**Trabajo de Investigación: Programación Orientada a Objetos en JavaScript**

**Objetivo:**

El objetivo de este trabajo práctico es investigar cómo aplicar los principios de la

Programación Orientada a Objetos (POO) en JavaScript destacando los cuatros

principios fundamentales: Encapsulamiento, Polimorfismo, Herencia y

Abstracción.

Instrucciones:

● Investigar cómo se aplican los principios de la POO en JavaScript.

● Analizar las ventajas y limitaciones de aplicar estos principios en

JavaScript.

● Basado en la investigación teórica, aplicar ejemplos de cada uno de los

siguientes principios de la POO utilizando JavaScript:

○ Encapsulamiento

○ Polimorfismo

○ Herencia

○ Abstracción

● Presentar ejemplos prácticos, explicando y detallando la sintaxis utilizada en

cada uno de los principios. Los ejemplos deben ser claros y representativos.

Entrega:

● Presentar un documento con capturas del código de cada ejemplo

práctico de POO en JavaScript, junto con una explicación detallada de

cómo se aplican los principios.

**Desarrollo**

**Aplicación de los Principios de la POO en JavaScript**

JavaScript es un lenguaje multiparadigma, lo que significa que soporta múltiples estilos de programación, incluyendo la Programación Orientada a Objetos (POO). En JavaScript, la POO se implementa a través de prototipos y, desde ECMAScript 6 (ES6), mediante la sintaxis de clases.

**Principios de la POO en JavaScript:**

1. **Encapsulación:**
   * La encapsulación se refiere a la agrupación de datos (propiedades) y métodos que operan sobre esos datos dentro de un solo objeto. En JavaScript, los objetos y las clases permiten encapsular datos y comportamiento.
   * Las propiedades pueden ser privadas mediante convenciones (como prefijos con \_) o utilizando closures o símbolos.

**2. Herencia:**

* La herencia permite que una clase (subclase) herede propiedades y métodos de otra clase (superclase). JavaScript permite la herencia prototípica y, desde ES6, la herencia basada en clases usando la palabra clave extends.

**3. Polimorfismo:**

* El polimorfismo permite que las subclases redefinan métodos de sus superclases. Esto permite que un mismo método se comporte de diferentes maneras dependiendo de la subclase que lo implemente.

**4. Abstracción:**

* La abstracción implica crear una representación simplificada de algo complejo mediante la ocultación de detalles innecesarios. En JavaScript, esto se puede lograr mediante la definición de clases abstractas (a través de convenciones) o utilizando interfaces implícitas.

**Ventajas de la POO en JavaScript:**

1. **Reusabilidad:**
   * La herencia y los métodos permiten reutilizar código, lo que reduce la duplicación y facilita el mantenimiento.
2. **Modularidad:**
   * La POO promueve la creación de módulos independientes, lo que mejora la organización y la legibilidad del código.
3. **Escalabilidad:**
   * La estructura orientada a objetos facilita la extensión y modificación de código a medida que crecen los proyectos.
4. **Facilidad de Mantenimiento:**
   * La encapsulación ayuda a mantener el código más ordenado y manejable, facilitando las actualizaciones y correcciones.

**Limitaciones de la POO en JavaScript:**

1. **Sobrecarga de Complejidad:**
   * La POO puede introducir una capa adicional de complejidad innecesaria en aplicaciones simples, lo que podría dificultar su comprensión y mantenimiento.
2. **Rendimiento:**
   * El uso intensivo de clases y herencia puede tener un impacto en el rendimiento en comparación con un enfoque funcional o basado en prototipos, especialmente en aplicaciones que requieren alta performance.
3. **Flexibilidad Limitada:**
   * A pesar de la sintaxis de clases, JavaScript sigue siendo un lenguaje basado en prototipos, lo que significa que algunas características avanzadas de la POO (como la herencia múltiple) no se soportan de manera directa.
4. **Compatibilidad:**
   * Las versiones más antiguas de JavaScript (antes de ES6) no soportan la sintaxis de clases, lo que puede causar problemas de compatibilidad si el código necesita ejecutarse en entornos legacy.