Practica de Conceptos

Identificar el concepto de tipos de datos abstractos.

Un tipo de dato abstracto (TDA) es una especificación de un conjunto de operaciones que se pueden realizar sobre un tipo de dato, sin especificar cómo se implementan esas operaciones. Es decir, un TDA define qué se puede hacer con un tipo de dato, pero no cómo se hace.

Características:

- Abstracción: Oculta los detalles de implementación.
- Encapsulación: Protege los datos y las operaciones del TDA.
- Interfaz: Define las operaciones que se pueden realizar sobre el TDA.

Ejemplos de TDA en Java:

- Listas: Almacenan una colección de elementos ordenados.
- Pilas: Almacenan una colección de elementos en orden LIFO (último en entrar, primero en salir).
- Colas: Almacenan una colección de elementos en orden FIFO (primero en entrar, primero en salir).

2. Comprender las diferencias entre tipo de dato y estructuras de datos.

Un tipo de dato es un conjunto de valores que se pueden almacenar y manipular en un programa. Los tipos de datos básicos en Java incluyen int, double, String y boolean.

Una estructura de datos es una forma de organizar y almacenar datos en la memoria de la computadora. Las estructuras de datos comunes incluyen arrays, listas, pilas y colas.

Diferencias:

Tipos de datos: Definen el tipo de valor que se puede almacenar.

Estructuras de datos: Definen cómo se organizan y almacenan los datos.

Un mismo tipo de dato puede implementarse con diferentes estructuras de datos.

3. Analizar el concepto de recursividad y su aplicación.

La recursividad es una técnica de programación en la que una función se llama a sí misma. Es útil para resolver problemas que se pueden dividir en subproblemas más pequeños del mismo tipo.

Ejemplos de aplicaciones de la recursividad:

- Cálculo del factorial de un número
- Impresión de una secuencia de Fibonacci
- Búsqueda binaria en un array ordenado

4. Comprender el manejo de métodos recursivos utilizando los principios de la orientación a objetos.

En la orientación a objetos, los métodos recursivos se pueden utilizar para encapsular la lógica de un algoritmo dentro de una clase. Esto hace que el código sea más modular y fácil de mantener.

Principios para el manejo de métodos recursivos:

Caso base: Es la condición que termina la recursividad.

Paso recursivo: Es la llamada a la función recursiva con un subproblema más pequeño.

Ejemplo de método recursivo en Java:

```java
public class Fibonacci {

    public static int fibonacci(int n) {
        if (n == 0 || n == 1) {
            return n;
        } else {
            return fibonacci(n - 1) + fibonacci(n - 2);
        }
    }

    public static void main(String[] args) {
        System.out.println(fibonacci(10)); // 55
    }
}
```