**Métodos de clase.**

Para recordar un Objeto es una instancia de nuestra clase.

Un Método es una función dentro de una clase.

En las Clases tenemos propiedades y métodos.

Clase A

Propiedad 1;

Propiedad 2;

Un método no es ni mas ni menos que una función dentro de una clase.

Según el lenguaje tendrán una firma o PROTOTIPO “ es la forma en que la declaramos.

Se define si es Pública o Privada. La public se puede utilizar de afuera de la clase.

La Private solo dentro de la clase.

Generalmente se declara Visibilidad (public o private) Nombre\_Metodo (Nombre del la función) parámetros y valor

En pseudo:

PUBLICA leer libros (Texto libro) Texto contenido.

En java:

Public void Acelerar (Int cuanta velocidad).

* Se puede acceder de cualquier lugar.
* No devuelve ningún valor.
* El nombre del método es “Acelerar”
* Recibe un dato de tipo entero.

Para acceder creamos un Objeto de nuestra clase

Ejemplo:

Coche coche = new Coche

Accederemos a través de la variable “coche”

Para invocar el método se usa el nombre de la variable. Método(par{ametros)

coche.Acelerar(50);

Puedo crear funciones dentro de MAin sin crear ninguna instancia u objeto.

**Interfaces.**

Se creo una interfaz (Vehículo ) con dos funiones.

Para poder utilizarlas creo una clase (Coche) que contenga esas funciones con los parámetros de la interfaz.

Luego se crea una función en el MAin cuyo parámetro es (Vehículo). Es decir le indico un parámetro que se llama como la interfaz porque le voy a exigir que el parámetro de ésta función sea una Clase que implemente esa interfaz.

La clase debe definir TODOS los Métodos de la interfaz.

Un ejemplo típico de interfaces.

INTERFAZ USUARIOS

METODO GETUSUARIO().

CLASE USUARIOSTXT IMPLEMENTA LA INTERFAZ USUARIOS.

METODO GETUSUARIOS()

LEER UN FICHERO .TXT

DEVOLVER TODAS LAS LINEAS

CLASE USUARIOSBDD IMPLEMENTA LA INTERFAZ USUARIOS

METODO GETUSUARIOS()

CONECTARME A MYSQL

EJECUTAR “SELECT” “FROM” ……..

DEVOLVER TODOS LOS RESULTADOS

FUNCION LISTAR USUARIOS (NECESITO INTERFAZ USUARIOS)

DE LA INTERFAZ USUARIOS EJECUTA EL METODO GETUSUARIOS.

INVOCO A LISTA DE USUARIOS (TE DOY OBJETO BDD)

La interfaz nos permite abstraer y simplificar

A través de una función envoltoria llamamos por debajo a una función real. El Código sel envoltorio (la interfaz) permanece inmutable. La función real la implementamos como queramos.

Lo que voy haciendo luego es ir implementando clases que satisfagan la interfaz.

**Parámetros de una función:**

Pueden ser de dos tipos.

1. Paso por valor. Consiste en que cuando llamo una función copio los valores en memoria y se los doy. Es lo que hemos estado haciendo. Ejemplo

FUNCION SUMA (INT A , INT B) {

DEVUELVE A + B

}

VAR VAL A = 5

VAR VALB = 10

SUMA (VAL A, VALB)

EN VALA se copia 5

EN VALB se copia 10

La desventaja del paso por valor es la cantidad de memoria que utilizo.

1. Paso por referencia. Le paso la dirección de memoria de las variables. Entonces no duplico capacidad de memoria.

En java cuando el parámetro de una función es un objeto se considera un paso de memoria por referencia.

Ejemplo

Coche coche new coche.

CocheChanger(coche)// por debajo hay un puntero

}

public static void cocheChanger (Coche coche){

coche.velocidad+=coche.velocidad +50

Coche coche hace referencia a un objeto. Por lo tanto manipulará la zona de memoria de ese objeto. Se habrá hecho un paso por referencia.

En otros lenguajes de programación los punteros se pasan explícitamente. No en Java.

En Go el puntero se indica con un asterisco antes de la variable.

\*int = 5.

**Recursividad**:

Consiste en que una función se va a llamar a si misma una y otra y otra vez.

FUNCION SUMA (INT A, INT B){

VAR TEMP = A + B

SUMA (A,TEMP)

} ES UN CICLO INFINITO.

Hay que saber cuando detenerla

Le puedo poner si temp mayor a 15 PARA

El ejemplo cláscico es el factorial.

Public static int factorial (int numero{

Int resultado;

If (numero ==1){

Return 1;

}

Resultado = factorial (numero:numero - 1) \* numero:

Return resultado

Tambien se podría hacer de forma no recursiva*. Ejemplo un bucle.*