Ejercicio: Sistema de Parque de Diversiones Virtual

Imagina que estás desarrollando un sistema para un parque de diversiones virtual. Deberás crear clases que representen diferentes aspectos del parque, utilizando encapsulamiento, polimorfismo, herencia y abstracción.

1. Crea una clase abstracta llamada `Atraccion` con propiedades como `nombre`, `capacidad`, y `tiempoDeEspera`. Incluye métodos abstractos como `iniciarRecorrido()` y `finalizarRecorrido()`.
2. Implementa al menos tres clases que hereden de `Atraccion`: `MontañaRusa`, `CasaEmbrujada`, y `Carrusel`. Cada una debe implementar los métodos abstractos de manera única.
3. Crea una interfaz `Mantenible` con métodos como `realizarMantenimiento()` y `verificarSeguridad()`.
4. Haz que `MontañaRusa` y `Carrusel` implementen la interfaz `Mantenible`.
5. Desarrolla una clase `ParqueDeDiversiones` que encapsule una colección de atracciones y métodos para añadir/remover atracciones, calcular tiempo de espera promedio, etc.
6. Implementa un sistema de `Visitante` con propiedades encapsuladas como `nombre`, `altura`, y `edad`. Añade métodos para verificar si un visitante puede acceder a una atracción basado en restricciones de altura/edad.
7. Crea un método polimórfico `describirAtraccion()` en la clase `Atraccion` y sobreescríbelo en las clases hijas para proporcionar descripciones únicas.
8. Desarrolla un método en `ParqueDeDiversiones` que use polimorfismo para iniciar todas las atracciones, independientemente de su tipo específico.

Este ejercicio permite practicar:

- Abstracción: Con la clase abstracta `Atraccion`
- Herencia: Con las clases que heredan de `Atraccion`
- Encapsulamiento: En la gestión de propiedades de `Visitante` y `ParqueDeDiversiones`
- Polimorfismo: A través de la implementación de métodos abstractos y el uso de interfaces

Aquí tienes una versión del ejercicio implementada en JavaScript:

```
// Clase abstracta Atraccion
class Atraccion {
    constructor(nombre, capacidad, tiempoEspera) {
        this.nombre = nombre;
        this.capacidad = capacidad;
        this.tiempoEspera = tiempoEspera;
    }

    iniciarRecorrido() {
        throw new Error("Método abstracto");
    }

    finalizarRecorrido() {
        throw new Error("Método abstracto");
    }

    describirAtraccion() {
        throw new Error("Método abstracto");
    }
}

// Interfaz Mantenible
class Mantenible {
    realizarMantenimiento() {
        throw new Error("Método abstracto");
    }

    verificarSeguridad() {
        throw new Error("Método abstracto");
    }
}

// MontañaRusa
class MontañaRusa extends Atraccion {
    constructor(nombre, capacidad, tiempoEspera, alturaMinima) {
        super(nombre, capacidad, tiempoEspera);
        this.alturaMinima = alturaMinima;
    }

    iniciarRecorrido() {
        return `¡La montaña rusa ${this.nombre} inicia su recorrido!`;
    }

    finalizarRecorrido() {
        return `La montaña rusa ${this.nombre} ha completado su recorrido.`;
    }

    describirAtraccion() {
        return `${this.nombre} es una emocionante montaña rusa con una altura mínima de ${this.alturaMinima}cm.`;
    }

    realizarMantenimiento() {
        return `Realizando mantenimiento en la montaña rusa ${this.nombre}.`;
    }

    verificarSeguridad() {
```

```
        return `Verificando sistemas de seguridad de la montaña rusa ${this.nombre}.`;
    }
}

// ParqueDeDiversiones
class ParqueDeDiversiones {
    constructor() {
        this.atracciones = [];
    }

    añadirAtraccion(atraccion) {
        this.atracciones.push(atraccion);
    }

    removerAtraccion(nombreAtraccion) {
        this.atracciones = this.atracciones.filter(a => a.nombre !== nombreAtraccion);
    }

    calcularTiempoEsperaPromedio() {
        if (this.atracciones.length === 0) return 0;
        const total = this.atracciones.reduce((sum, a) => sum + a.tiempoEspera, 0);
        return total / this.atracciones.length;
    }

    iniciarTodasLasAtracciones() {
        return this.atracciones.map(a => a.iniciarRecorrido());
    }
}

// Visitante
class Visitante {
    constructor(nombre, altura, edad) {
        this.nombre = nombre;
        this.altura = altura;
        this.edad = edad;
    }

    puedeAcceder(atraccion) {
        if (atraccion instanceof MontañaRusa) {
            return this.altura >= atraccion.alturaMinima;
        }
        // Añadir más condiciones para otros tipos de atracciones
        return true;
    }
}

// Ejemplo de uso
function main() {
    const parque = new ParqueDeDiversiones();

    const montañaRusa = new MontañaRusa("Dragón de Fuego", 30, 45, 140);
    parque.añadirAtraccion(montañaRusa);

    const visitante = new Visitante("Ana", 150, 25);

    console.log(montañaRusa.describirAtraccion());
    console.log(montañaRusa.iniciarRecorrido());
    console.log(montañaRusa.realizarMantenimiento());
```

```
    console.log(`Tiempo de espera promedio: ${parque.calcularTiempoEsperaPromedio()} minutos`);

    if (visitante.puedeAcceder(montañaRusa)) {
        console.log(`${visitante.nombre} puede acceder a ${montañaRusa.nombre}`);
    } else {
        console.log(`${visitante.nombre} no cumple con los requisitos para ${montañaRusa.nombre}`);
    }

    console.log(parque.iniciarTodasLasAtracciones());
}

main();
```

Para completar el ejercicio, podrías:

1. Implementar más tipos de atracciones (como `CasaEmbrujada` y `Carrusel`).
2. Añadir más funcionalidades a `ParqueDeDiversiones`, como encontrar la atracción más popular.
3. Expandir la lógica de `Visitante` para manejar diferentes tipos de restricciones.
4. Mostrar el Parque de Diversiones gráficamente en una página Web