Título:	Introducción a la Programación Orientada a Objetos
Tipo:	Monografía
Autor:	Franklin Alexis Díaz Díaz
Fecha:	Diciembre 2018
Palabras claves:	Clase, objeto, constructor
Aprobado por:	Decano de la Facultad de Ingeniería Mg. Ing. Enrique Durand Bazán
Firma:	

INTRODUCCIÓN

La presente monografía tiene por objetivo adentrarnos en el mundo de la Programación Orientada a Objetos, en el cual comenzaremos abordando los conceptos fundamentales de este paradigma como son las Clases y Objetos, la relación estrecha entre ambos conceptos y los componentes que lo conforman.

Se detalla a profundidad los elementos que conforman las clases en la Programación Orientada Objetos (POO), como están estructurados y los estereotipos que utilizan en sus sintaxis, además de cómo se pueden vincular con el método principal (*main*) del programa.

Comprenderemos como podemos acceder a los elementos de una clase con los denominados controles de accesos, en el cual estudiaremos sus 3 tipos básico que son comunes en la mayoría de los lenguajes de programación: público, privado y protegido.

Así mismo conoceremos algunos conceptos nuevos como método constructor, métodos GET y SET aplicados a los atributos de una clase, el operador *this*.

Aplicaremos un ejemplo práctico con su respectivo código fuente en la creación de clases y objetos para su ejecución, explicando cada línea de código.

PROGRAMACIÓN ORIENTADA A OBJETOS

La tecnología orientada a objetos se define como una metodología de diseño de software que modela las características de objetos reales o abstractos por medio del uso de clases y objetos. Hoy en día, la orientación a objetos es fundamental en el desarrollo de software, sin embargo, esta tecnología no es nueva, sus orígenes se remontan a la década de los años sesenta.

Las clases tienen 2 elementos bien identificados, los atributos y los métodos.

Clase

Una clase actúa como una plantilla, modelo o prototipo a partir de la cual se obtienen instancias habitualmente llamados *objetos*. Dichos objetos son como "clones" o copias repetidas que ocuparán un espacio de memoria diferente para cada uno de ellos.

Atributos y métodos de una clase

- Los atributos constituyen la estructura interna de los objetos de una clase. Son las propiedades que manifiestan todos los objetos de una clase, se utilizan para describir, identificar o informar el estado solo de información necesaria y esencial dependiendo del entorno de la situación.
- Los métodos forman lo que se denomina interfaz o medio de acceso a la estructura interna de los objetos. Desde el punto de la POO, el conjunto de estos métodos se corresponde con el conjunto de mensajes a los que los objetos de una clase pueden responder. Son algoritmos que procesan los datos o atributos de un objeto. Pueden ser:
  - ✓ Métodos que retornan un valor. Se identifican porque usan la palabra reservada *return* en muchos lenguajes de programación.
  - ✓ Métodos que no retornan ningún valor.

Método constructor de una clase

Es un método especial de la clase que se aplica automáticamente a los objetos en el momento de su instanciación (creación del objeto de la clase). Se utilizan para inicializar (darles valores iniciales los atributos del objeto de la clase.

Se nombran igual que la clase y generalmente llevan tantos parámetros como atributos tenga la clase; no es una regla ya que existen excepciones como el caso de los atributos *static*. Pueden sobrecargarse, lo que indica que se pueden definir varios constructores los cuales deben diferenciarse en los parámetros pero manteniendo el mismo nombre.

Control de acceso a los miembros de una clase

El concepto de clase incluye la idea de la ocultación de datos, que consiste en que no se puede acceder directamente a los atributos de un objeto, sino es por medio de los métodos de la propia

clase. Lo que implica que el usuario solo tendrá acceso a uno o más métodos que le permitirán acceder a los miembros privados, ignorando la disposición de éstos.

Para controlar el acceso a los miembros de una clase, JDK provee las palabras reservadas *private, protected, public*. De omitir uno de estas palabras, por defecto se asume que es *public*. Estas palabras se denominan *modificadores de acceso*, y son utilizadas para indicar el tipo de acceso permitido a cada miembro de la clase.

Acceso público

Un miembro de una clase declarado como *public* puede ser accedido por un objeto de esa clase en cualquier parte de la aplicación donde el objeto en cuestión sea accesible. Los miembros públicos de una clase constituyen la interfaz pública de los objetos de la clase.

Acceso privado

Un miembro de una clase declarado como *private* puede ser accedido por un objeto de esa clase solo desde los métodos de dicha clase. Esto significa que no puede ser accedido por los métodos de cualquier otra clase, incluidas las subclases, ni por las funciones externas de la aplicación.

Acceso protegido

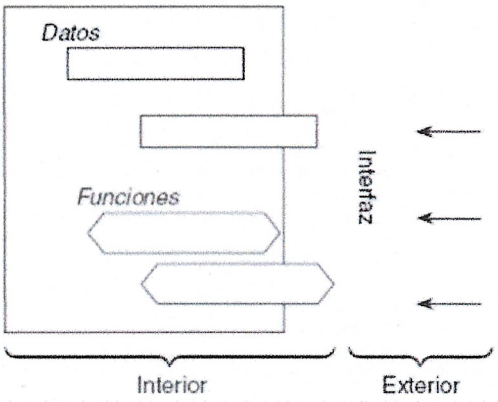
Un miembro de una clase declarado como *protected* se comporta exactamente como uno de tipo *private* para las funciones externas o métodos de cualquier otra clase, pero actúa como *public* para los métodos de sus subclases.

Objeto

El objeto es el elemento principal de la programación orientada a objetos y alrededor del cual gira este paradigma, de ahí el nombre Programación Orientad a Objetos (POO). Es una entidad que representa un concepto o una entidad del mundo real.

Un objeto es una *instancia* de una clase, la cual es una plantilla o modelo para crear objetos. Mientras que una clase define los atributos y métodos que usarán los objetos que se deriven de ella, pero no contiene ningún valor específico para esos atributos. Los objetos por otro lado, contienen valores específicos para los atributos y proporcionan una forma de acceder y manipular esos valores a través de los métodos de la clase.

De forma gráfica podemos ver a una clase, compuesto por atributos y métodos (funciones), la cual es accedida desde la interfaz gráfica de usuario (GUI) de manera externa. Esto nos señala que las clases son archivos independientes del resto de código fuente que contiene nuestra aplicación.



Sintaxis de la una clase en Java (JDK):

```
class nombre_de_la_clase{
        atributos
        método_constructor()
        métodos();
}
```

Se utiliza los { } como delimitadores del contenido de dicha clase.

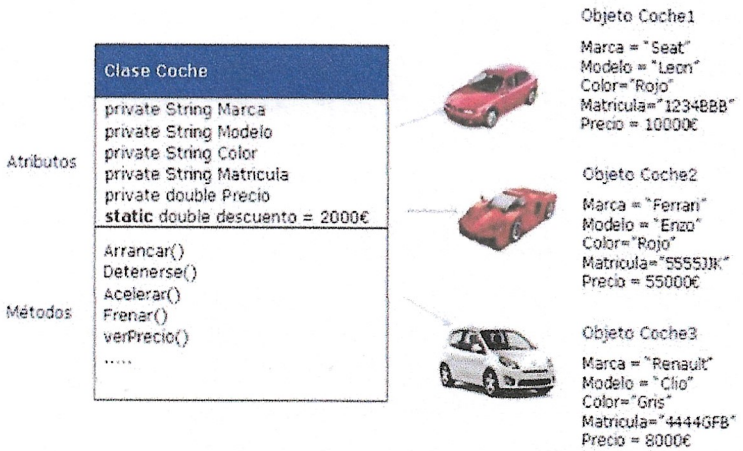
Para la creación de un objeto de una clase, se utiliza el operador *new*. La sintaxis sería de la siguiente manera:

```
nombre_de_la_clase  nombre_del_objeto = new nombre_de_la_clase ()
```

A continuación del nombre de la clase se ha puesto (), que identifica al método "constructor" que se ejecutará en la instanciación. El "constructor" es un método cuyo objetivo es inicializar los valores de los atributos de los objetos.

En este otro ejemplo en forma de imagen, se puede apreciar la clase Coche con sus atributos marca, modelo, color, matrícula, precio y descuento (éste está como *static*); y sus métodos arrancar(), detenerse(), acelerar(), frenar(), verPrecio().

Se nota que los valores asignados a los atributos corresponden a los objetos Coche1, Coche2 y Coche3, y no a la clase en sí. Además los métodos se les identifican con paréntesis.



Vemos en la imagen que el atributo descuento está declarado como *static*, lo cual indica que ya tiene un valor predefinido.

Veamos un ejemplo práctico

Considere una entidad bancaria, en ella se puede identificar elementos que son cuentas bancarias. Una cuenta bancaria puede verse como un objeto que tiene atributos, propiedades o características tales como nombre del titular, número de la cuenta, saldo, categoría, entre otros; y también se puede realizar una serie de operaciones, denominados métodos, como ingresar dinero, retirar dinero, abonar intereses, consulta de saldo, transferir dinero, cierre de cuenta, entre otros métodos.

```
Class cuentaBancaria{

        private String nombre;

        privaye String nro_cuenta;

        private String categoría;

        private 6oublé saldo;

}
```


Previamente se mencionó que los atributos de un objeto de una clase se ocultan a los usuarios del mismo, por lo que tendrá que acceder a ellos por medio de los métodos que implemente la clase. Esta protección se consigue con el modificador *private*, con lo cual solamente podrá ser accesible mediante los métodos de su propia clase.

Se añade el método constructor de la clase.

```
Public cuentaBancaria(String nombre, String nro_cuenta, String categoría, double saldo){

        this.nombre = nombre;

        this. Nro_cuenta = nro_cuenta;
```

```
        this.categoría = categoría;
        this.saldo = saldo;
    }
```

Se utiliza el operador *this* para diferenciar entre atributos y parámetros, ya que como podemos ver los nombres de los parámetros son idénticos a los nombres de los atributos, el cual es una práctica común en la POO. El operador *this* se aplica a los atributos de la clase, por lo que estamos asignado el valor de los atributos serán los valores de los parámetros del método constructor.

Se agrega métodos que corresponden a la modificación de los valores de los atributos de la clase, comúnmente conocidos como métodos SET:

```
    public void setNombre(String nombre){
        if(nombre.length() = = 0)
        {
            System.out.println("Error, debe ingresar un valor");
        }
        this.nombre = nombre;
    }

    public void setNroCuenta(String nro_cuenta){
        if(nro_cuenta.length() = = 0)
        {
            System.out.println("Error, debe ingresar un valor");
        }
        this.nro_cuenta = nro_cuenta;
    }

    public void setCategoría(String categoría){
        if(categoría.length() = = 0)
        {
            System.out.println("Error, debe ingresar un valor");
        }
        this.categoría = categoría;
```

```
}

public void setSaldo(double saldo){
        if(saldo<0)
        {
                System.out.println("Error, debe ingresar un valor superior a 0");
        }
        this.saldo = saldo;
}
```

Los métodos SET tienen un parámetro del mismo tipo del método, es decir, el método setNombre tiene un parámetro nombre de tipo String que le corresponde al atributo nombre del mismo tipo String.

Tenga en cuenta que el método ha sido declarado como *public*, lo cual puede ser accesible para cualquier otra clase o subclase que necesite utilizarlo. Un método consta de su nombre precedido por el tipo de valor que devuelve cuando finalice su ejecución, a excepción de cuando se utilice la palabra reservada *void*, y seguido por parámetro.

Se agrega métodos que corresponden a la lectura de los valores de los atributos de la clase, comúnmente conocidos como métodos GET:

```
public String getNombre(){
        return nombre;
}
public String getNroCuenta(){
        return nro_cuenta;
}
public String getCategoría(){
        return categoría;
}
public 8oublé getSaldo(){
        return saldo;
}
```

Los métodos GET son del mismo tipo de sus atributos, es decir, el atributo nombre es de tipo String, entonces su respectivo método GET será de tipo String. Lo mismo aplica para los demás atributos.

Y otro método que permite ingresar valor al saldo de la cuenta.

```
Public void abonar(9oublé monto){
        saldo = saldo + monto;
        System.out.println("Se abonó a la cuenta bancaria");
}
```

En el siguiente método de retirar, primero se debe validar si existe saldo suficiente para realizar la operación.

```
Public void retirar(9oublé monto){
        if(saldo < monto)
        {
                System.out.println("Saldo insuficiente en la cuenta bancaria");
        }
        else
        {
                saldo = saldo – monto;
        }
}
```

Podemos seguir alimentando de métodos que la clase requiera. Ahora veremos cómo implementar el ejercicio con el uso de objetos. Para ello desde la clase principal *main* estructuramos el siguiente código.

```
Public static void main(String[] args){
        cuentaBancaria ctabanca01 = new cuentaBancaria("Ana López"," 747-2893872-87","Estándar",2409.60);

        ctabanca01.abonar(200.5);
        ctabanca01.retirar(180);
        System.out.println(ctabanca01.getSaldo());
}
```


El método *main* siempre se declara público y estático (static), no devuelve ningún resultado y tiene un parámetro args que es una matriz de dimensión de cadenas de caracteres.

La primera línea crea un objeto de la clase Cuenta y almacena una referencia al mismo en la variable ctabanca01. Esta variable se utiliza para acceder a ese objeto.

Las 3 líneas siguientes establecen un determinado estado para el objeto referenciado por ctbancaa01 enviándole los mensajes.

CONCLUSIONES

El uso de la Programación Orientada a Objetos (POO) es el paradigma de programación más usado en la actualidad, cuyo elemento fundamental son los objetos que interactúan con otros objetos para la transferencia de datos y su posterior procesamiento de los mismos.

Los objetos se basan en las clases que están compuestos por atributos que son las características de la clase y por los métodos que son los encargados de procesar los valores de los atributos, por ende todo lo que tiene una clase, se replica en los objetos que se derivan de esta clase.

Los valores que el usuario ingresa ya sea desde consola o desde alguna interfaz se almacenan en los atributos de los objetos y no en el de las clases. Podemos crear tantos objetos de una misma clase.

BIBLIOGRAFÍA

Ceballos Sierra, F. Programación orientada a objetos con C++. Rama Editorial, 2018. Digitalia, https://www.digitaliapublishing.com/a/110114

García y Beltrán, A. Programación con Java 7. Vision Libros, 2012. Digitalia, https://www.digitaliapublishing.com/a/88949

Moreno Pérez, J. Programación orientada a objetos (MF0227_3). Rama Editorial, 2016. Digitalia, https://www.digitaliapublishing.com/a/110026

Vegas Gertrudix, J. Java 17 Programación Avanzada. Rama Editorial, 2022. Digitalia, https://www.digitaliapublishing.com/a/116382