

EXAMEN DE EVALUACIÓN PARA DEVELOPERS

Esta evaluación consiste en la resolución de un problema dividido en 6 secciones. La sección cero consiste en analizar los conceptos básicos de la POO.

La primera sección consiste en el diseño de arquitectura de una base de datos en base a un requerimiento.

La segunda sección consta de poder generar querys para extraer información y resolver unos ejercicios de consulta.

**INSTRUCCIONES**:

Coloca tu nombre y fecha sobre las hojas sobre las cuales vayas a contestar, se sugiere

**letra legible** en las respuestas y no contestes en esta hoja.

# SECCIÓN 0

1.- Menciona los 4 pilares de la programación orientada a objetos.

* Herencia: La herencia es la forma en que nosotros reutilizamos código de una clase, haciendo que otra herede sus atributos y métodos para poder ser usados y ahorrar el escribir lo mismo.
* Polimorfismo: Lo conozco como la forma en que algo tiene diferentes formas, la forma más común de polimorfismo es en los métodos, donde se llaman varios de la misma forma, pero recibiendo diferentes atributos.
* Encapsulamiento: El encapsulamiento es una forma de
* Abstracción: La abstracción es como el arte, tiene una forma predeterminada dependiendo la persona que la vea, generalmente se usa más en interfaces donde se implementa en una clase y se hace uso de sus métodos dando forma a lo que se necesita en esa clase.

2.- Realiza un ejemplo de la implementación de una interfaz.(Ayuda: Persona tiene vida.).

public interface Animal{

public void vivo(boolean vivo);

}

public class Perro implements Animal{

public void vivo(boolean vivo){

System.out.println(vivo);

}

}

public class Hamster implements Animal{

public void vivo(boolean vivo){

System.out.println(vivo);

}

}

3.- Realiza un ejemplo de una clase abstracta (Ayuda: Hombre, Mujer, Persona, sexo).

abstract class Persona{

private String sexo;

public Persona( String sexo) {

this.sexo = sexo;

}

public String gender(){

return sexo;

}

//getter y setter para sexo

}

Public class Hombre extends Persona{

public Hombre(String sexo) {

super(sexo);

}

@Override

public String gender() {

return super.gender();

}

}

public class Main {

public static void main(String[] args) {

Hombre hombre = new Hombre(“Masculino”);

System.out.println(hombre.gender());

//Instanciamos Hombre y no Persona porque Persona al ser clase abstracta sólo funciona como una plantilla y no nos deja instanciarla para hacer uso de sus métodos.

}

}

4.- ¿Cómo crearías la abstracción de un objeto ***Perro*** que tiene, *años*, *género*, *tamaño* y

*nombre*?.

Un perro es un animal, un animal tiene lo mismo que el perro, años, género, tamaño y nombre, abstraerlo como un animal haría que al hacer uso de herencia en la clase Perro este pueda tener todas esas características y métodos extras como ladrar, sentarse y que otras clases parecidas puedan hacer uso de esta como un gato, hámster, etc.

abstract class Animal {

private int age;

private String name;

private String gender;

private String size;

public Animal(int age, String name, String gender, String size){

this.age = age;

this.name = name;

this.gender = gender;

this.size = size;

}

}

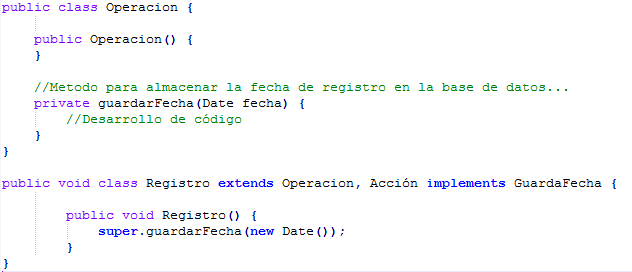
Después de esto crearía la clase Perro, heredo de animal e implemento los métodos específicos de Perro.

5.- ¿Qué función tiene la palabra reservada **this** y **super**?**.**

This tiene la función de hacer referencia a un elemento dentro de la misma clase, cuando pasamos un atributo a un método con el mismo nombre que tiene un atributo en la clase usamos this para hacer referencia que usaremos el de la clase.

Super tiene la función de implementar el método del padre cuando hacemos uso de herencia tal cual está en la clase padre.

6.- Encuentra los 4 errores en el siguiente código de ejemplo (Ayuda considera que Accion y GuardaFecha sí existen):



1.- El primer error que veo es en el método guardarFecha, pues no tiene un retorno

2.- El segundo error que veo es que en registro está mal estructurada la clase, empieza con public void class, cuando debería ser solo public class Registro

3.- El tercer error que veo es que el constructor de Registro tiene un tipo de retorno void cuando no debería de tener ningún retorno

4.- El cuarto error que veo es que se está intentando usar herencia múltiple lo cual en java no es posible, interfaces múltiples sí, pero clases no.  
5.- Un error que pudiese estar presente es que en la clase Registro se implementa la interfaz GuardaFecha pero en ningún momento se manda a llamar sus métodos en la clase Registro

6.- Un error que es más de escritura es que la clase Acción tiene un acento, lo cuál al momento de compilar puede traer errores

# SECCIÓN 1

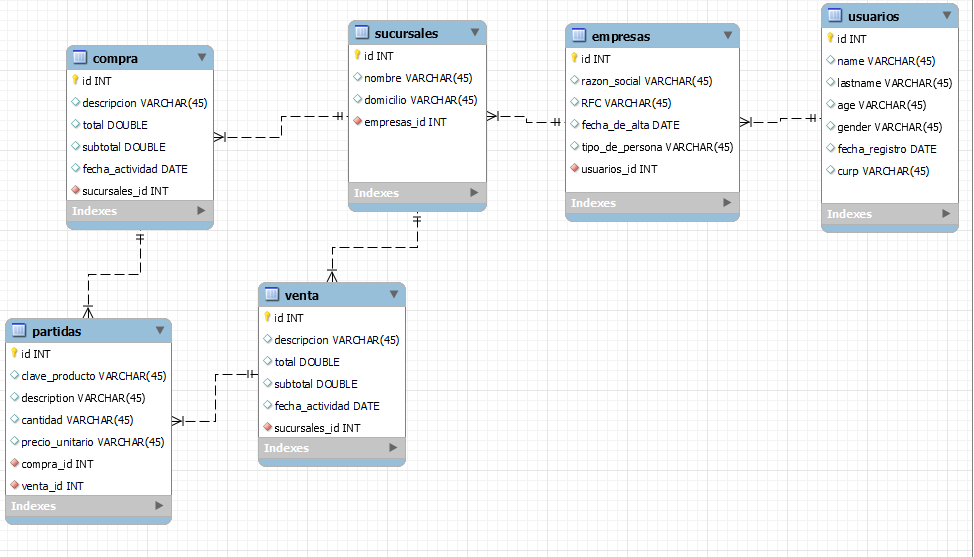
Se requiere la arquitectura relacional básica de un sistema que maneja múltiples usuarios con su respectiva información que debe cumplir con los siguientes requisitos:

* Un usuario puede tener varias empresas.
* Cada empresa puede tener muchas sucursales.
* Cada una de las sucursales tiene actividades tanto de compra como de venta.
* Para las compras y las ventas, se requiere registrar sus partidas (representación desglosada de la actividad realizada).

Hint\* La información básica con la que se debe contar es la siguiente:

* Para usuarios: Nombre, apellidos, edad, sexo, fecha registro, curp.
* Para empresas: Razón social, RFC, fecha de alta, tipo de persona (física o moral).
* Para sucursal: Nombre, domicilio.
* Para compra y venta: Descripción, total, subtotal, fecha de actividad.
* Para partidas: Clave de producto, descripción, cantidad, precio unitario.

Actividad:

Diseña un modelo relacional para la arquitectura de la base de datos que cubra las necesidades mostradas en el planteamiento. 

Los requisitos que se piden es que el usuario puede tener varias empresas entonces la tabla empresas posee el FK de usuario para poder relacionar a que usuario le pertenece (1:n), lo mismo con sucursales, tiene el FK de empresas porque necesitamos saber a que empresa le pertenece(1:n), las sucursales compran producto y a la vez venden un producto, la compra y venta necesita saber que sucursal es la que hizo la compra o venta(1:n), las partidas es como el corte de caja, entonces se necesita el FK de compra y venta para poder hacer el corte(1:n).

# SECCIÓN 2

En base al diseño realizado en la sección 1, se desea preparar querys que resuelvan las siguientes situaciones:

* Extraer todos los nombres de las sucursales de un usuario.

SELECT sucursales.nombre FROM usuarios

JOIN empresas ON usuarios.id = empresas.usuarios\_id

JOIN sucursales ON empresas.id = sucursales.empresas\_id

WHERE usuarios.id = “Aquí el id sin las comillas”;

* Conocer el nombre de las empresas que venden un producto con clave : **1234.**

SELECT DISTINCT empresas.razon\_social FROM empresas

JOIN sucursales ON empresas.id = sucursales.empresas\_id

JOIN venta ON sucursales.id = venta.sucursales\_id

JOIN partidas ON venta.id = partidas.venta\_id

WHERE partidas.clave\_producto = '1234';

* Conocer el ingreso (ventas) de las sucursales de el usuario con curp: **A1A1A** en el periodo comprendido del 15 de Enero al 15 de Febrero**.**

SELECT s.nombre AS sucursal, SUM(v.total) AS ingreso\_ventas

FROM usuarios u

JOIN empresas e ON u.id = e.usuarios\_id

JOIN sucursales s ON e.id = s.empresas\_id

JOIN venta v ON s.id = v.sucursales\_id

JOIN partidas p ON v.id = p.venta\_id

WHERE u.curp = 'A1A1A' AND v.fecha\_actividad >= '2023-01-15' AND v.fecha\_actividad <= '2023-02-15'

GROUP BY s.nombre;

* Conocer los gastos de la empresa con RFC: **XAXX010101000** en el mes de Octubre del 2018.

SELECT e.razon\_social AS empresa, SUM(c.total) AS total\_gastos

FROM empresas e

JOIN sucursales s ON e.id = s.empresas\_id

JOIN compra c ON s.id = c.sucursales\_id

JOIN partidas p ON c.id = p.compra\_id

WHERE e.RFC = 'XAXX010101000' AND MONTH(c.fecha\_actividad) = 10 AND YEAR(c.fecha\_actividad) = 2018

GROUP BY e.razon\_social;