POO  
Programacion Orientado a Objeto

Conceptos que veremos pero es mucho mejor que los lean antes ayudaran mas de lo creen.  
  
Clases se que para este te parece ilogico que te hable de esto si desde el comienzo con la programacion estructurada lo has visto pero no esta demas recordarte que las clases son los pilares de la poo ya que es aquí donde creamos nuestros obejtos , pero no sin olvidarnos de nuestro queridisimo metodo main (metodo principal) ya que este seguira teniendo recerpusion en esta nueva forma de programacion te reclaco despues de comprender, interpretar y tener el minimo conocimiento tendras aprender practicar crear o innovar no solo tenes que con esto el software y el desarrollo de este es un gran mundo y sigue avanzado estoy seguro que es posible que pierdas en el camino creeme valdra la pena sin mas que decir empecemos.  
  
Estas son los que definen los objetos a que refiero que en el momento que cremos un clase el objeto ya estara defino por medio de la clase.  
  
Pero que son los objetos no estoy seguro si para este punto ya lo mencione pero tu me querido lector es todo lo que esta. Tu alrededor y tenemos dos tipos aquellos que puedes ver tocar, los que solo viven en tu mente estos sin importar de que tipo sean tiene sus repstivos verbos y acciones no quiero profundizar mas pronto lo veras.  
  
Abstracciones estos se lo definen como la simplificacion de los objetos ya que como antes mencione trabajaremos con el mundo real, este nos permite modular sus verbos y acciones para poder tener una compresion de este y poder desarrollar codigo de calidad.  
  
Polimorfismo este nos dice que cada objeto pueda tomar un comportamiento diferente, este nos ayudara a reutilizar codigo y sobreescribir desde cualquier clase para darle el uso que esta desea.  
  
  
  
Herencia esta es un gran ayuda cuando tengamos muchas subclases, aquella nos dice que podemos heredar nuestros atributos y acciones a nuestras subclases conocidas como hijos o hijas.  
  
Encapsulamiento este nos ayudara a usar palabras reservadas como protected, private, public esto no dejara poner un limite a nuestras variables por ejemplo proteted nos dejara llamarla mientras nos encontremos en el mimso archivo, public la dejara publica para todos y private la encapsulara de tal forma como si no existira y la unica forma de llamarla sera por los getter pronto profundizaremos en aquello.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Por que es impresindible la programacion orienta a objectos?

Primero que es y porque necesitamos aprender poo este es un plus mas a nuestro pensamiento logico para poder resolver problemas de forma rapida. Y acertada sin tener que copiar codigo de otro lado aprenderemos conceptos que nos ayudaran mas tarde como.  
  
Encapsulamiento.  
Abstraccion.  
Herencia.  
Polimorfismo.  
  
Analizar problemas a un nivel mas explicito para poder dar un codigo de calidad en este camino cambiaremos nuestra forma de pensar nuevamente para poder hacer nuestro diagrama y despues escribir nuestro codigo pensando a futuro.   
  
recuerden esto:  
  
Analizar 🡪 Plasmar -> Programar.  
  
Este aparece por primera vez en 1960 con simula 67 donde se desarrollaron los prinicipios de esta pero no fue hasta 1970 que se le dio el termino que conocemos hoy en dia por Alan kay:  
  
**Clases:**  
plantillas que definen las propiedades y comportamiento de los objetos.  
  
**Objetos:**Instancias concretas de clases.  
  
  
**Herencia:**

Mecanismo que permite crear nuevas clases a partir de clases existentes, heredando sus características.

Este fue creado originalmente para realizar simulaciones este revoluciono la forma de pensar, modular y hacer un codigo mas facil de leer y mantener esto hasta hoy en dia sigue y es uno de los principales para paradigmas ya que este se utiliza en casi todos los lenguajes.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Que resuelve la poo?  
  
Esta es una excelente pregunta este resuelve muchos problemas de la programacion estructura y vacios que nos deja esta desde la repetencion de codigo, extremadamente largos, muy complejos para poder darle mantiemiento, lectura, si hago se daña todo se daña esto son algunos de los vacios de que deja la programacion estructura y el mas famoso de todos que es muy posible que lo tengas en tus archivos el famosisimo codigo espagueti esta es una gran pesadilla para quienes los intentan comprender y darle su respectivo mantemiento por lo tanto la POO resuelve todo estos problemos dandonos como resultado codigo mas modular con menos lineas como un pequeño dato hace algunos años o decadas los programadores cobraban por linea de codigo mucho despues descubrimos que mas no es mejor y podemos simplificar estos haciendolo mucho mas ordenador y compresible para poder entender, comprender y simplificar la programacion sin mas que decir sigamos.  
  
  
PARADIGMA POO  
  
Que es un paradigma?  
  
  
  
  
Que es Paradigma?  
Este por su concepto dado en la RAE nos dice que es un medio para la resolucion de un problema y exactamente como lo usaremos en la con POO este cuenta con 4 elementos que son:  
  
\*Clases:  
\*Propiedades:  
\*Metodos;  
\*Objectos   
  
Pero como estos elementos nos ayudaran en la resolucion de problemas dejen profundizar el paradigma nos expresa al mundo real con sus elementos es decir desde este preciso momento comenzamos a modular con el mundo real para poder para poder crear solucion en base a este.  
Bueno veamos y enfoquemonos como estos elementos antes mencionados nos llevaran al mundo real mi estimado estudiante preparese lo que se viene es chido desde ya suena muy esencial ☺.  
  
\***Clases:**  veamolas por el momento como planos o moldes que definen su caracteriticas y comportamientos comunes de un grupo de objectos o por otra parte comprendamolas con esta analogia son como los cimientos de las casas o edificios en base a estos se determinan las caracteristias de la casa o edificios ya sea longitud o anchura y altura es asi como las clases siendo las bases define a los objectos.  
  
**\*Objectos:** cada objecto es instancia de una clase o con una analogia si en la clase que son las bases damos unas bases para una casa en los objectos sacamos la casa ya que este esta defino por las clases.  
  
**\*Propiedades**: (atributos) estas definen el estado del objecto color marca etc.  
en nuestro caso la casa color rosa palo puertas verdes etc.  
  
  
**\*Metodos:** son las acciones o funciones que un objecto puede realizar define el comportamiento de un objecto.  
  
Que es modelado?  
Antes de pasar a un nuevo concepto o tecnologia debemos comprener que es modelado y para que nos sirve es posible que ya lo hallas usado, este es una forma de plasmar o explicar y analizar conceptos abstratos de forma simple donde tendremos una vista de este donde podremos ver sus problemas de una forma sencilla cualquier persona sin ser conocedores del tema podran comprenderlo no esta demas decir que no solo se usa para el software sino para la ciencia la matematica y la fisica.  
  
Que es UML (Unifed Modeling Language)?  
  
Este es un lenguaje estandar de modelado esto quiere decir que podemos expresar o plasmar de una manera grafica de representar problemas o situaciones como ya antes lo mencionamos.  
Debemos comprender, entender para poder usar el uml de la mejor forma a continuacion:  
  
  
  
  
  
  
  
🡪 Las clases se representan asi:  
En la parte superior se colocan los atributos o propiedades y debajo las operaciones de la clase, te daras cuenta que el primer carácter con el que empieza es un simbolo este nos dejara ver la visibilidad del atributo o metodo “este es un termino que tiene que ver encapsulamientos”.

Niveles visibilidad.

* - private
* + public \*
* # proteted

~default  
  
  
Formas de representar relaciones entre si veamos las mas comunes:  
  
Asociacion  
  
-------------🡪  
  
cada vez que veas esta fecha te daras significara que este elemento contiene a otro en su definicion.  
  
  
  
Herencia  
  
---------------|>  
  
Esta flecha nos dara conocer la herencia la dirrecion de esta ira desde el hijo al padre.  
  
  
Agregacion  
  
<>-----------

Un elemento dependera de muchos otros, con esto decimos que la claseA contiene varios elementos de claseB.

Composicion  
  
  
Este es muy similar al anterior pero con la diferencia que estas clases no pueden vivir sin el otro es decir si se elimina una clase a la cual relacionamos con la composicion la otra se va porque trabajan en conjunto.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Objetos  
  
  
Primero debemos saber como identificarlos como ya sabemos estos son defindos por las clases, estos llevan nombre de sustantivos son nombres de personas, animales o cosas con esta pequeña inroduccion te garantizo que podras encontrar tu objeto.  
  
Dentro de aquellos nos encontraremos con sus propiedades que son estas son descripciones o caracteristicas de este tambien conocidad como atributos digamos que tenemos un avion como objeto sus atributos seran tamaño, color, año, cuantos pasajeros, largo y ancho no esta demas decir que las caractersiticas tambien son considerados sustantivos.  
  
  
Tambien contamos con verbos que estos representan a las acciones o el comportamiento del objecto en nuestro caso tenemos un avion este vuela,aterriza y frena en algunos casos se estrella esto serian sus verbos.  
Por otro lado tenemos cuando una misma frase o palabra tiene un sustantivo y verbo esta las veremos como una accion no se me confundan mis queridos estudiantes igual no esta demas decirle que debena analizar su problema muy bien.  
  
Sin mas que decir debemos definir que es lo que hara nuestro objeto este es el mundo real o las varias tareas que cumplira.

Modularidad  
  
  
Este es un viejo conocido pero muy util este ayuda en muchos ambitos al escribir codigo entre ellos la repeticion de este la reutilizacion, una mayor legibilidad, menor riesgo de errores, codigo listo para escalar es decir hacer crecer tu codigo a futuro esto va de la mano con las funciones que nos dicen en sus coneptos que separamos nuestros codigos en parte y es asi como logramos que cualquiera pueda leerlo sin importar muchas y podermos modificarlo a futuro de forma muy rapida y simple por tal te recomiendo que uses las modularidad veamolo de la siguiente un lego encima de otro y asi no hay un limite, algo que acotar es podemos poner cada class en distintos archivos y despues conectarlas para no tener asi un codigo encima de otro que sea incompresible esto nace a lo ya conocidos codigos espaguetis que creeme fueron y siguen siendo la pesadilla de mucho ya que es muy posible que tu como junior o estudiante de recien ingreso hicieras tu primer codigo de esta forma sin mas que decir y aportar les dejo ejemplos de la modularidad para que puedan aplicarlos en sus proyectos futuros.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
ANAliZAR UNA PROBLEMA SACARLE SUS REPESTIVOS OBJETOS   
  
Para este punto ya comprendemos los verbos y sustantivos en la poo por tal ire directamente al grano te recomiendo que te hagas un ejercicio igual.  
  
En nuetro caso sacaremos los objetos de un mecanico.  
  
Estamos en el taller de un mecanico el se dedica hace 20 años a esta profesion muy buena y dura le entra una llamada de un cliente con su carro dañado. A unos 10 minutos de su ubicación sin mas el comenta el precio y pregunta por el año del carro sus motor color , cuantas puertas tiene y como fue que se termino apagando sin motivo alguno.   
  
El mecanino se dirige a la ubicación de este user y remolca el carro aclarando que cobrara lo de la grua en la factura final.Despues de tres dias laborales este tiene el auto como nuevo corriendo, frenado, sonando como en su epocas de gloria cabe recalcar que este es un carro del los años 80s de 4 puertas de color negro. Muy bien conservado al igual que su dueño extremadamente joven aunque no parece para su edad parece de mucho menos uno 20 años pero se ve de 17 sin mas que aportar le doy mi factura que son de 230$ por todo el servicio automotriz dado al cliente, soy una persona o personaje muy razonable. En los precios para mis competidores.  
  
  
\*Obteniendo sus obejtos   
  
.- Sustantivos   
taller mecanico   
mecanico   
cliente = muy joven  
carro = color negro = estado muy bien conservado = puertas 4 puertas   
grua  
  
  
.- Verbo  
  
mecanico hace el viaje a donde el user para remolcar el carro  
mecanico dice el precio de la grua al final   
carro acciones frenar, imovil, corriendo  
da su factura entre el carro  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Sintaxis de UML  
  
Conocer la sintaxis de este es grandioso para poder saber como ubicar nuestro objetos en nuestros lenguajes de programacion veremos esto en 4 distintos codigos nunca esta demas saber uno poco de cada uno yo personamente aprendi a programar en java.  
  
Que es lo haremos drectamente como definir las clases y los objetos como sabemos dentro de una clase podemo agregar muchos objetos cada uno con diferentes tareas y tipos   
  
No les puedo mostrar graficamente pero imagineselo como un sustantivo conceptual por lo tanto describire tenemos un cuadro donde en el titulo ponemos la identidad o nombre de la clase despue de esto , el estado de esta y por ultimo el comportamiento.  
  
Ahora veamolo directamente en cuatro lenguaje Java, Phyton , Javascript y Php en todas la sintaxis sera diferente:  
  
  
  
  
  
  
  
  
  
  
  
  
Algo que acotar aquí es que cada lenguaje tiene un sintaxis distintas como vemos en java no se declara igual que en Php   
Java como saben es un lenguaje hecho para la poo, Python es mas de para data scientist la web es un lenguaje muy facil de comprender entender y explicar, javascript esta en varios lugares desde el backend, frotend que es su fuerte, app moviles, data scienist, videojuegos etc.  
Php para la web este es directamente interperetado para esta, servidor, base de datos.

import *java.util.ArrayList*;

import *java.util.Scanner*;

import *java.util.Random*;

*public* *class* EjerciciosIntemedio{

*public* *static* *ArrayList*<*Integer*> CreamosElArregloEnBaseAlUsuario(*int* *numberEntero*){

*ArrayList*<*Integer*> numerosTest = new *ArrayList*<>();

*ArrayList*<*Integer*> numerosOrden = new *ArrayList*<>();

for(*int* i = 1; i <= numberEntero; i++){

numerosTest.add(i);

}

return numerosTest;

}

*public* *static* *void* OrdenamientoFormaDesendentesAcedente(*ArrayList*<*Integer*>*procesoLin*){

*Random* numberTe = new Random();

*int* numberT12 = numberTe.nextInt(100);

*Scanner* escuchandoTEL = new Scanner(System.in);

*ArrayList*<*Integer*> NumberWoo = new *ArrayList*<>();

for(*int* ii = 0; ii < procesoLin.size(); ii++){

NumberWoo.add(procesoLin.get(ii));

}

for(*int* iv = 0; iv < numberT12; iv ++){

NumberWoo.add(iv);

}

do {

System.out.println("buenas usted desea ordenar en decendiente o ascendente y por medio de los algoritmos burbuja o inserción recuerde solo puedo elegir uno a uno");

System.out.println("Ejemplo: ascendente, burbuja");

*String* usuarioOne = escuchandoTEL.nextLine();

*String*[] comprobandoUser = usuarioOne.split(",");

*// for(int i =0; i < comprobandoUser.length; i++){*

*boolean* preguntaOne = (comprobandoUser.length >= 2) ? true : false;

*String* userOne1;

*String* userOne2;

if(preguntaOne == true ){

userOne1 = comprobandoUser[0].trim();

userOne2 = comprobandoUser[1].trim();

}else{

continue;

}

if(userOne1.equals("decendiente") && use rOne2.equals("burbuja")){

System.out.println("usted selecciono decendiente y burbuja");

for(*int* i = 0; i < NumberWoo.size(); i++){

for(*int* iv = NumberWoo.size(); iv >= 0; i--){

if(NumberWoo.get(i) < NumberWoo.get(iv)){

numerosOrden.add(NumberWoo.get(i));

}else{

NumberWoo.set( NumberWoo.get(i), NumberWoo.get(i));

continue;

}

}

}

}else if(userOne1.equals("descendente") && userOne2.equals("insercion")){

System.out.println("usted selecciono decendiente y insercion");

}else if(userOne1.equals("ascendente") && userOne2.equals("burbuja")){

System.out.println("usted selecciono ascendente y burbuja");

}else{

System.out.println("usted selecciono ascendente y insercion");

}

} while (true);

*// System.out.println("usted a terminado este programa con exito muchas gracias por contribuir en el aprendizaje");*

}

*public* *static* *void* main(*String*[] *args*){

*// Descripción: Implementa una función que ordene un arreglo de números enteros de forma ascendente utilizando el algoritmo de burbuja o inserción.*

*// Desafíos adicionales:*

*// Implementa una función que ordene un arreglo de objetos personalizados (por ejemplo, una clase Persona con atributos nombre y edad).*

*// Permite al usuario elegir el tipo de ordenamiento (ascendente o descendente).*

*// Mide el tiempo de ejecución de cada algoritmo para comparar su eficiencia.*

*Scanner* enterosHumano = new Scanner(System.in);

System.out.println("ingrese un numero entero para poder crear un arreglo de numeros");

*int* numberEntero = enterosHumano.nextInt();

*ArrayList*<*Integer*> procesoLin = CreamosElArregloEnBaseAlUsuario( numberEntero);

System.out.println(procesoLin);

Php  
  
class Person{}

Python  
  
Class Person:

*Java*

*class* Person{}

JavaScript  
  
Function Person() {}

Java  
Class Person{  
  
String name = “testOne”;  
  
void walk{} []  
  
}  
java es muy estricto en como declaramos por lo tanto ojo con las mayusculas minusculas y espacios

Python  
Class Person{  
  
 name = “ ”;  
 def walk();  
  
  
  
  
}

Php  
Class Person{  
  
$name =” ”;  
  
function walk() []  
  
  
  
  
}

Javascript  
  
Person.prototype.walk= function{} []

HERENCIA  
  
Esta cuenta con una filosofia algo particular pero acertada “Don’t repeat yourseft” esta nos dice que no debemos tener repeticiones los atributos o verbos de nuestros objectos por tal nace esta que nos ayudara a eliminar esto con lo que conoceremos como la “SUPERCLASE” aquella lleva la gerarquia padre dando lugar como “SUPERCLASE” e hijo como “subclases” los elementos que tendremos en la clase padre podremos decir cuales heredarles a las diferentes subclases.  
  
Caracteristicas:  
  
  
\*Reutilizacion del codigo.  
  
\*No repetir codigo  
  
\*Organización de codigo  
  
\*Polimorfismo  
  
Este ultimo que es y que tiene que ver con la herencia este nos ayuda de tal manera que podemos tratar a varios objetos de diferentes clases como si fueran uno solo claro no esta de mas recordar que tiene que estar relacionado con la herencia.

-->Imagina que nuestro sistema de adopciones creció y ahora ofrece adoptar pericos, loros, gatos y hamsters.

Genera un nuevo análisis, aplica herencia para abstraer mejor el problema y lograr modularidad en el software.Comparte tus resultados en la sección de discusiones.Imagina que nuestro sistema de adopciones creció y ahora ofrece adoptar pericos, loros, gatos y hamsters.Genera un nuevo análisis, aplica herencia para abstraer mejor el problema y lograr modularidad en el software.

USER  
  
  
+ String name  
+ String id  
+ String edad  
+ String email  
# String  
  
adopter   
pago  
viaja(ir al centro de adopcion)

Centro de adoption  
  
String name  
String color  
String dirrecion  
String edad  
String color  
  
   
facture

Animales C.A  
  
  
id   
String name  
String raza  
String color  
String estatura  
String edad  
  
  
  
Cariñoso   
loco  
tranquilo

Timido

hamsters

Gato

Loros

Pericos

Objeto Metodo Constructor y su respetiva sintaxis  
  
  
Primero le debo una disculpa pero se me lo terminaras agredeciendo hasta el momento hemos visto fragmentos de 4 lenguajes de programacion se que puede llegar a mariar un poco pero creamen nunca esta mal saber un poco de todo sin mas.  
Recordemos. Como ya sabemos los objetos son definidos por las clases y estas son conocidas como superclases.  
  
  
Que es el metodo constructor?  
  
Este es un metodo que va de la mano con la poo es fundamental para que logre funcionar tal y como lo conocemos, este se ejecuta de forma automatica cuando se crea una clase y sus respetivos objetos este es el encargado de inicializar las propiedades de la clase tanto para argumentarla y pasarla a subclases pero no solo se encarga de esta tarea podemos hacer validaciones de datos, realizacion de calculos iniciales, creacion de objetos relacionales se que dicho temas nuevos pero pronto profundizaremos por lo momento te en cuenta que el constructor es un honor por donde se procesa la comida (propiedades) para salir al cliente final.

Sintaxis en codigo  
  
Como ya lo menciones y te has estado dando cuenta hemos visto 4 tipos lenguajes, su respetiva sintaxis:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Esta es la sintaxis para crear un constructor en los cuatros distintos lenguajes como nos damos cuenta todas tiene elementos parecidos desde la palabra new hasta “()” que en muchos lenguajes nos dicen o hace referencia a metodos podemos empezar un constructor sin ningun valor desde clase main o dependiendo de nuestro requerimietos con valores.  
  
  
  
  
  
  
  
  
  
  
  
DECLARANDO METODO CONSTRUCTOR EN NUESTRO 4 LENGUAJES Y CLASES  
  
  
Como ya sabemos debemos incializar nuestro constructor en la clase main que es el metodo de entrada principal o al español “metodo principal” nuestro metodo cumple la misma funcion en los 4 distintos lenguajes

Python  
  
person = Person()

Java

Person person = new Person

PHP  
  
  
  
  
$person = new Person();

JavaScript  
  
  
  
  
var person = new Person();

*public* *class* main{

*public* *static* *void* main(*String*[] *args*) {

*Car* car = new Car("cd13dd", 1, new Account("lucas ", "11cvw2", "bet@gmail.com"));

*Account* account = new Account("LUCAS", "123", "TEN1SX", "SOUP12@GMAIL.COM", "1234DAD4");

}

}

Como se daran cuenta tenemos dos constructores con datos declarados en el main en JAVA.

Mostrar en python y java en construccion

HERENCIA EN TODOS NUESTROS LENGUAJES

Comencemos con nuestro lenguaje de favorito en mi caso java  
  
Para poder declarar la herencia en nuestro lenguaje lo hacemos con nuestra palabra extends y el nombre de la clase padre a continuacion la clase padre.  
  
  
  
Class nombredetusuSubclase extends nombredelaclasePadre quedaria de la siguiente forma:

*class* UberX *extends* *Car*{

*String* model;

*String* marca;

*public* UberX(*String* *license*, *String* *passegenger*, *String* *id* , *String* *model*, *String* *marca*, *Account* *driver* ){

*super*(license, passegenger, id, model, marca, driver);

*this*.model = model;

*this*.marca = marca;

}

}

En esta tenemos nuestra subclase UberX y definimos sus atributos nuevos y hacemos la llamada de las propiedades desde la clase padre dentro de este y recalcamos con la palabra super.  
Por ultimo definimos con la palabra this las nuevas propiedades de esta subclase aquí le hago una pregunta Ustedes creen que podamos heredar o pasar estos nuevos atributos a la Family ()? Es una nueva interrogante muy buena pero creamen si es posible pronto profundizare en esto pero piensen como lo haran les dare los conceptos necesarios para poder hacer este reto o poder tener una idea clara sin sigamos.

En javascript tenemos la siguiente sintaxis.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
como ya sabemos js es uno de los lenguajes que funciona de una forma distinta con los propotipos pero eso ya lo sabemos esta es un de las formas mas nuevas de hacerlo y como nos damos cuenta la palabra this. Y le constructor sigue siendo lo mimso.  
  
  
  
  
En python

*class* user:

id = *str*

name = *str*

edad = *int*

address = *str*

document = *str*

def\_\_init\_\_(id, name, edad, address, document){

*self*.name = name

*self*.edad = edad

*self*.address = address

*self*.document = document

}

*class* Employt{

*constructor*(*id*, *name*, *identification*, *age*){

*this*.id = id;

*this*.name = name;

*this*.identification= identification;

*this*.age = age;

}

}

= $address

PHP  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
La herencia en pocas palabras no ayudara reutilizar codigo y hacer nuestro codigo mas flexible a cambios y legible sin mas que decir les dejo no olviden practicar y perderse en el camino.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Metodos locales y globales   
  
  
Se que muchos para este punto ya saben sus definiciones si es asi simplemente avance al siguiente.  
  
  
locales   
  
Son metodos o tributos que se encuentran dentro de las subclases y estas solo tienen el alcance dentro de estas nomas alla es decir no pueden ser llamados por otros hijos o clases.  
  
  
  
Globales   
  
Son metodos declarados o atributos en la clase padre que le podemos llamar o heredar a nuestros hijos es decir podemos llamarlo desde cualquier parte del codigo sin mas.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Herencias

class user{

public function \_\_construct($id, $address, $name, $document){

$this->id = $id;

$this->address = $address;

$this->name = $name;

$this->document = $document;

}

}

Se que para este momento ya hemos hablado de aquella pero les garantizo que es la punta del todo ya desde ahora ya nos referiremos a esta como superclase o subclases digamelo family, esto se debe a aquellos imagenos que tenemos un hijo y nosotro somos su padre pero con el tiempo este hijo tiene su hijo yo pasaria hacer el abuelo y el nuevo padre es por tal que. Le llamaremos familia de ahora en adelante example.

*public* *class* Main {

*public* *static* *void* main(*String*[] *args*){

*SmartPhone* smartPhone1 = new SmartPhone(12.4, "blue", 199.99, 12.4);

smartPhone1.printSmartPhone();

*// System.out.println();*

}

}

Tenemos mucha clase main donde como ya sabemos esta es la entrada al programa es lo primero en ejecutarse , eh aquí donde declaramos. Nuestros constructor con su respectivos valores y hacemos el llamado al metodo void declarado en nuestra (superclase o pronto abuelo)

*class* SmartPhone{

*Double* weight;

*String* color;

*Double* price;

*Double* measures;

*public* SmartPhone(*Double* *weight*, *String* *color*, *Double* *price*, *Double* *measures* ){

*this*.weight = weight;

*this*.color = color;

*this*.price = price;

*this*.measures = measures;

}

*public* *void* printSmartPhone() {

System.out.println("the weigth " +weight);

System.out.println("the color is " +color);

System.out.println("the price " +price);

System.out.println("the measures " +measures);

}

}

Esta es nuestro abuelo o por el momento padre donde claramos y de definimos nuestros atributos mas abajo lo haremos pero por el momento centremonos en su respestiva sintaxis

Encapsulamiento  
  
  
Este nos ayudara reutulizar codigo y encontraremos metodo de gran ayudara para llevar un orden en nuestros programas

*class* Telefono *extends* *SmartPhone*{

*public* Telefono(*Double* *weigth*, *String* *color*, *Double* *measure*, *Double* *price* ){

*super*(weigth, color, measure, price);

}

}

Este es su hijo donde usamos la palabra clave “EXTENDS”

*class* Tablet *extends* *Telefono*{

*public* Tablet(*Double* *weigth*, *String* *color*, *Double* *measure*, *Double* *price* ){

*super*(weigth, color, measure, price);

}

Si pones atencion te daras cuenta que esta subclase se hereda de una subclase se que vas decir como es posible este es el hijo del hijo del padre es decir el ahora es antiguo padre es abuelo. Y hijo es padre del nuevo hijo

Ahora una vez entendido y comprendido esto podremos seguir recuerdas que dijo que tenemos distintos tipos de herencia simple, jerarquica, , multiple y hibrida.  
 Veamolas mas a fondo para poder comprender sus caracteristicas.  
  
  
  
  
  
  
--> Herencia simple por su nombre ya nos hace referencia que seguro sera compresible. E indentificable a plena vista esta es una de las formas basicas de la herencia esta se caracteriza por que una clase solo puede herederar de una superclase es decir no tiene mas de una superclase.  
example:

//el metodo main estara fuera de este ejemplo debido a que ya sabemos eso   
  
  
class padre{  
  
String id;  
String name;  
String age;  
String measures;  
  
  
  
  
  
  
}

🡪 Herencia jerarquica esta nos dice que una clase padre puede tener muchos hijos creando una especie de arbol digamele la conocida como familia example:

class Abuelo{  
  
String age;  
String measure;  
String stature;  
String name;  
String id;  
  
 public Abuelo(String name, String age){  
 this.name = name;  
 this.age = age;  
  
}  
  
  
  
}  
  
//subclase Padre   
  
class Padre extends Abuelo{

public Padre(String name, String age, String id){  
 super(name, age, id);  
}  
  
}  
  
  
class Hijo extends Padre{  
  
   
   
 public Hijo(Stirng id, String name, String age){  
  
 super(id, name, age);  
}  
  
  
  
}

🡪Herencia Multiple:  
Esta no es soportada en java pero es posible hacerla, esto consiste en que una subclase puede heredar atributos o propiedades de mas de una superclase.

Class AmBerC{  
  
  
  
  
  
  
  
  
}  
  
  
  
  
  
Class UmberC{  
  
  
  
  
  
  
Class HeredatorTwo implements. AmberC , UmberC{  
  
  
  
  
  
}

Porque no es soportado por java esto se debe a problemas de ambigüedad y complejidad pero que quiero decir con estos tecnisimos primero esto se conoce como el problema del diamante , simplemente veamos el ejemplo expuesto y que la clase heredatorTwo despues de heredar a las dos clases hacemos un llamado para que muestro sus datos(propiedades )y (acciones) heredados pero cual de los dos mostrara en otros lenguajes como C++ que manejan y permiten esta herencia tiene un manejo especial ya que esto puedo causar mucho errores.   
Pero entonces como manejamos o podemos ejecutar la herencia multiple en java simplemente este lo prohibe , sin embargo es posible hacer por medio de interfaces pero que son estas y por que es posible. Primero debemos comprender que las interfaces son capacez de usar o heredar mas de una superclase debido a que estas tiene un base pero esta no nos limita a como usarlo no nos dice que asi se usa por lo tanto podemos darle un uso exacto para heredar mas de una. Esto es posible gracias a que las interfaces son lo que son debido a que no tiene. Un metodo o varios al iniciarse o implementarse estas inician con metodo abstractos.  
  
  
  
🡪Herencia hibrida esta es aquella que une la herencia simple y la multiple para usuarios de java interfaces esta parece ser compleja pero creeme no lo es tanto.  
  
example:

*class* HerenciHibrida {

*String* id;

*String* wigth;

*String* age;

*String* color;

*public* HerenciHibrida(*String* *id*, *String* *wigth*){

*this*.id = id;

*this*.wigth = wigth;

}

}

*Class* *PruebaOne* extends HerenciHibrida{

*public* PruebaOne(*String* id, *String* name, *String* age){

*super*(id, name, age)

}

}

*/\*\**

*e PruebaTwo*

*\*/*

*public* *interface* PruebaTwo {

*void* Hablar();

*void* Caminar();

}

*/\*\**

*\* InnerHerenciHibrida*

*\*/*

*public* *class* InnerHerenciHibrida *extends* *HerenciHibrida* *implements* *PruebaTwo* {

}

Encapsulamiento  
  
Este tiene multiples beneficios al momento de usarlo primero tendremos en cuenta que este nos ofrece palabras reservadas para poder encapsular son proteted, public y privated.  
Estas palabra son las que nos ayudaran hacer nuestros codigos mas legibles para poder simplemente enfocarnos en el comportamiento de los objetos en lugar de sus implementaciones internas veamos sus respestivas caracteristicas de cada palabra.  
  
  
🡪 Public este como no da entender la palabra y comprender el atributo o metodo que le dejemos con public sera publico para todos y podra ser llamado por cualquier clase.  
  
  
🡪 Proteted este no dejara encapsular nuestras varibles pero nos dice que solo se podran llamar desde el mismo archivo  
  
🡪 Private este nos dice que una vez encapsulada en este metodo quedara como si no exitiera y la unica forma de llamar este metood sera por medio de get pero son estos:  
  
  
  
public void getNombreDelAtributoo() {  
 estos son los get que no ayudaran a traer el valor de atributo o accion   
}  
Y los set son aquellos que nos ayudaran a cambiar o sobreescribir el valor de estos atributos   
  
public void setNombreDelAtributo( String NombreDelAtributo). {  
 this. NombreDelAtributo = newNombreDelAtributo;  
}  
  
  
  
  
 Polimorfismo  
  
  
  
Este nos ayudara a reutilizar codigos para poder usarlos. Y modificarlos en las diferentes subclases es decir sobreescribirlo para poder usarloa. Nuestro gusto en cada hijo o hija con su respetivo metodo veamos un ejemplo.