Universidad Tecnológica de Metropolitana

Estructura de datos

Profesor: Ruth Betsaida Martinez Dominguez

Alumno: Abril Contreras Suaste

Tarea: Actividades del 1 al 4

10 de Octubre de 2024

Actividad 1

Lista de productos disponibles y retirados.

Primero definimos nuestra clase 'Producto', esta clase represena cada producto de la lista. Tiene atributos 'nombre' y 'precio' que almacenan el nombre y el precio del producto, respectivamente. También tiene un atributo 'stock' que almacena el stock disponible del producto. Además, tiene un atributo 'next' que almacena el siguiente producto en la lista.

```
class Producto {
    constructor(id, name, precio, stock) {
        this.id = id;
        this.name = name;
        this.precio = precio;
        this.stock = stock;
        this.next = null;
    }
}
```

Luego, definimos la clase 'ListaProductos' que representa la lista de productos. Tiene un atributo 'head' que almacena el primer producto en la lista. Este atributo se inicializa en null. La clase tiene un método 'agregarProducto(id, name, precio, stock)' que crea un nuevo objeto 'Producto' con los valores proporcionados y lo agrega a la lista. El método 'retirarProducto(id)' busca el producto con el ID proporcionado en la lista y lo retira de la lista. El método 'mostrarProductos()' muestra todos los productos en la lista.

```
class List {
    constructor() {
        this.head = null;
        this.size = 0;
    }

    add(producto) {
        if (!this.head) {
            this.head = producto;
        } else {
            let current = this.head;
            while (current.next) {
                current = current.next;
            }
            current.next = producto;
        }
        this.size++;
    }

    removeById(id) {
```

```
        if (!this.head) return;


        if (this.head.id === id) {
            this.head = this.head.next;
            this.size--;
            return;
        }

        let current = this.head;
        let previous = null;


        while (current && current.id !== id) {
            previous = current;
            current = current.next;
        }


        if (current) {
            previous.next = current.next;
            this.size--;
        }
    }

    removeAll() {
        this.head = null;
        this.size = 0;
    }

    getAllProducts() {
        let current = this.head;
        const products = [];
        while (current) {
            products.push(current);
            current = current.next;
        }
        return products;
    }
}
```

Por último, creamos una instancia de la clase 'ListaProductos' y agregamos algunos productos a la lista. Luego, mostramos todos los productos en la lista. Finalmente, retiramos un producto de la lista y mostramos los productos restantes.

Actividad 2

Separación de numeros en pares e impares.

Primero definimos nuestra clase 'Numero', esta clase represena cada numero de la lista. Tiene un atributo valor que almacena el número y método 'esPar()' que determina si el número es par(True) o impar(false) usando el operador módulo (%).

```
class Numero {
    constructor(valor) {
        this.valor = valor;
}
esPares() {
    return this.valor % 2 === 0;
    }
}
```

Luego, definimos la clase 'ListaNumeros' que representa la lista de números. Tiene un atributo 'numeros' que es una matriz de objetos 'Numero'. Este atributo se inicializa en un array vacío. La clase tiene un método 'agregarNumero(valor)' que crea un nuevo objeto 'Numero' con el valor proporcionado y lo agrega a la matriz 'numeros'. El método 'obtenerPares()' devuelve una nueva matriz con los números pares de la lista. El método 'obtenerImpares()' devuelve una nueva matriz con los números impares de la lista.

```
class ListaNumeros {
    constructor() {
        this.numeros = [];
    }
    agregarNumero(valor) {
        const numero = new Numero(valor);
        this.numeros.push(numero);
    }
    obtenerPares() {
        return this.numeros.filter(numero => numero.esPares()).map(num => num.valor);
    }
    obtenerImpares() {
        return this.numeros.filter(numero => !numero.esPares()).map(num => num.valor);
    }
}
```

Finalmente, creamos una instancia global de la clase 'ListaNumeros' y una función 'generarNumeros()' que genera 20 números aleatorios entre 0 y 99 y los agrega a la lista. Luego, muestra los números pares e impares en la consola.

Actividad 3

Aprobados y Reprobados

Primero definimos nuestra clase 'Estudiante', esta clase represena cada estudiante de la lista. Tiene un atributo 'nombre' que almacena el nombre del estudiante y 'calificacion' que almacena la calificacion del estudiante. Tiene dos metodos 'obtenerAlumnosAprobados()' y 'obtenerAlumnosReprobados()' que devuelve una nueva matriz con los estudiantes aprobados y reprobados respectivamente.

```
class Estudiante {
    constructor(nombre, calificacion) {
        this.nombre = nombre;
        this.calificacion = calificacion;
        }
        aprobado() {
            return this.calificacion >= 7;
            }
}
```

Luego, definimos la clase 'ListaAlumnos' que representa la lista de estudiantes. Tiene un atributo 'lista' que es una matriz de objetos 'Estudiante'. Este atributo se inicializa en un array vacío. La clase tiene un método 'agregarAlumno(alumno)' que agrega un nuevo objeto 'Estudiante' a la matriz 'lista'. El método 'obtenerAlumnosAprobados()' devuelve una nueva matriz con los estudiantes aprobados de la lista. El método 'obtenerAlumnosReprobados()' devuelve una nueva matriz con los estudiantes reprobados de la lista.

```
class ListaAlumnos {
    constructor() {
        this.lista = [];
    }

    agregarAlumno(alumno) {
        this.lista.push(alumno);
    }

    obtenerAlumnos() {
        return this.lista;
    }

    obtenerAlumnosAprobados() {
        return this.lista.filter(alumno => alumno.aprobado());
    }

    obtenerAlumnosReprobados() {
        return this.lista.filter(alumno => !alumno.aprobado());
    }
```

```
}
```

Luego, creamos una instancia global de la clase 'ListaAlumnos' y una función 'mostrarAlumnos()' que muestra los alumnos en la consola. Finalmente, creamos un evento 'click' en el botón 'Agregar Alumno' que agrega un nuevo alumno a la lista y muestra los alumnos en el html.

Actividad 4

Eliminar y ordenar productos

Primero definimos nuestra clase 'Producto', esta clase represena cada producto de la lista. Tiene un atributo 'nombre' que almacena el nombre del producto y 'precio' que almacena el precio del producto. También tiene un atributo 'id' que almacena el ID del producto. Además, tiene un método 'calcularPrecio()' que calcula el precio total de un producto.

```
class Producto {
    constructor(id, nombre, precio) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
    }

    calcularPrecio() {
        return this.precio * this.cantidad;
    }
}
```

Luego, definimos la clase 'ListaProductos' que representa la lista de productos. Tiene un atributo 'lista' que es una matriz de objetos 'Producto'. Este atributo se inicializa en un array vacío. La clase tiene un método 'agregarProducto(producto)' que agrega un nuevo objeto 'Producto' a la matriz 'lista'. El método 'eliminarProducto(id)' elimina un producto de la lista por su ID. El método 'ordenarProductos()' ordena los productos de la lista por precio. El método 'obtenerProductos()' devuelve la lista de productos. El método 'calcularTotal()' calcula el precio total de todos los productos en la lista.

```
class ListaProductos {
    constructor() {
        this.lista = [];
    }

    agregarProducto(producto) {
        this.lista.push(producto);
    }

    eliminarProducto(id) {
```

```
        this.lista = this.lista.filter(producto => producto.id !== id);
    }

    ordenarProductos() {
        this.lista.sort((a, b) => a.precio - b.precio);
    }

    obtenerProductos() {
        return this.lista;
    }

    calcularTotal() {
        return this.lista.reduce((total, producto) => total + producto.precio, 0);
    }
}
```

Finalmente, creamos una instancia global de la clase 'ListaProductos' y una función 'mostrarProductos()' que muestra los productos en el html.