UNIDAD 6

INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

MÓDULO: Programación en Java

ÍNDICE DE CONTENIDOS

1.OBJETIVOS

- Entender uno de los principios del lenguaje Java, los objetos.
- Conocer la estructura, y sus posibles usos.
- Aprender a crear nuestros propios objetos, además de utilizar los predefinidos del lenguaje Java.
- Diferenciar los elementos de los objetos, métodos y propiedades.

2. PROGRAMACIÓN ORIENTADA A OBJETOS

La **POO** se basa en dividir el sistema en componentes que contienen operaciones y datos. Cada componente se denomina objeto.

Un objeto es una unidad que contiene datos y operaciones que operan sobre esos datos. Los objetos de un sistema se comunican entre sí mediante mensajes.

Beneficios de la POO

- **Modularidad.** El código fuente de un objeto puede mantenerse y reescribirse sin que ello implique la reprogramación del código de otros objetos de la aplicación.
- **Reutilización de código**. No hay que conocer los detalles de la implementación interna sino solamente su interfaz.
- **Ocultación de información**. En la POO se ocultan los detalles de implementación y lo que priva es la interfaz.

2. PROGRAMACIÓN ORIENTADA A OBJETOS

DIGI TECH
Centro Oficial FP Digital & Tech

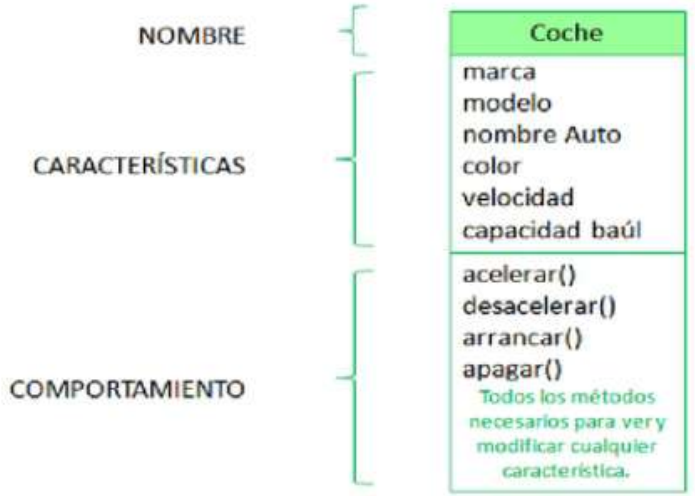
Características de la POO

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.1 Elementos de una clase

- Todos los objetos pertenecen a una clase.
- Una clase es un modelo o prototipo que define las variables y métodos comunes a todos los objetos de ciertas características comunes.

Contiene:

- **Conjunto de atributos comunes.** Datos que pueden ser tipos primitivos o bien objetos de otra clase.
- **Comportamiento por medio de métodos**.



2. PROGRAMACIÓN ORIENTADA A OBJETOS

2.1 Elementos de una clase

Declaración de una clase

```
class NombreClase{
        [atributos]
        [métodos]
}
```

Métodos

- Los métodos se pueden utilizar tanto para consultar información sobre el objeto como para modificar su estado.
- La información consultada del objeto se devuelve a través de lo que se conoce como **valor de retorno**, y la modificación del estado del objeto, es decir, de sus atributos, se hace mediante la **lista de parámetros**.

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.1 Elementos de una clase

ESTRUCTURA GENERAL DE UN MÉTODO JAVA

La estructura general de un método Java es la siguiente:

```
[control de acceso] tipoDevuelto nombreMetodo([lista parámetros]) [throws listaExcepciones]
{
   // instrucciones
  [return valor;]
}
```

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.2 Control de acceso

Control de acceso (opcional): determinan el tipo de acceso al método. Ej. protected (Acceso desde la clase y sus hijos "herencia).

Si el método no devuelve nada (tipoDevuelto = void) la instrucción return es opcional. Un método puede devolver un tipo primitivo, un array, un String o un objeto.

La instrucción return puede aparecer en cualquier lugar dentro del método, no tiene que estar necesariamente al final. **Ejem:**

```
public int obtenerValor(int parametro){
    // dentro de los paréntesis se declara una variable esta variable es el parámetro
  int x=parámetro/2;
      //utilizamos una variable x para calcular el parametro dividido entre 2
  return x;   //se obtiene finalmente la variable "parametro" dividida entre 2}
```

Visibilidad	public	protected	default	private
Desde la Misma Clase.	Sí	Sí	Sí	Sí
Desde cualquier clase en el mismo paquete.	Sí	Sí	Sí	NO
Desde cualquier clase fuera del paquete.	Sí	NO	NO	NO
Desde una subclase en el mismo paquete.	Sí	Sí	Sí	NO
Desde una subclase fuera del mismo paquete.	Sí	Sí	NO	NO

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.3 Métodos

En general, la lista de parámetros de un método se puede declarar de dos formas diferentes:

Por valor. El valor de los parámetros no se devuelve al finalizar el método, es decir, cualquier modificación que se haga en los parámetros no tendrá efecto una vez se salga del método. Esto es así porque cuando se llama al método desde cualquier parte del programa, dicho método recibe una copia de los argumentos, por tanto cualquier modificación que haga será sobre la copia, no sobre las variables originales.

Por referencia. La modificación en los valores de los parámetros sí tienen efecto tras la finalización del método. Cuando pasamos una variable a un método por referencia lo que estamos haciendo es pasar la dirección del dato en memoria, por tanto cualquier cambio en el dato seguirá modificado una vez que salgamos del método.

En Java, todas las variables de tipo primitivo se pasan por valor, los objetos se pasan por referencia.

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.3 Métodos

La declaración de un método tiene dos restricciones:

- **Un método siempre tiene que devolver un valor** (no hay valor por defecto). Este **valor de retorno** es el valor que devuelve el método cuando termina de ejecutarse, al método o programa que lo llamó. Puede ser un tipo primitivo, un tipo referenciado o bien el tipo void, que indica que el método no devuelve ningún valor.

- **Un método tiene un número fijo de argumentos.** Los argumentos son variables a través de las cuales se pasa información al método desde el lugar del que se llame, para que éste pueda utilizar dichos valores durante su ejecución. Los argumentos reciben el nombre de **parámetros** cuando aparecen en la declaración del método.

- **Métodos estáticos.** Los métodos estáticos son aquellos métodos definidos para una clase que se pueden usar directamente, sin necesidad de crear un objeto de dicha clase. También se llaman **métodos de clase**.

A diferencia de los métodos normales o métodos de instancia, los métodos de clase tienen la cláusula **static** y todos los objetos de la misma clase compartirán dichos miembros (variables globales).

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.3 Métodos

Dos aspectos característicos de utilizar el calificador static en un elemento Java son los siguientes:

- **No puede ser generada ninguna instancia** (uso de *new*) de un elemento *static* puesto que solo existe una instancia.
- Todos los elementos definidos dentro de una estructura *static* deben ser *static* ellos mismos, o bien, poseer una instancia ya definida para poder ser invocados.

NOTA: Lo anterior no implica que no puedan ser generadas instancias dentro de un elemento *static*; no es lo mismo llamar/invocar que crear/generar.

Los métodos de clase (estáticos) tienen estas limitaciones:

- No pueden acceder a campos de instancia (lógico, pues los campos van asociados a objetos).
- No pueden invocar a un método de instancia de la misma clase (lógico pues los métodos de instancia van asociados a objetos).

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.3 Métodos

Para llamar a un método estático se utiliza:

- El nombre del método, si lo llamamos desde la misma clase en la que se encuentra definido.
- El nombre de la clase, seguido por un punto, más el nombre del nombre del método estático, si lo llamamos desde una clase distinta a la que se encuentra definido.

```
public class metodo {


        public static void cambiar(int x){

                x=x+5;
                }
        public static int modificar (int x){
                int j=x+5;
                return j;
        }
        public static void cambiar2(int[] par){
                for (int i=0;i<3;i++){
                par[i]=par[i]+5;
                }
        }
        public static void main(String[] args) {
                int x=3;
                int[] arr={3,5,7};
                cambiar(x);
                System.out.println(x);
                int j=modificar(x);
                System.out.println(j);
                cambiar2(arr);
                for (int i=0;i<3;i++){
                System.out.print(arr[i]+"  ");
                }
        }
}
```

2. PROGRAMACIÓN ORIENTADA A OBJETOS

2.4 Métodos Get y Set

Normalmente un problema resuelto con la metodología de programación orientada a objetos no interviene una sola clase, sino que hay muchas clases que interactúan y se comunican.

Dentro de un programa los objetos se comunican llamando unos a otros a sus métodos.
No se deberá tener acceso directo a la estructura interna de las clases. El acceso a los atributos será a través de **getters** y **setters**.
Los setter y getters, son métodos de acceso en una clase, estos sirven para establecer y obtener datos de los atributos de nuestra clase, estos dos métodos deben ser públicos:

El método **set** (traducción al español: *"establecer"*): nos sirve para asignar un valor (inicializar) a un atributo de nuestra clase, esto se hace de manera directa con este método, como este método no retorna nada, el nombre de este debe ser precedido por void, y siempre debe recibir un parámetro de entrada.

2. PROGRAMACIÓN ORIENTADA A OBJETOS
2.4 Métodos Get y Set

El método **get** (traducción al español: *"obtener"*): este tipo de método accede a la clase para retornarnos el valor de algún atributo que queramos, el método get si debe retornar un valor por lo cual el nombre de este método debe ser precedido por el tipo de valor que vamos a retornar con ese método.

En general se aconseja declarar **todos los atributos como private**, y cuando necesitemos consultar su valor o modificarlo, **utilicemos los métodos get y set**.

RESUMEN DE LA UNIDAD

- Consiste en agrupar datos y métodos que operan sobre esos datos dentro de una clase, protegiendo el acceso directo a los datos y permitiendo la manipulación a través de métodos definidos.
- El trabajo con objetos nos permite que una clase (subclase) herede atributos y métodos de otra clase (superclase), facilitando la reutilización de código y la creación de jerarquías de clases que reflejan relaciones del mundo real .
- Los objetos de diferentes clases responder a la misma interfaz de manera distinta, proporcionando flexibilidad y capacidad para definir métodos que pueden ser usados de forma general en distintas clases.
- Los objetos permiten representar entidades del mundo real con sus atributos (campos) y comportamientos (métodos).

BIBLIOGRAFÍA/WEBGRAFÍA

- Thierry Richard. “Los fundamentos del lenguaje Java”.
- Vozmediano,A.M. “Java para novatos”.
- Jimenez, Alfonso.”Aprender a programar en Java”.
- Gervás, Luc. “Aprender los fundamentos del lenguaje Java”

RECURSOS DE INTERÉS

- **POO:** https://acesse.dev/hiaV4
- **Principios POO:** https://l1nq.com/2r7ov
- **Get y Set:** https://l1nq.com/mla5N
- **Métodos, objetos y clases:** https://acesse.dev/oOQVz