Laboratorio 1

Principios y Prácticas de Desarrollo de Software Orientado a Objetos

grupo A1

Estudiantes:   
2191954 Daniel Eduardo Cobos Ayala

2172167 Carlos Ivan Ramirez Diaz

Presentado a:

Darío Alejandro Riaño Velandia

Bucaramanga

2023

**Introducción**

En el contexto de nuestro proyecto, nos embarcamos en la elaboración y desarrollo de un juego de blackjack utilizando el lenguaje de programación JavaScript. A lo largo de esta empresa, exploramos a fondo los conceptos fundamentales de la programación orientada a objetos, en particular, la utilización de clases y objetos para estructurar y organizar el código de manera modular. En el presente informe, compartimos una exhaustiva retrospectiva de nuestras experiencias, abordando los desafíos encontrados al introducir nuevas clases en el juego y las mejoras implementadas para optimizar el rendimiento del código. Además, abordamos la transición del paradigma de programación JavaScript al uso del lenguaje de modelado UML, destacando las lecciones aprendidas en este proceso de conversión. Asimismo, ponemos de manifiesto la relevancia de la documentación en cada etapa del proyecto, subrayando su papel crucial como herramienta de comunicación y referencia. Este informe ofrece una perspectiva formal y reflexiva sobre los obstáculos superados y los conocimientos adquiridos durante nuestra travesía en el desarrollo de software.

**Contenido**

[**Ejercicios 1**](#_heading=h.gjdgxs)

[**Conclusiones 1**](#_heading=h.30j0zll)

[**Lecciones aprendidas 2**](#_heading=h.1fob9te)

[**Referencias 3**](#_heading=h.3znysh7)

# Ejercicios

• El trabajo consiste en aplicar principios y prácticas de desarrollo de software orientado a objetos utilizando el código proporcionado, que implementa un juego de Blackjack en JavaScript. Esto implica comprender cómo las clases y métodos están organizados para representar el juego y sus componentes. Además, se requiere analizar cómo se aplican conceptos como encapsulación, modularidad y reutilización de código en el diseño de las clases y en la interacción entre ellas. A través de esta actividad, se busca demostrar cómo los principios de desarrollo orientado a objetos influyen en la estructura y la funcionalidad del juego, proporcionando una base sólida para abordar desafíos similares en el desarrollo de software.

Las clases utilizadas en el código son:

* Carta: Esta clase se utiliza para representar una carta en un juego de blackjack. Contiene propiedades para el valor y el palo de la carta, y métodos para obtener el valor numérico de la carta.
* Baraja: Esta clase se utiliza para representar una baraja de cartas. Se encarga de crear una baraja completa de cartas, barajarlas y repartirlas.
* BlackjackGame: Esta clase representa el juego de blackjack en sí. Administra la lógica del juego para los jugadores y la computadora. Tiene métodos para inicializar el juego, pedir cartas para el jugador, detener al jugador, manejar el turno de la computadora y determinar al ganador.

En resumen, las clases se utilizan para encapsular la funcionalidad relacionada con las cartas, la baraja y el juego de blackjack en objetos reutilizables y modularizados. Esto ayuda a mantener el código organizado y facilita la comprensión y mantenimiento del juego.

# Conclusiones

Explique de manera breve lo resultados obtenidos de la semana

## La implementación del juego de blackjack utilizando clases demuestra la habilidad para estructurar y desarrollar aplicaciones más complejas. La semana involucró la comprensión de conceptos clave como clases, objetos y encapsulación, y aplicó estos conocimientos para crear un juego funcional y modular.

Concepto de encapsulación y abstracción

El código refleja el principio de encapsulación, ya que las clases (Carta, Baraja, BlackjackGame) encapsulan su propia lógica y datos internos. Esto permite ocultar detalles de implementación y separar responsabilidades, lo que facilita la modificación y el mantenimiento. Además, la abstracción se manifiesta al modelar las entidades del juego como clases, extrayendo las características y comportamientos esenciales.

Concepto de clase y objeto

El concepto de clase, que es un plano o plantilla para crear objetos. Cada clase (Carta, Baraja, BlackjackGame) define propiedades y métodos que los objetos creados a partir de ellas pueden tener y ejecutar. Los objetos, por otro lado, son instancias concretas de esas clases y representan elementos específicos del juego.

Diferencia entre clase y objeto

## Una clase es una descripción abstracta que define cómo se comportará un objeto. Contiene las propiedades y métodos que los objetos tendrán. Un objeto, por otro lado, es una instancia concreta de una clase, con valores específicos para sus propiedades. Mientras que una clase es como un plano, los objetos son las construcciones reales basadas en ese plano.

Diseño (UML)

## El diseño del código podría representarse en un diagrama UML (Unified Modeling Language) que mostraría las clases, sus relaciones y los métodos que contienen. Las clases Carta, Baraja y BlackjackGame estarían conectadas, indicando su interacción. Los métodos y propiedades también se mostrarían en el diagrama para visualizar la estructura general.

Codificación y creación de objetos a partir de una clase

## El código muestra cómo se utilizan las clases para crear objetos. Por ejemplo, la instancia juego se crea a partir de la clase BlackjackGame. Esto implica llamar al constructor de la clase para inicializar un nuevo objeto. Los objetos tienen sus propias propiedades y métodos que se pueden acceder y utilizar en función de la clase a la que pertenecen.

# Lecciones aprendidas

## Durante el proceso de desarrollo del proyecto, nos topamos con errores al incorporar nuevas clases, ya que a veces afectaban cosas en el juego que ya estaban funcionando, como las interacciones con el DOM y solicitudes. Aprendimos que agregar nuevas clases requiere planificación y pruebas para evitar dolores de cabeza. Además, notamos que mejorar constantemente el código, hacerlo más ordenado y comprensible, nos facilitó la vida a largo plazo y nos ayudó a encontrar y solucionar errores más rápido. Pasar del JavaScript al UML fue un poco complicado, pero nos enseñó la importancia de entender bien los conceptos del lenguaje de modelado y adaptar nuestro diseño. La documentación resultó ser nuestra mejor amiga. Tomar notas de las decisiones de diseño, la lógica detrás del código y las soluciones a problemas fue como encontrar una brújula en un laberinto. Nos ayudó a comunicarnos mejor entre el equipo y nos sirvió como guía cuando nos quedamos atrapados. En resumen, aprendimos que encontrar un equilibrio entre planificación y adaptación es clave. La resolución de problemas, el código ordenado, los diagramas claros y la documentación detallada son como las herramientas en nuestra mochila de desarrollador para superar cualquier desafío en el camino del software.

# Referencias

Referencias bibliográficas consultadas (Utilice formato de normas APA)