ESTRUCTURAS DE DATOS

ADT

Tipos de Datos Abstractos

CONTENIDO

1 *ADT*

2 *Pruebas unitarias*

3 *Herencia*

4 *Práctica*

ADT

Es una descripción abstracta (es decir, sin detalles de implementación) de una estructura de datos, donde se define qué operaciones se pueden realizar y cómo debe comportarse, pero sin decir cómo está construida internamente.

BENEFICIOS

- Abstracción
- Modularidad
- Mantenibilidad
- Reutilización
- Encapsulamiento

COMPOSICIÓN

Datos

Información que almacena

Operaciones

Acciones que permite

Comportamiento

Reglas que definen cómo funcionan las operaciones

¿UN ADT ES UNA CLASE?

¿QUÉ SE HACE EN UNA CLASE AL APLICAR UN ADT?

JUNIT

PRUEBAS UNITARIAS

Fragmentos de código diseñados para comprobar que una unidad específica (función, método o proceso) responde correctamente frente a múltiples situaciones, incluyendo posibles respuestas erróneas.

FRAMEWORK

- JUnit
- TestNG
- Mockito
- Arquillian

BENEFICIOS

Pruebas Automatizadas
Confiabilidad
Facilitan el mantenimiento
Aumentan la productividad

https://www.baeldung.com/junit-assertions

ASSERTIONS

JUNIT GitHub Copilot

Las aserciones son la herramienta con la que comprobamos si el resultado del código es el esperado.

assertEquals / **assertNotEquals**: Verifica la igualdad entre dos valores.

assertTrue / **assertFalse:** Verifica el resultado de una condición.

assertNull / **assertNotNull**: Verifica el estado de un objeto.

assertThrows: Verifica que una excepción específica sea lanzada.

```
public class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }
}
```

```
@Test
public void testSumar() {
    Calculadora calc = new Calculadora();
    assertEquals(5, calc.sumar(2, 3));
}
```

INTERFACES VS IMPLEMENTACIONES

HERENCIA

INTERFACES

Ocurre cuando una clase implementa una interfaz (con implements en caso de java) o una interfaz extiende otra.

IMPLEMENTACIÓN

Es cuando una clase hereda de otra clase (con extends en caso de java) y reutiliza su código ya implementado.

Usamos interfaces cuando queremos garantizar que una clase cumpla con un conjunto de métodos. Usamos herencia entre clases cuando queremos compartir funcionalidad real, como métodos ya implementados o atributos comunes.

HERENCIA

OBJECT

En Java, todas las clases heredan implícitamente de Object
En este caso, al presentarse la herencia, se pueden utilizar algunos métodos importantes a considerar que en muchos casos se deben de sobre-escribir al crear un **ADT**

TOSTRING

Devuelve una representación en forma de texto del objeto. Por defecto, imprime algo como: Persona@1a2b3c

Sobrescribirlo para mostrar contenido útil del objeto.

EQUALS

Por defecto, compara con == (referencias).

Sobrescribirlos para que dos objetos sean considerados iguales por su contenido e instancias

HASHCODE

Devuelve un número entero (int), que representa un valor hash del objeto.

Siempre que sobrescribas equals, también deberías sobrescribir hashCode para mantener coherencia en estructuras como HashSet o HashMap

ESTRUCTURAS DE DATOS

PRÁCTICA

Tipos de Datos Abstractos