MisiónTIC 2022-Ciclo 2

Fundamentos de Programación

Jairo Armando Riaño Herrera

21 de julio de 2021

Contenidos

Definición

La Programación Orientada a Objetos (POO) se define como:

- "Paradigma de Programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos" [1]
- "La *Programación Orientada a Objetos*, organiza un programa alrededor de sus datos (es decir, objetos) y de un conjunto de interfaces bien definidas para esos datos. Un programa orientado a objetos se puede definir como un conjunto de datos que controlan el acceso al código" [2]

Definición

La Programación Orientada a Objetos (POO) se define como:

- "Paradigma de Programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos" [1]
- "La *Programación Orientada a Objetos*, organiza un programa alrededor de sus datos (es decir, objetos) y de un conjunto de interfaces bien definidas para esos datos. Un programa orientado a objetos se puede definir como un conjunto de datos que controlan el acceso al código" [2]

Definición de Clase

Una clase es una plantilla que permite definir las características (atributos) y comportamiento (métodos) de los objetos que se crearán (instancias) a partir de la clase. Una clase es una especie de molde para los objetos o instancias de la clase.

Código Java

```
1   class Computer{
2    private String idComputer;
3    private String brand;
4    private double value;
5    private boolean state;
6
7    public Computer(String
    idComputer, String brand, double
    value){
8      this.idComputer = idComputer;
9      this.brand = brand;
10     this.value = value;
11     state = false;
12   }
13
14   public setTurnOn(){
15     state = true;
16   }
17
18   public setTurnOff(){
```

Definición de Objeto

Un objeto es una instancia de una clase, a través de un objeto se accede a los miembros de la clase (atributos, métodos). Un objeto toma las características (atributos) de la clase y puede realizar las acciones definidas en sus métodos (comportamiento).

Código Java

```
1  class RunComputer{
2    static public void main( String[] args ){
3      //Se instancia un objeto acer de la clase Computer
4      Computer acer = new Computer ("S/N 234233","Acer",780000 );
5
6      //Se instancia un objeto asus de la clase Computer
7      Computer asus = new Computer ("S/N 97434","Asus",1200000 );
8
9      if ( asus.isState( ) ){
10       System.out.println(''Pc Asus Prendido'');
11     }
12     else{
13       System.out.println(''Pc Asus Apagado'');
```

Características

- *Abstracción.* Permite que se pueda usar un objeto sin conocer su estructura interna, en POO, se puede interactuar con una clase sin conocer su lógica interna.
- *Encapsulamiento.* Una clase encapsula tanto sus datos (atributos o variables), como las operaciones que se pueden realizar con ellos a través de los métodos, el encapsulamiento evita que desde otras clases se cambien valores de los atributos, solamente lo deben poder hacer los miembros de la clase. El encapsulamiento evita los efectos colaterales de una variable en un programa, es decir, efectos no deseados cuando no se controla en forma adecuada el ámbito de las variables.

Características

- *Abstracción.* Permite que se pueda usar un objeto sin conocer su estructura interna, en POO, se puede interactuar con una clase sin conocer su lógica interna.
- *Encapsulamiento.* Una clase encapsula tanto sus datos (atributos o variables), como las operaciones que se pueden realizar con ellos a través de los métodos, el encapsulamiento evita que desde otras clases se cambien valores de los atributos, solamente lo deben poder hacer los miembros de la clase. El encapsulamiento evita los efectos colaterales de una variable en un programa, es decir, efectos no deseados cuando no se controla en forma adecuada el ámbito de las variables.

Características

- *Herencia.* Permite la reutilización de software haciendo que una nueva clase (extendida) herede los atributos y métodos de otra clase (base), mejorándola con nuevas capacidades o mdificando las que ya existen.
- *Polimorfismo.* Permite que un objeto tome diferentes comportamientos, dependiendo del contexto en que se utilice, por ejemplo, el comportamiento *liquidarNomina()* en una clase Empleado, es diferente si se trata de un Empleado de planta o un Empleado por días.

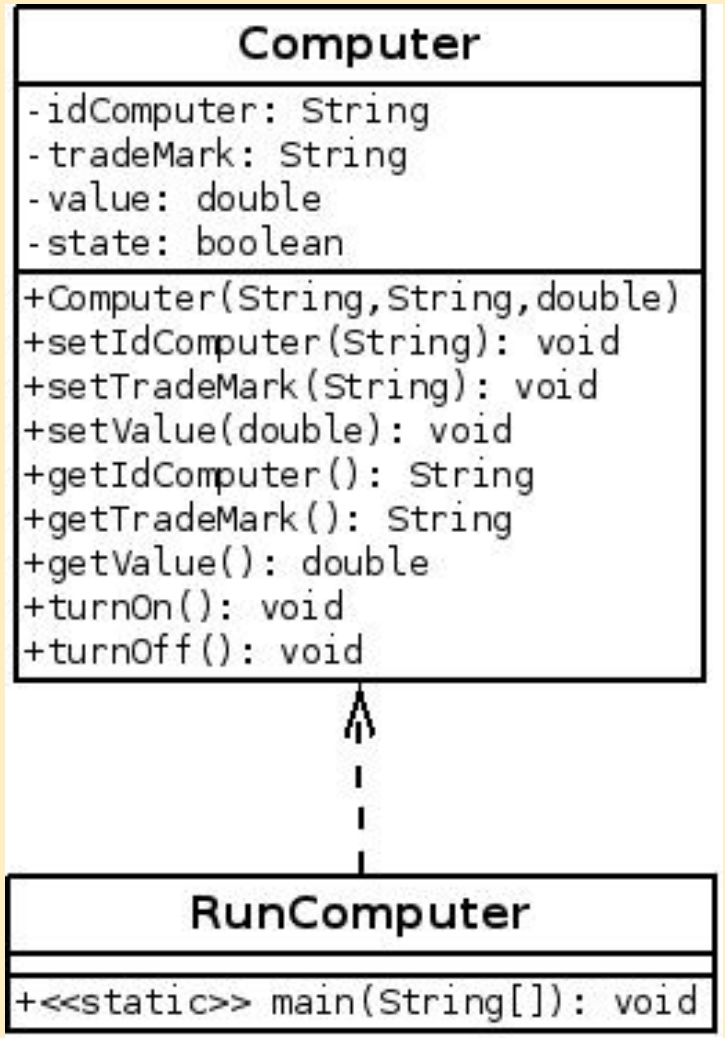
Características

- *Herencia.* Permite la reutilización de software haciendo que una nueva clase (extendida) herede los atributos y métodos de otra clase (base), mejorándola con nuevas capacidades o mdificando las que ya existen.
- *Polimorfismo.* Permite que un objeto tome diferentes comportamientos, dependiendo del contexto en que se utilice, por ejemplo, el comportamiento *liquidarNomina()* en una clase Empleado, es diferente si se trata de un Empleado de planta o un Empleado por días.

UML

El *Unified Modeling Language* (UML) permite especificar diferentes aspectos de un sistema de información a través de diagramas. Un diagrama de clases especifica los objetos de importancia del dominio de un problema. El programador interpreta el diagrama y lo implementa o codifica en un lenguaje de programación orientado a objetos

Ejemplo



Clase UML

Una clase se representa mediante un rectángulo dividido en tres partes como se ilustra en la figura de la derecha. Para el nombre de la clase usar un sustantivo que describa la clase y si es una palabra compuesta usar el estilo de escritura *CamelCase* (Primer letra de cada palabra en mayúscula), los atributos se definen con la primer letra en minúscula y si el identificador es compuesto, igual, aplicar *CamelCase*. Los nombres de los métodos son verbos (denota acción o comportamiento) y aplica la misma norma de los atributos (primer letra en minúscula y CamelCase)

Figura de Clase

Modificadores de Acceso

Definición

Los modificadores de acceso definen la forma como se acceden a los miembros de una clase (atributos y métodos) desde otras clases. La siguiente tabla resume los diferentes modificadores de acceso y su alcance.

Modificador	Misma Clase	Mismo Paquete	Subclase Otro Paquete	Universo
private (-)	Si	No	No	No
default (∼)	Si	Si	No	No
protected (#)	Si	Si	Si	No
public (+)	Si	Si	Si	Si

Cuando no se especifica el modificador de acceso para un atributo o un método, se asume el modificador por defecto (*default*) con el alcance especificado en la tabla

Referencias

[1] Wikipedia, "Programación orientada a objetos." [Web; accedido el 06-01-2016].

[2] S. Herbert, *Java Manual de Referencia*. México: Mc Graw Hill Educación, 2009.