



Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Aragón

Ingeniería en Computación

Asignatura: Estructura de datos

TAREA 2: Implementación del ConjuntoADT

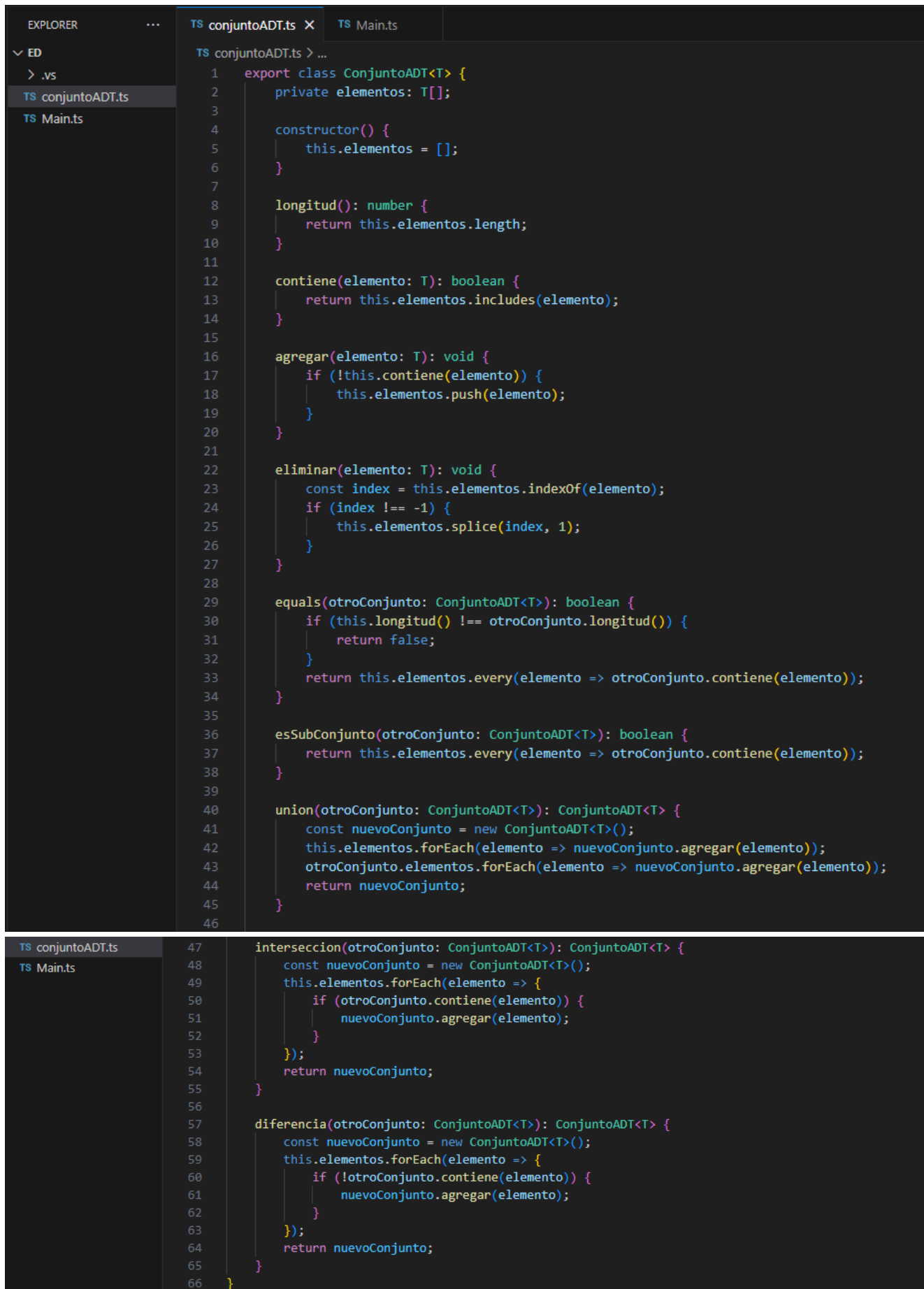
Profesor: Jesús Hernández Cabrera

Alumno: Juan Diego Ortiz Cruz

Grupo: 1360

Fecha: 12/08/2024

Capturas Clase ConjuntoADT:



The image shows a Visual Studio Code editor with two TypeScript files open: `TS conjuntoADT.ts` and `TS Main.ts`. The Explorer sidebar on the left shows the project structure with `.vs`, `TS conjuntoADT.ts`, and `TS Main.ts`. The main editor displays the code for `TS conjuntoADT.ts`, which defines a `ConjuntoADT<T>` class. The code includes a private array `elementos`, a constructor, and methods for `longitud`, `contiene`, `agregar`, `eliminar`, `equals`, `esSubConjunto`, `union`, `interseccion`, and `diferencia`.

```
TS conjuntoADT.ts > ...
1  export class ConjuntoADT<T> {
2      private elementos: T[];
3
4      constructor() {
5          this.elementos = [];
6      }
7
8      longitud(): number {
9          return this.elementos.length;
10     }
11
12     contiene(elemento: T): boolean {
13         return this.elementos.includes(elemento);
14     }
15
16     agregar(elemento: T): void {
17         if (!this.contiene(elemento)) {
18             this.elementos.push(elemento);
19         }
20     }
21
22     eliminar(elemento: T): void {
23         const index = this.elementos.indexOf(elemento);
24         if (index !== -1) {
25             this.elementos.splice(index, 1);
26         }
27     }
28
29     equals(otroConjunto: ConjuntoADT<T>): boolean {
30         if (this.longitud() !== otroConjunto.longitud()) {
31             return false;
32         }
33         return this.elementos.every(elemento => otroConjunto.contiene(elemento));
34     }
35
36     esSubConjunto(otroConjunto: ConjuntoADT<T>): boolean {
37         return this.elementos.every(elemento => otroConjunto.contiene(elemento));
38     }
39
40     union(otroConjunto: ConjuntoADT<T>): ConjuntoADT<T> {
41         const nuevoConjunto = new ConjuntoADT<T>();
42         this.elementos.forEach(elemento => nuevoConjunto.agregar(elemento));
43         otroConjunto.elementos.forEach(elemento => nuevoConjunto.agregar(elemento));
44         return nuevoConjunto;
45     }
46
47     interseccion(otroConjunto: ConjuntoADT<T>): ConjuntoADT<T> {
48         const nuevoConjunto = new ConjuntoADT<T>();
49         this.elementos.forEach(elemento => {
50             if (otroConjunto.contiene(elemento)) {
51                 nuevoConjunto.agregar(elemento);
52             }
53         });
54         return nuevoConjunto;
55     }
56
57     diferencia(otroConjunto: ConjuntoADT<T>): ConjuntoADT<T> {
58         const nuevoConjunto = new ConjuntoADT<T>();
59         this.elementos.forEach(elemento => {
60             if (!otroConjunto.contiene(elemento)) {
61                 nuevoConjunto.agregar(elemento);
62             }
63         });
64         return nuevoConjunto;
65     }
66 }
```

Capturas Main:

EXPLORER	...	TS conjuntoADT.ts	TS Main.ts	X
ED		TS Main.ts > ...		
> .vs				
TS conjuntoADT.ts				
TS Main.ts				
		<pre>1 import { ConjuntoADT } from './conjuntoADT'; 2 3 // Función para imprimir conjuntos 4 function imprimirConjunto<T>(nombre: string, conjunto: ConjuntoADT<T>): void { 5 console.log(`\${nombre}: [\${conjunto['elementos'].join(', ')}]`); 6 } 7 8 // Función principal 9 function main() { 10 // Creación de dos conjuntos 11 const conjuntoA = new ConjuntoADT<number>(); 12 const conjuntoB = new ConjuntoADT<number>(); 13 14 // Agregación de elementos 15 console.log("Agregando elementos a los conjuntos:"); 16 [1, 2, 3, 4, 5].forEach(num => conjuntoA.agregar(num)); 17 [4, 5, 6, 7, 8].forEach(num => conjuntoB.agregar(num)); 18 19 imprimirConjunto("Conjunto A", conjuntoA); 20 imprimirConjunto("Conjunto B", conjuntoB); 21 22 // Eliminación de un elemento 23 console.log("\nEliminando el elemento 3 del Conjunto A:"); 24 conjuntoA.eliminar(3); 25 imprimirConjunto("Conjunto A después de eliminar", conjuntoA); 26 27 // Comprobación de pertenencia 28 console.log("\nComprobación de pertenencia:"); 29 console.log(`¿El Conjunto A contiene 2? \${conjuntoA.contiene(2)}`); 30 console.log(`¿El Conjunto B contiene 3? \${conjuntoB.contiene(3)}`); 31 32 // Operación de unión 33 console.log("\nUnión de conjuntos:"); 34 const union = conjuntoA.union(conjuntoB); 35 imprimirConjunto("A ∪ B", union); 36 37 // Operación de intersección 38 console.log("\nIntersección de conjuntos:"); 39 const interseccion = conjuntoA.interseccion(conjuntoB); 40 imprimirConjunto("A ∩ B", interseccion); 41 42 // Operación de diferencia 43 console.log("\nDiferencia de conjuntos:"); 44 const diferencia = conjuntoA.diferencia(conjuntoB); 45 imprimirConjunto("A - B", diferencia); 46 47 // Comprobación de igualdad 48 console.log("\nComprobación de igualdad:"); 49 console.log(`¿A es igual a B? \${conjuntoA.equals(conjuntoB)}`); 50 51 // Comprobación de subconjunto 52 console.log("\nComprobación de subconjunto:"); 53 console.log(`¿A es subconjunto de B? \${conjuntoA.esSubConjunto(conjuntoB)}`); 54 } 55 56 // Ejecutar la función principal 57 main(); 58</pre>		
TS conjuntoADT.ts				
TS Main.ts				

Capturas consola:

```

✓ ED
  > .vs
  TS conjuntoADT.ts
  TS Main.ts

● PS C:\Users\juani\OneDrive\Escritorio\ED> ts-node Main.ts
Agregando elementos a los conjuntos:
Conjunto A: [1, 2, 3, 4, 5]
Conjunto B: [4, 5, 6, 7, 8]

Eliminando el elemento 3 del Conjunto A:
Conjunto A después de eliminar: [1, 2, 4, 5]

Comprobación de pertenencia:
¿El Conjunto A contiene 2? true
¿El Conjunto B contiene 3? false

Unión de conjuntos:
A ∪ B: [1, 2, 4, 5, 6, 7, 8]

Intersección de conjuntos:
A ∩ B: [4, 5]

Diferencia de conjuntos:
A - B: [1, 2]

Comprobación de igualdad:
¿A es igual a B? false

Comprobación de subconjunto:
¿A es subconjunto de B? false
○ PS C:\Users\juani\OneDrive\Escritorio\ED> 
```