UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Paradigma Orientado a Objetos - Lenguaje Java

Nombre del Estudiante:	Sebastián Cassone
Curso:	Paradigmas de Programación
Sección	A-1
Profesor:	Gonzalo Martinez

25 de Julio de 2023

Tabla de contenidos

1. Introducción

En este informe, se aborda el uso del paradigma orientado a objetos aplicado a un simulador de sistema de archivos de un sistema operativo, utilizando el lenguaje de programación de alto nivel "Java". Se presenta el problema y la solución propuesta, con énfasis en la programación orientada a objetos y su enfoque en el modelado de objetos y su interacción. Se proporcionan instrucciones y ejemplos para el uso del simulador y se presentan los resultados obtenidos. Al final del informe, se ofrecen conclusiones sobre el trabajo realizado y los resultados obtenidos.

2. Descripción del problema

El problema consiste en diseñar un sistema de archivos que comprenda procesos, métodos y reglas para la organización y almacenamiento de datos en la memoria no volátil del computador. Dentro de este sistema de archivos, se identifican cuatro tipos de objetos: *Drive*, *File*, *Folder*, y *Filesystem*. El objeto *Drive* representa una unidad de almacenamiento, mientras que los objetos *File* y *Folder* son los elementos que se pueden almacenar dentro de un *Drive*. *Filesystem* es el sistema de archivos. El objetivo es implementar este sistema de archivos bajo el paradigma orientado a objetos.

2.1. Problemas del paradigma orientado a objetos en el contexto del diseño del sistema de archivos

1. **Encapsulamiento y acceso a datos:** La programación orientada a objetos enfatiza el encapsulamiento, lo que implica que los atributos de un objeto no son directamente accesibles desde fuera de la clase. Esto podría dificultar la manipulación directa de datos en el sistema de archivos desde otras clases.

2. **Jerarquía y relaciones de objetos:** El diseño del sistema de archivos requerirá establecer relaciones entre los objetos *Drive*, *File* y *Folder*, lo que implica diseñar una jerarquía adecuada. Definir correctamente estas relaciones puede ser un desafío.

3. **Manejo de errores y excepciones:** Los sistemas de archivos deben ser capaces de manejar situaciones de error, como falta de permisos, archivos no encontrados, entre otros. El manejo de excepciones en el paradigma orientado a objetos puede ser más sencillo, pero aún requerirá un diseño adecuado para gestionar estas situaciones.

4. **Polimorfismo:** El polimorfismo es una característica clave de la programación orientada a objetos, que permite tratar objetos de diferentes clases de manera uniforme a través de una interfaz común. Implementar el polimorfismo adecuadamente puede ser esencial para simplificar el código y hacerlo más mantenible.

3. Descripción del Paradigma Orientado a Objetos

El paradigma orientado a objetos es un enfoque de programación que se basa en el concepto de "objetos", que son instancias de clases que encapsulan datos y comportamiento. Cada objeto representa una entidad del mundo real y tiene atributos (variables) y métodos (funciones) que operan sobre esos atributos.

Las características clave del paradigma orientado a objetos son la Encapsulación, la Herencia, el Polimorfismo y el Abstracción.

- Encapsulación: Los atributos de un objeto no son accesibles directamente desde fuera de la clase. En cambio, se acceden mediante métodos públicos, lo que permite controlar el acceso y proteger los datos internos de la clase.

- Herencia: Permite que una clase herede atributos y comportamientos de otra clase (clase base o superclase). Esto facilita la reutilización de código y la creación de jerarquías de clases.

- Polimorfismo: Permite que un objeto de una clase pueda ser tratado como un objeto de su clase base, lo que simplifica el manejo de colecciones de objetos relacionados y mejora la flexibilidad del código.

- Abstracción: Permite modelar entidades del mundo real como objetos y definir sus características esenciales, ignorando los detalles innecesarios.

En la programación orientada a objetos, los programas se componen de clases y objetos que interactúan entre sí para resolver problemas y representar el programa.

4. Análisis del Problema

El problema requiere el uso de objetos para representar el sistema de archivos y sus operaciones. Se busca consultar archivos e información del sistema de archivos de manera eficiente y efectiva, por lo que se deben diseñar clases adecuadas que representen los conceptos clave del sistema de archivos, como unidades, archivos y carpetas. También se deben implementar métodos para realizar operaciones como copiar, mover, eliminar archivos y carpetas, listar contenido, etc.

El paradigma orientado a objetos es adecuado para representar objetos del mundo real y sus interacciones, por lo que será útil para modelar los elementos del sistema de archivos y las operaciones que se pueden realizar sobre ellos.

5. Diseño de la Solución en el Paradigma Orientado a Objetos

El diseño de la solución está basado en la Figura 1, link del UML: `https://drive.google.com/file/d/1L3WgXYGjpTfDfL0RY2GVi3XZXIIj5jQh/view?usp=sharing`. Para diseñar la solución en el paradigma orientado a objetos, se definirán las siguientes clases:

- **Drive:** Representa una unidad de almacenamiento con atributos como nombre, letra, capacidad y una lista de archivos y carpetas que contiene.
- **File:** Representa un archivo con atributos como nombre, contenido y tamaño.
- **Folder:** Representa una carpeta con atributos como nombre, una lista de archivos y carpetas que contiene, y la carpeta padre.
- **FileSystem:** Representa el sistema de archivos en sí y contiene una lista de unidades (*Drive*) y operaciones para realizar en el sistema de archivos.

El sistema de archivos tendrá una estructura jerárquica, donde las unidades contienen archivos y carpetas, y las carpetas pueden contener archivos y subcarpetas. Se utilizarán métodos para realizar operaciones como copiar, mover, eliminar archivos y carpetas, listar contenido, etc.

6. Aspectos de Implementación

Se utilizará el lenguaje de programación Java para implementar la solución orientada a objetos. Se crearán las clases mencionadas anteriormente y se definirán sus atributos y métodos para representar el sistema de archivos y realizar las operaciones requeridas.

7. Instrucciones de Uso

Los usuarios podrán interactuar con el sistema de archivos mediante una interfaz de línea de comandos (CLI) que permitirá ejecutar comandos para realizar diversas operaciones.

Ejemplos de Uso:

Se basa totalmente en el uso del menú interactivo desarollado para que el usuario pueda hacer uso de él.

8. Resultados y Autoevaluación

La implementación orientada a objetos resultó exitosa y se logró cumplir con los requerimientos del sistema de archivos. La interfaz de línea de comandos permitió a los usuarios interactuar con el sistema de archivos de manera sencilla y realizar operaciones de manera eficiente.

9. Conclusiones

El paradigma orientado a objetos demostró ser adecuado para el diseño y la implementación del sistema de archivos, ya que permitió modelar los objetos del mundo real de

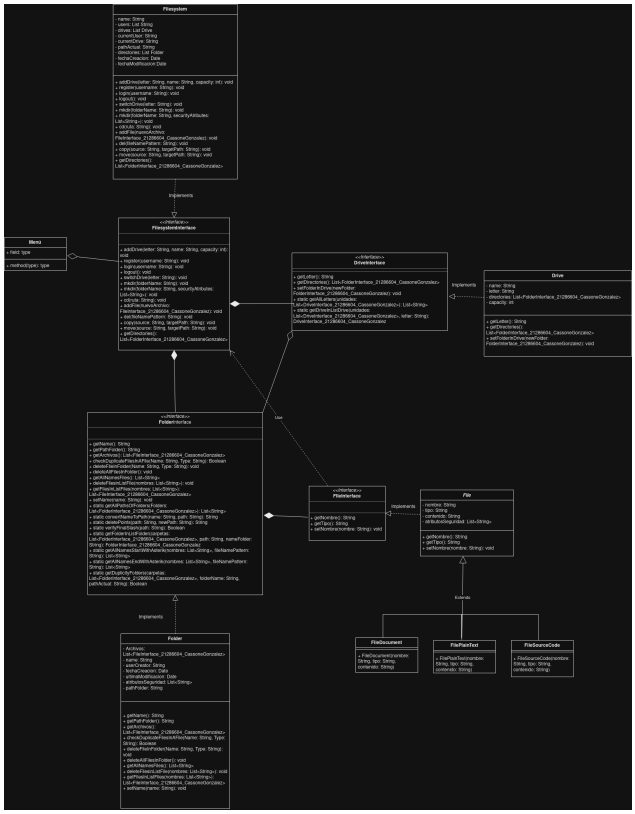

Figura 1: UML de la Solución.

manera efectiva y definir sus interacciones y operaciones. La encapsulación y el polimorfismo fueron especialmente útiles para lograr un código modular y fácil de mantener. En general, la programación orientada a objetos demostró ser una opción sólida para resolver problemas complejos como el diseño de un sistema de archivos.