UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

Facultad de Ingenieria

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Modelos de Programacion II

Sintaxis y Semantica POO

Proyecto Curricular: Ingenieria de Sistemas

Nicolas Andres Mayorga Velasquez - 20211020078

Daniel Mateo Montoya Gonzales – 20202020098

Laura Carina Alvarez Campos – 20212020006

Lilly Sofia Ayala Rojas - 20211020020

01 de Septiembre del 2023

Sintaxis y semántica de Poo:

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en la idea de estructurar un programa en torno a objetos, que son entidades que encapsulan datos y funcionalidades relacionadas. Al utilizar la POO, es importante seguir ciertas reglas de sintaxis y semántica para asegurar que el código sea correcto y coherente.

Reglas de sintaxis en la Programación Orientada a Objetos:

1. ***Clases***: En la POO, se utilizan clases para definir la estructura y el comportamiento de los objetos. Una clase se define con la palabra clave `class`, seguida del nombre de la clase y el cuerpo de la clase encerrado entre llaves `{}`.

   Ejemplo:

   ```
   class MiClase {

       // Cuerpo de la clase

   }
   ```

2. **Objetos**: Los objetos se crean a partir de una clase utilizando el operador `new`. Se puede acceder a los miembros de un objeto utilizando el operador punto `.`

   Ejemplo:

   ```
   MiClase objeto = new MiClase();

   objeto.metodo(); // Acceso a un método de la clase

   objeto.atributo = 10; // Acceso a un atributo de la
   clase.
   ```

3. **Métodos**: Los métodos son funciones asociadas a una clase y definen el comportamiento de los objetos de esa clase. Se definen dentro de la clase y pueden ser llamados desde los objetos de la clase.

   Ejemplo:

   ```
   class MiClase {
       void metodo() {
           // Cuerpo del método
       }
   }
   ```

4. **Atributos**: Los atributos son variables asociadas a una clase y representan el estado de los objetos de esa clase. Se definen dentro de la clase y pueden ser accedidos desde los objetos de la clase.

   Ejemplo:

```
class MiClase {
    int atributo; // Declaración de un atributo
}
```

Reglas de semántica en la Programación Orientada a Objetos:

1. **Encapsulación**: La encapsulación es un principio importante en la POO que consiste en ocultar los detalles internos de una clase y proporcionar una interfaz pública para interactuar con los objetos. Se utiliza el modificador de acceso `public`, `private` y `protected` para controlar el acceso a los miembros de una clase.

   Ejemplo:

```
class MiClase {
    private int atributo; // Atributo privado
    public void metodo() {
        // Cuerpo del método
    }
}
```

2. **Herencia**: La herencia es un mecanismo que permite definir una nueva clase basada en una clase existente, heredando sus atributos y métodos. La clase base se conoce como superclase o clase padre, y la clase derivada se conoce como subclase o clase hija.

   Ejemplo:

```
class SuperClase {
    // Atributos y métodos de la superclase
}

class SubClase extends SuperClase {
    // Atributos y métodos de la subclase
}
```

3. **Polimorfismo**: El polimorfismo es la capacidad de un objeto de tomar muchas formas diferentes. Permite que un objeto de una clase base pueda ser tratado

como un objeto de una clase derivada. Esto se logra mediante el uso de métodos sobrescritos y la implementación de interfaces.

Ejemplo:

```
interface MiInterfaz {
    void metodo();
}

class MiClase implements MiInterfaz {
    void metodo() {
        // Cuerpo del método
    }
}

MiInterfaz objeto = new MiClase();
objeto.metodo(); // Llamada al método de la clase MiClase
```

Al seguir las reglas de sintaxis y semántica de la POO, podemos crear clases, objetos, métodos y atributos que nos permiten encapsular datos y funcionalidades relacionadas. Las clases definen la estructura y el comportamiento de los objetos, mientras que los objetos son instancias de una clase. Los métodos definen el comportamiento de los objetos y los atributos representan el estado de los objetos.

Es importante tener en cuenta la encapsulación, que consiste en ocultar los detalles internos de una clase y proporcionar una interfaz pública para interactuar con los objetos. Además, la herencia permite definir nuevas clases basadas en clases existentes, heredando sus atributos y métodos. El polimorfismo, por otro lado, nos permite tratar un objeto de una clase base como un objeto de una clase derivada.

La POO es un enfoque poderoso y ampliamente utilizado en el desarrollo de software, ya que nos permite escribir código modular, reutilizable y fácil de mantener. Al comprender y aplicar correctamente la sintaxis y semántica de la POO, podemos aprovechar al máximo este paradigma de programación.

En conclusión, la Programación Orientada a Objetos nos brinda una forma eficiente y estructurada de desarrollar software, permitiéndonos crear programas más flexibles, escalables y fáciles de mantener.