en casos donde existan entidades con características comunes. Por ejemplo, `Estación` puede tener subclases que representen tipos específicos de estaciones (por ejemplo, `Estación de Transbordo`, `Estación Terminal`), cada una con comportamientos adicionales o modificados.

Polimorfismo: Permite manejar objetos de diferentes clases derivadas de manera uniforme. Por ejemplo, los métodos que trabajan con una lista de station pueden manejar instancias de cualquier subclase de station sin necesidad de conocer su tipo específico.

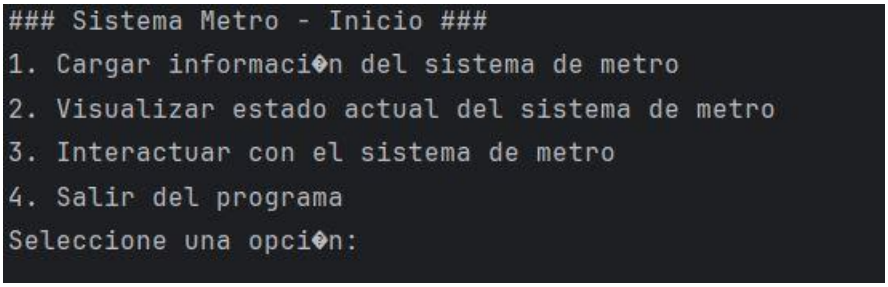
El diseño propuesto garantiza que el sistema no solo sea funcional y eficiente, sino también fácil de mantener y escalable, con la flexibilidad necesaria para adaptarse a futuros cambios y expansiones de la red de metro.

VI. Aspectos de implementación

El proyecto está estructurado en 8 archivos, cada uno conteniendo las clases con cada tipo de dato abstracto necesario para el funcionamiento del sistema, y un octavo archivo (main) que reúne las clases esenciales para utilizar la herramienta de gestión e interacción del metro. Cada archivo de TDA está separado (en caso de existir dicha sección) en Constructores, funciones de Pertenencia, Selectores, Modificadores y Otras funciones, incluyendo las funciones que se requieran para la implementación global del programa. Para el desarrollo se empleó el lenguaje de programación Java utilizando IntelliJ IDEA 2024.1

VII. Instrucciones de uso

Para utilizar el sistema, se debe abrir el cmd, luego en la parte buscar o navegar hasta la dirección donde se encuentra la carpeta Lab3_180311815_VladimirVidalLueiza cabe destacar que debe tener todos los demás archivos en la misma ubicación del archivo main. Luego se debe colocar el comando gradlew.bat build para compilar el proyecto luego se colocar el comando gradlew.bat run para ejecutar el proyecto luego de eso se desplegará el menú del programa.



Luego lo que se debe hacer es cargar la información al sistema de metro, es decir carga los archivos de texto al programa.

```
### Sistema Metro - Cargar informaci□n del sistema de metro ###
1. Creaci□n de una l□nea de metro b□sica (cargar archivo lineas.txt)
2. Combinaciones entre carros (cargar archivo carros.txt)
3. Definici□n de trenes con distintos n□mero de carros (cargar archivo trenes.txt)
4. Conductores asignados a una L□nea (cargar archivo conductores.txt)
5. Retorno al men□ de Inicio
Seleccione una opci□n:
```

Hay que escoger uno por para cargar los datos necesarios al sistema.

```
Seleccione una opci□n: 3
Error al crear el tren: Train ID debe ser positivo
Error al crear el tren: Configuraci□n de carros inv□lida.
Error al crear el tren: Configuraci□n de carros inv□lida.
Se cargaron 2 trenes.
```

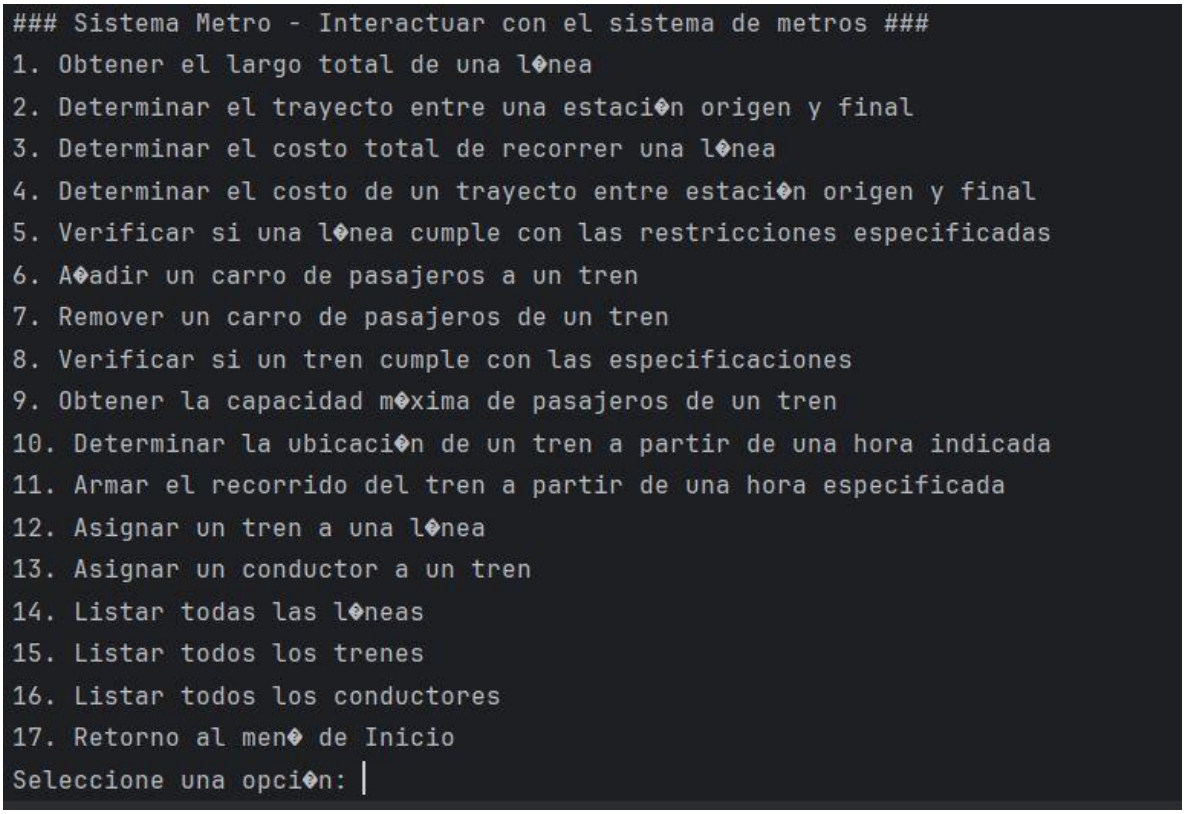
Luego se puede retornar el menú inicio para ver el sistema actual o interactuar con el metro.

```
Seleccione una opci□n: 2

### Sistema Metro - Visualizaci□n del estado actual del sistema de metros ###
Red de Metro: Metro de Santiago (ID: 1)

LINEAS:
TRENES:
  Tren ID: 1
    Fabricante: CAF
    Velocidad: 70
    N□mero de carros: 5
    Capacidad total: 500 pasajeros

  Tren ID: 2
    Fabricante: CAF
    Velocidad: 70
    N□mero de carros: 8
    Capacidad total: 850 pasajeros
```



Se espera que, si los flujos, enlaces y opciones fueron correctamente definidos, las interacciones sigan el diseño propuesto para las relaciones entre un metro que luego se podrá agregar los diferentes actores de este sistema.

VIII. Resultados e implementación

Se logró una implementación que se considera satisfactoria en cuanto coincide adecuadamente con los resultados esperados, coincidiendo la salida de la síntesis que representa que las diferentes funciones realizan las acciones que deben llevar a cabo para que el resultado sea el correcto.

Se utilizo el menú interactivo para hacer uso de los métodos creados para esta implementación, los que entregan el resultado esperado.

Se implementaron adecuadamente, a evaluación personal, 25 de los 25 requerimientos funcionales, pudiendo haber detalles diferentes a los solicitados explícitamente en un par TDAs pero que producen el resultado esperado exitosamente.

IX. Conclusiones

En conclusión, el proyecto aborda de manera efectiva el diseño y desarrollo de un sistema de control para una red de metro simplificada, enfrentando al mismo tiempo los desafíos que plantea la creciente demanda de soluciones de transporte eficientes y sostenibles en entornos urbanos.

Adoptar un enfoque orientado a objetos proporciona un marco robusto y flexible para el procesamiento de datos seguro y eficiente. Este paradigma garantiza la coherencia en operaciones críticas, como la creación y gestión de líneas y estaciones, la planificación de rutas y la administración de tarifas y trenes. Principios como la encapsulación, la herencia y el polimorfismo han demostrado ser especialmente útiles para gestionar estructuras de datos complejas e implementar funcionalidades clave con claridad y precisión.

Además, la estructura del laboratorio contribuyó significativamente al aprendizaje y éxito del proyecto, ya que las tareas se organizaron de forma progresiva, desde las más simples hasta las más complejas. Esta progresión permitió consolidar la comprensión de conceptos fundamentales antes de pasar a problemas más avanzados, facilitando una curva de aprendizaje más eficiente y la aplicación práctica del paradigma orientado a objetos en un contexto realista y complejo.

A pesar de las ventajas, también surgieron desafíos, principalmente relacionados con la curva de aprendizaje del paradigma orientado a objetos y la adaptación de este enfoque a un sistema complejo como el control del metro. Sin embargo, los resultados obtenidos hasta ahora indican que el enfoque elegido es viable y prometedor para el desarrollo continuo y la futura expansión del proyecto, preparando a los estudiantes para resolver problemas complejos de infraestructura y operación en sistemas de transporte público en la vida real.

X. Referencias

Admin. (2022, 9 noviembre). Programación Orientada a Objetos en Java - AEPI. *Escuela de Programación AEPI*. Recuperado 19 de Agosto de 2024, de https://asociacionaepi.es/programacion-orientada-a-objetos-en-java/

Blasco, J. L. (2023, 27 octubre). Introducción a POO en Java: Atributos y constructores. *OpenWebinars.net*. Recuperado 19 de Agosto de 2024, de https://openwebinars.net/blog/introduccion-a-poo-en-java-atributos-y-constructores/

Paradigma Orientado a Objetos. Fundamentos y origen de JAVA. (s. f.). http://dis.um.es/~lopezquesada/documentos/IES_1415/IAW/curso/UT3/ActividadesAlumnos/java7/paginas/pag1.html

XI. Anexo