UCES

Tipos abstractos de Datos y Clases en Java

Programación II

Definición

Un Tipo de Dato Abstracto (TDA), en inglés "Abstract Data Type," es un concepto en programación y ciencias de la computación que se refiere a una estructura de datos que se define en términos de su comportamiento y operaciones permitidas, en lugar de su implementación concreta. En otras palabras, un TDA especifica qué operaciones se pueden realizar con los datos y cuáles son sus propiedades, pero no especifica cómo se implementan internamente esas operaciones.

Para que se usa

Los TDAs son una abstracción importante en la programación porque permiten separar la especificación de datos y operaciones de su implementación. Esto significa que puedes definir cómo se deben comportar los datos y luego proporcionar diferentes implementaciones según tus necesidades. Esto es especialmente útil en el desarrollo de software, ya que te permite cambiar la implementación subyacente sin afectar el código que utiliza el TDA, siempre y cuando las operaciones definidas en el TDA se mantengan consistentes.

Ejemplos

1. **Listas:** Un TDA lista puede definir operaciones como "agregar un elemento," "eliminar un elemento," "obtener el elemento en una posición dada," etc. Puedes implementar esta lista como una lista enlazada, un arreglo, una lista doblemente enlazada, etc.

2. **Pilas:** Un TDA pila puede definir operaciones como "empujar" y "pop." La implementación puede ser una lista enlazada, un arreglo, etc.

Invariante de representación

Un invariante de representación, en el contexto de la programación orientada a objetos y la ingeniería de software, es una condición o restricción que debe mantenerse válida en todo momento para una estructura de datos o una clase. Estas restricciones son esenciales para asegurar la consistencia y la integridad de los datos dentro de la estructura o la instancia de la clase. Los invariantes de representación son parte fundamental de la encapsulación y el diseño de clases en la programación orientada a objetos.

En otras palabras, un invariante de representación establece las condiciones que deben cumplirse en un objeto o estructura de datos en un momento dado o después de realizar ciertas operaciones. Siempre que se realice una operación en el objeto o la estructura, el invariante de representación debe seguir siendo válido. Esto garantiza que los datos se mantengan en un estado coherente y que no se violen las restricciones establecidas.

Ejemplos de invariantes de representación:

1. **Invariante de representación para un TAD de fecha:** En un TAD que representa fechas, un invariante podría ser que el día, el mes y el año estén dentro de rangos válidos y coherentes entre sí.

2. **Invariante de representación para una estructura de datos de pila:** En una pila, un invariante podría ser que los elementos se agreguen al principio y se retiren del principio de la pila.

Consigna de ejemplo

Dado un pequeño juego de simulación de las naves, donde cada nave tiene un nombre, un color y coordenadas, se pide agregar comportamiento: mover nave hacia arriba, mover nave hacia abajo. La nave debe tener obligatoriamente un nombre y coordenadas. Las coordenadas deben ser positivas, en eje x hasta, 600, en eje y hasta, 400. Se pide identificar los TAD e IREP.

Ejemplo TAD

Atributos: nombre, color y coordenadas (posición en el eje x y eje y).

Operaciones: mover nave hacia arriba, mover nave hacia abajo.

IREP: Cada nave debe tener obligatoriamente un nombre y coordenadas.

Las coordenadas deben ser positivas, con un límite de 600 en el eje x y 400 en el eje y.

Clases en java

Una clase es una plantilla o un plano para crear objetos.

Define la estructura y el comportamiento de los objetos que se crearán a partir de ella.

En una clase, se especifican los atributos (variables de instancia) y los métodos (funciones) que los objetos de esa clase tendrán.

Ejemplo de Clase

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

    public void saludar() {
        System.out.println("Hola, soy " + nombre + " y tengo " + edad + " años.");
    }
}
```

Objeto en java

Un objeto es una instancia concreta de una clase.

Representa una entidad del mundo real que tiene atributos y puede realizar acciones.

Los objetos son creados a partir de una clase mediante la palabra clave new.

Ejemplo de objeto

```
Persona persona1 = new Persona(); // Creación de un objeto de la clase Persona

persona1.nombre = "Juan";

persona1.edad = 30;

persona1.saludar(); // Llamada a un método del objeto
```

Instancias

El término "instancia" se utiliza a menudo de manera intercambiable con "objeto".

Una instancia es una representación individual de un objeto concreto creado a partir de una clase.

Cada instancia tiene su propio conjunto único de valores para los atributos definidos en la clase.

Ejemplo de instancia

Persona persona1 = new Persona(); // Instancia u objeto 1 de la clase Persona

Persona persona2 = new Persona(); // Instancia u objeto 2 de la clase Persona

Implementación en Java del ejemplo de Naves

```
public class Nave {
    private String nombre;
    private String color;
    private int coordenadaX;
    private int coordenadaY;

    public Nave(String nombre, String color, int coordenadaX, int coordenadaY) {
        this.nombre = nombre;
        this.color = color;
        this.coordenadaX = coordenadaX;
        this.coordenadaY = coordenadaY;
    }


}
```

Implementación en Java (continuación de diapo. anterior)

```
public void moverNaveArriba(int cantidad) {
        if (coordenadaY + cantidad <= 400) {
            coordenadaY += cantidad;
        } else {
            System.out.println("La nave no puede moverse más arriba.");
        }
    }

    public void moverNaveAbajo(int cantidad) {
        if (coordenadaY - cantidad >= 0) {
            coordenadaY -= cantidad;
        } else {
            System.out.println("La nave no puede moverse más abajo.");
        }
    }

    // Otros métodos, getters y setters según sea necesario.
```

Ejercicio

Se pide preparar un pequeño sistema para administrar, con el menor uso posible de memoria, los datos de 5 clientes, cada cliente tiene nombre, apellido y edad de forma obligatoria. Se pide agregar, quitar clientes. El sistema tiene una descripción y una fecha de actualización de datos. Los clientes pueden tener hasta 3 teléfonos. Cada cliente debe tener por lo menos un teléfono. Se piden los TAD necesarios indicando datos y operaciones e invariantes de representación. Realizar la implementación en java.