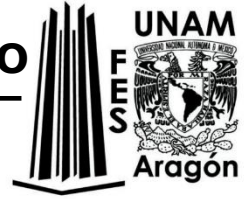




UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES  
ARAGON



## TAREA

### P R E S E N T A

Alexis Hernández Zamudio

### APROFESOR

Jesús Hernández Cabrera

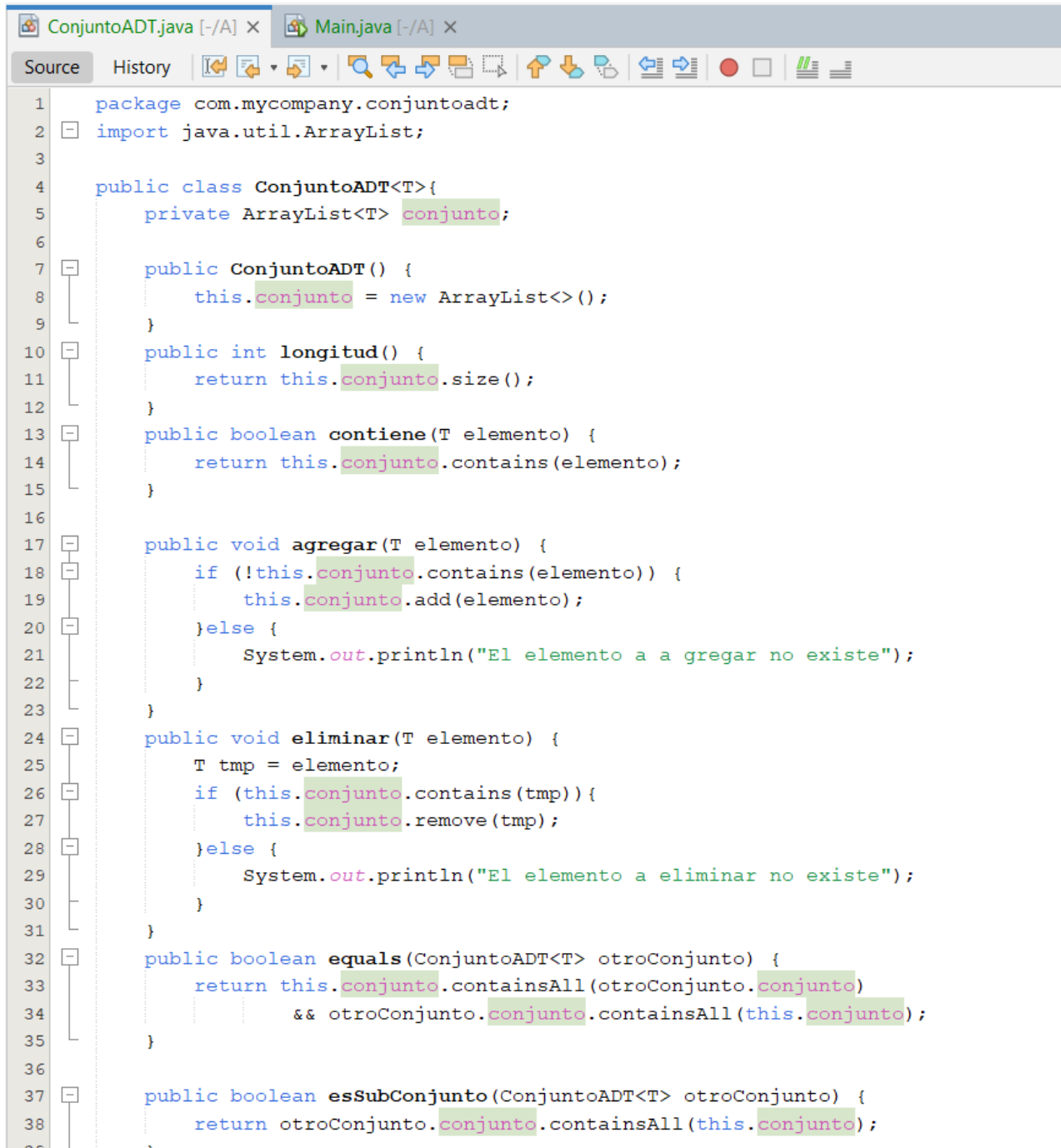
Gpo:1360

URL del repositorio:

<https://github.com/TyrBalder1439/Estructura-de-Datos->



Ciudad Nezahualcóyotl, EDOMEX. 22 de agosto del 2024



The image shows a screenshot of an IDE with two tabs: 'ConjuntoADT.java' and 'Main.java'. The 'ConjuntoADT.java' tab is active, displaying the source code of a Java class. The code defines a generic class 'ConjuntoADT<T>' that uses an 'ArrayList<T>' to store elements. The class includes methods for initialization, size calculation, containment checks, adding and removing elements, and set comparison. The variable 'conjunto' is consistently used to refer to the internal ArrayList.

```
1 package com.mycompany.conjuntoadt;
2 import java.util.ArrayList;
3
4 public class ConjuntoADT<T>{
5     private ArrayList<T> conjunto;
6
7     public ConjuntoADT() {
8         this.conjunto = new ArrayList<>();
9     }
10    public int longitud() {
11        return this.conjunto.size();
12    }
13    public boolean contiene(T elemento) {
14        return this.conjunto.contains(elemento);
15    }
16
17    public void agregar(T elemento) {
18        if (!this.conjunto.contains(elemento)) {
19            this.conjunto.add(elemento);
20        }else {
21            System.out.println("El elemento a a gregar no existe");
22        }
23    }
24    public void eliminar(T elemento) {
25        T tmp = elemento;
26        if (this.conjunto.contains(tmp)) {
27            this.conjunto.remove(tmp);
28        }else {
29            System.out.println("El elemento a eliminar no existe");
30        }
31    }
32    public boolean equals(ConjuntoADT<T> otroConjunto) {
33        return this.conjunto.containsAll(otroConjunto.conjunto)
34            && otroConjunto.conjunto.containsAll(this.conjunto);
35    }
36
37    public boolean esSubConjunto(ConjuntoADT<T> otroConjunto) {
38        return otroConjunto.conjunto.containsAll(this.conjunto);
39    }
40 }
```

```
ConjuntoADT.java [-/A] x Main.java [-/A] x
Source History
40
41 public ConjuntoADT<T> union(ConjuntoADT<T> otroConjunto) {
42     ConjuntoADT<T> resultado = new ConjuntoADT<>();
43     resultado.conjunto.addAll(this.conjunto);
44     for (T elemento : otroConjunto.conjunto) {
45         if (!resultado.conjunto.contains(elemento)) {
46             resultado.conjunto.add(elemento);
47         }
48     }
49     return resultado;
50 }
51
52 public ConjuntoADT<T> interseccion(ConjuntoADT<T> otroConjunto) {
53     ConjuntoADT<T> resultado = new ConjuntoADT<>();
54     for (T elemento : this.conjunto) {
55         if (otroConjunto.conjunto.contains(elemento)) {
56             resultado.conjunto.add(elemento);
57         }
58     }
59     return resultado;
60 }
61
62 public ConjuntoADT<T> diferencia(ConjuntoADT<T> otroConjunto) {
63     ConjuntoADT<T> resultado = new ConjuntoADT<>();
64     for (T elemento : this.conjunto) {
65         if (!otroConjunto.conjunto.contains(elemento)) {
66             resultado.conjunto.add(elemento);
67         }
68     }
69     return resultado;
70 }
71
72 @Override
73 public String toString() {
74     return conjunto.toString();
75 }
76 }
77
```

```
ConjuntoADT.java [-/A] x Main.java [-/A] x
Source History
1
2 package com.mycompany.conjuntoadt;
3
4 public class Main {
5     public static void main(String[] args) {
6
7         ConjuntoADT<Integer> conjuntoA = new ConjuntoADT<>();
8         ConjuntoADT<Integer> conjuntoB = new ConjuntoADT<>();
9
10        conjuntoA.agregar(1);
11        conjuntoA.agregar(2);
12        conjuntoA.agregar(3);
13        conjuntoB.agregar(3);
14        conjuntoB.agregar(4);
15        conjuntoB.agregar(5);
16
17        System.out.println("imprimiendo conjunto A " + conjuntoA);
18        System.out.println("imprimiendo conjunto B " + conjuntoB);
19        System.out.println("eliminando del conjunto A a 2");
20        conjuntoA.eliminar(2);
21        System.out.println("imprimiendo conjunto A " + conjuntoA);
22        System.out.println("¿El conjunto A contiene el 1? " + conjuntoA.contiene(1));
23        System.out.println("¿El conjunto b contiene al 2?" + conjuntoB.contiene(2));
24        ConjuntoADT<Integer> union = conjuntoA.union(conjuntoB);
25        System.out.println("Unión de A y B: " + union);
26        ConjuntoADT<Integer> interseccion = conjuntoA.interseccion(conjuntoB);
27        System.out.println("Intersección de A y B: " + interseccion);
28        ConjuntoADT<Integer> diferencia = conjuntoA.diferencia(conjuntoB);
29        System.out.println("Diferencia de A y B: " + diferencia);
30        System.out.println("¿A es un subconjunto de B? " + conjuntoA.esSubConjunto(conjuntoB));
31        System.out.println("¿A y B son iguales? " + conjuntoA.equals(conjuntoB));
32    }
33 }
34
```

```
--- exec:3.1.0:exec (default-cli) @ ConjuntoADT ---
imprimiendo conjunto A [1, 2, 3]
imprimiendo conjunto B [3, 4, 5]
eliminarndo del conjunto A a 2
imprimiendo conjunto A [1, 3]
¿El conjunto A contiene el 1? true
¿El conjunto b contiene al 2?false
Unión de A y B: [1, 3, 4, 5]
Intersección de A y B: [3]
Diferencia de A y B: [1]
¿A es un subconjunto de B? false
¿A y B son iguales? false
```

---

**BUILD SUCCESS**

---

Total time: 1.493 s

Finished at: 2024-08-22T21:17:37-06:00

---