```
class ConjuntoADT:
    def __init__(self):
        self.elementos = (
            set()
        )  # Usamos un conjunto (set)
interno para manejar los elementos

    def longitud(self):
        return len(self.elementos)

    def contiene(self, elemento):
        return elemento in
self.elementos

    def agregar(self, elemento):
        self.elementos.add(elemento)  #
El set no permite duplicados
automáticamente

    def eliminar(self, elemento):
        self.elementos.discard(
            elemento
        )  # `discard` no lanza error
si el elemento no existe
```

```
    def equals(self, otroConjunto):
        return self.elementos ==
otroConjunto.elementos

    def esSubConjunto(self,
otroConjunto):
        return
self.elementos.issubset(otroConjunto.el
ementos)

    def union(self, otroConjunto):
        nuevo_conjunto = ConjuntoADT()
        nuevo_conjunto.elementos =
self.elementos.union(otroConjunto.eleme
ntos)
        return nuevo_conjunto

    def interseccion(self,
otroConjunto):
        nuevo_conjunto = ConjuntoADT()
        nuevo_conjunto.elementos =
self.elementos.intersection(otroConjunt
o.elementos)
```

```
        return nuevo_conjunto

    def diferencia(self, otroConjunto):
        nuevo_conjunto = ConjuntoADT()
        nuevo_conjunto.elementos =
self.elementos.difference(otroConjunto.
elementos)
        return nuevo_conjunto

    def __str__(self):
        return "{" + ", ".join(map(str,
self.elementos)) + "}"


# Ejemplo práctico
def main():
    conjunto1 = ConjuntoADT()
    conjunto2 = ConjuntoADT()

    # Agregar elementos a los conjuntos
    conjunto1.agregar(1)
    conjunto1.agregar(2)
    conjunto1.agregar(3)
```

```
    conjunto2.agregar(3)
    conjunto2.agregar(4)
    conjunto2.agregar(5)

    # Mostrar los conjuntos
    print("Conjunto 1:", conjunto1)
    print("Conjunto 2:", conjunto2)

    # Comprobar longitud
    print("Longitud del Conjunto 1:",
conjunto1.longitud())
    print("Longitud del Conjunto 2:",
conjunto2.longitud())

    # Verificar si un elemento
pertenece al conjunto
    print("Conjunto 1 contiene 2:",
conjunto1.contiene(2))
    print("Conjunto 1 contiene 4:",
conjunto1.contiene(4))

    # Eliminar un elemento
    conjunto1.eliminar(2)
```

```
    print("Conjunto 1 después de
eliminar 2:", conjunto1)

    # Uniones y diferencias
    print("Unión:",
conjunto1.union(conjunto2))
    print("Intersección:",
conjunto1.interseccion(conjunto2))
    print("Diferencia (Conjunto 1 -
Conjunto 2):",
conjunto1.diferencia(conjunto2))
    print("Diferencia (Conjunto 2 -
Conjunto 1):",
conjunto2.diferencia(conjunto1))

    # Comprobar igualdad
    print("Conjuntos son iguales:",
conjunto1.equals(conjunto2))

    # Comprobar subconjunto
    print(
        "Conjunto 1 es subconjunto de
Conjunto 2:",
conjunto1.esSubConjunto(conjunto2)
```

```
    )
    conjunto2.agregar(1)
    print(
        "Conjunto 1 es subconjunto de
Conjunto 2 después de agregar 1 a
Conjunto 2:",

conjunto1.esSubConjunto(conjunto2),
    )


if __name__ == "__main__":
    main()
```

C: > Users > vicke > Downloads > Tarea2 > tarea2.py > ConjuntoADT

```python
class ConjuntoADT:
    def __init__(self):
        self.elementos = (
            set()
        )  # Usamos un conjunto (set) interno para manejar los elementos

    def longitud(self):
        return len(self.elementos)

    def contiene(self, elemento):
        return elemento in self.elementos

    def agregar(self, elemento):
        self.elementos.add(elemento)  # El set no permite duplicados automáti

    def eliminar(self, elemento):
        self.elementos.discard(
            elemento
        )  # `discard` no lanza error si el elemento no existe
```

```python
    def eliminar(self, elemento):
        self.elementos.discard(
            elemento
        )  # `discard` no lanza error si el elemento no existe

    def equals(self, otroConjunto):
        return self.elementos == otroConjunto.elementos

    def esSubConjunto(self, otroConjunto):
        return self.elementos.issubset(otroConjunto.elementos)

    def union(self, otroConjunto):
        nuevo_conjunto = ConjuntoADT()
        nuevo_conjunto.elementos = self.elementos.union(otroConjunto.elementos
        return nuevo_conjunto
```

```
    def diferencia(self, otroConjunto):
        nuevo_conjunto = ConjuntoADT()
        nuevo_conjunto.elementos = self.elementos.difference(otroConjunto.elem
        return nuevo_conjunto

    def __str__(self):
        return "{" + ", ".join(map(str, self.elementos)) + "}"


 Ejemplo práctico
ef main():
    conjunto1 = ConjuntoADT()
    conjunto2 = ConjuntoADT()
```

}

```
    # Agregar elementos a los conjuntos
    conjunto1.agregar(1)
    conjunto1.agregar(2)
    conjunto1.agregar(3)

    conjunto2.agregar(3)
    conjunto2.agregar(4)
    conjunto2.agregar(5)

    # Mostrar los conjuntos
    print("Conjunto 1:", conjunto1)
    print("Conjunto 2:", conjunto2)

    # Comprobar longitud
    print("Longitud del Conjunto 1:", conjunto1.longitud())
    print("Longitud del Conjunto 2:", conjunto2.longitud())

    # Verificar si un elemento pertenece al conjunto
    print("Conjunto 1 contiene 2:", conjunto1.contiene(2))
```

```python
def main():
    # Mostrar los conjuntos
    print("Conjunto 1:", conjunto1)
    print("Conjunto 2:", conjunto2)

    # Comprobar longitud
    print("Longitud del Conjunto 1:", conjunto1.longitud())
    print("Longitud del Conjunto 2:", conjunto2.longitud())

    # Verificar si un elemento pertenece al conjunto
    print("Conjunto 1 contiene 2:", conjunto1.contiene(2))
    print("Conjunto 1 contiene 4:", conjunto1.contiene(4))

    # Eliminar un elemento
    conjunto1.eliminar(2)
    print("Conjunto 1 después de eliminar 2:", conjunto1)

    # Uniones y diferencias
    print("Unión:", conjunto1.union(conjunto2))
```

```python
    # Eliminar un elemento
    conjunto1.eliminar(2)
    print("Conjunto 1 después de eliminar 2:", conjunto1)

    # Uniones y diferencias
    print("Unión:", conjunto1.union(conjunto2))
    print("Intersección:", conjunto1.interseccion(conjunto2))
    print("Diferencia (Conjunto 1 - Conjunto 2):", conjunto1.diferencia(conjun
    print("Diferencia (Conjunto 2 - Conjunto 1):", conjunto2.diferencia(conjun

    # Comprobar igualdad
    print("Conjuntos son iguales:", conjunto1.equals(conjunto2))

    # Comprobar subconjunto
    print(
        "Conjunto 1 es subconjunto de Conjunto 2:", conjunto1.esSubConjunto(co
    )
    conjunto2.agregar(1)
```

```
    print(
        "Conjunto 1 es subconjunto de Conjunto 2 después d
        conjunto1.esSubConjunto(conjunto2),
    )


if __name__ == "__main__":
    main()
```