DATOS ABSTRACTOS

```
PYTHON
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def saludar(self):
        return f"Hola, soy {self.nombre} y tengo {self.edad} años."

# Uso del TDA
p1 = Persona("Gustavo", 20)
print(p1.saludar())
```

```
JAVA
class Persona {
    String nombre;
    int edad;

    // Constructor
    Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    // Método
    String saludar() {
        return "Hola, soy " + nombre + " y tengo " + edad + " años.";
    }
}

// Uso del TDA
public class Main {
    public static void main(String[] args) {
        Persona p1 = new Persona("Gustavo", 20);
        System.out.println(p1.saludar());
    }
}
```

DATOS ABSTRACTOS

```javascript
class Persona {
    constructor(nombre, edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
    saludar() {
        return `Hola, soy ${this.nombre} y tengo ${this.edad} años.`;
    }
}

// Uso del TDA
let p1 = new Persona("Gustavo", 20);
console.log(p1.saludar());
```

```csharp
using System;
class Persona {
    public string nombre;
    public int edad;

    // Constructor
    public Persona(string nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    // Método
    public string Saludar() {
        return $"Hola, soy {nombre} y tengo {edad} años.";
    }
}

// Uso del TDA
class Program {
    static void Main() {
        Persona p1 = new Persona("Gustavo", 20);
        Console.WriteLine(p1.Saludar());
```

DATOS ABSTRACTOS

```
    }
}
CPP
#include <iostream>
using namespace std;

class Persona {
private:
    string nombre;
    int edad;

public:
    // Constructor
    Persona(string n, int e) {
        nombre = n;
        edad = e;
    }

    // Método
    string saludar() {
        return "Hola, soy " + nombre + " y tengo " + to_string(edad) + "
años.";
    }
};

// Uso del TDA
int main() {
    Persona p1("Gustavo", 20);
    cout << p1.saludar() << endl;
    return 0;
}
```

- En todos los lenguajes, los **TDAs se declaran con clases (o structs en algunos casos)**.
- Permiten **agrupar atributos (datos) y métodos (operaciones)**.
- Son la base de la **programación orientada a objetos**.

¿Qué son los Tipos de Datos Abstractos (TDA)?

Un **Tipo de Dato Abstracto (TDA)** es una forma de **definir nuevos tipos de datos creados por el programador**, distintos a los tipos primitivos (como `int`, `float`, `char`).

Se llaman “abstractos” porque **describen qué operaciones se pueden hacer con ellos, sin importar cómo se implementan internamente**.
En otras palabras:

- **El “qué” importa más que el “cómo”**.
- Tú defines un “molde” (clase o estructura) que agrupa **atributos** (datos) y **métodos** (acciones).

Ejemplo en la vida real

Imagina que un **carro** es un TDA:

- **Atributos (datos):** color, marca, velocidad.
- **Métodos (operaciones):** arrancar(), frenar(), acelerar().

No necesitas saber cómo está construido el motor para **usar** el carro, solo sabes qué puedes hacer con él. Esa es la idea detrás de un TDA.

🔹 Características principales

1. **Definidos por el usuario:** no vienen “de fábrica”, los crea el programador.
2. **Encapsulan datos y operaciones:** juntan atributos + funciones.
3. **Abstracción:** ocultan detalles internos de implementación.
4. **Base de la Programación Orientada a Objetos (POO).**

Ejemplos comunes de TDAs

- **Clases y objetos** en Java, Python, C#, JavaScript y C++.
- **Structs** en C y C++.
- **Colecciones** como pilas, colas, listas, árboles, grafos (todos son TDAs clásicos en estructuras de datos).