**Prueba #1 Conceptos de programación**

**1. (1 Pto) ¿En programación orientada a objetos (POO), a que se refiere el término clase?**

es la descripción de un **conjunto de objetos** similares; consta de métodos y de datos que resumen las **características comunes** de dicho conjunto

**2. (1 Pto) ¿En programación orientada a objetos (POO), a que se refiere el término objeto?**

Objeto es el concepto clave de la programación orientada a objetos, la idea de objeto es similar a la del mundo real, un objeto puede ser una silla, una mesa. ... los objetos tienen dos características: un estado y un comportamiento.

**3. (1 Pto) ¿Cuál es la diferencia entre una clase y un objeto?**

Un objeto es un simple elemento, no importa lo complejo que pueda ser. Una clase, por el contrario, describe una familia de elementos similares. En la práctica, una clase es un esquema o plantilla que se usa para definir o crear objetos.

**4. (1 Pto) ¿Qué es una arquitectura basada en capas?, defina cada uno de sus niveles o capas.**

La programación por capas es una técnica de ingeniería de software propia de la programación por

objetos, éstos se organizan principalmente en 3 capas: la capa de presentación o frontera, la capa de lógica

de negocio o control, y la capa de datos.

Siguiendo el modelo, el desarrollador se asegura avanzar en la programación del proyecto de una

forma ordenada, lo cual beneficia en cuanto a reducción de costos por tiempo, debido a que se podrá

avanzar de manera más segura en el desarrollo, al ser dividida la aplicación general en varios módulos y

capas que pueden ser tratados de manera independiente y hasta en forma paralela.

Por otra parte, otra característica importante de recalcar es la facilidad para las actualizaciones de

la aplicación. En este aspecto, la programación en capas juega un papel de suma importancia ya que sigue

un estándar conocido en el ambiente de desarrollo de aplicaciones, lo cual da al programador una guía para

hacer mejoras a la aplicación sin que esto sea una tarea tediosa y desgastante, siguiendo el estándar

establecido para tal fin y dividiendo las tareas en partes específicas para cada capa del proyecto.

Las principales capas que siempre deben estar en este modelo son:

**Capa de Presentación o Frontera:**

La presentación del programa ante el usuario, debe manejar interfaces que cumplan con el objetivo

principal de este componente, el cual es facilitar al usuario la interacción con la aplicación. Para esto se

utilizan patrones predefinidos para cada tipo de aplicación y para cada necesidad del usuario. La interfaz

debe ser amigable y fácil de utilizar, ya que el usuario final es el que se va a encargar de utilizar el sistema

y de dar retroalimentación al equipo de desarrollo en caso de que haya algo que mejorar.

Las interfaces deben ser consistentes con la información que se requiere, no se deben utilizar más

campos de los necesarios, así como la información requerida tiene que ser especificada de manera clara y

concisa, no debe haber más que lo necesario en cada formulario y por último, las interfaces deben satisfacer

los requerimientos del usuario, por lo cual no se debe excluir información solicitada por el usuario final y

no se debe incluir información no solicitada por el mismo.

Dentro de la parte técnica, la capa de presentación contiene los objetos encargados de comunicar

al usuario con el sistema mediante el intercambio de información, capturando y desplegando los datos

necesarios para realizar alguna tarea. En esta capa los datos se procesan de manera superficial, por ejemplo,

para determinar la validez de su formato o para darles algún orden específico.

Esta capa se comunica únicamente con la capa de Reglas de Negocio o Control.

**Capa de Lógica de Negocio o Control:**

Es llamada capa de reglas de negocio porque en esta se definen todas las reglas que se deben

cumplir para una correcta ejecución del programa. [2]

Es aquí donde se encuentra toda la lógica del programa, así como las estructuras de datos y objetos

encargados para la manipulación de los datos existentes, así como el procesamiento de la información

ingresada o solicitada por el usuario en la capa de presentación.

Representa el corazón de la aplicación ya que se comunica con todas las demás capas para poder

llevar a cabo las tareas. Por ejemplo, mediante la capa de presentación obtiene la información ingresada por

el usuario, y despliega los resultados. Si la aplicación se comunica con otros sistemas que actúan en

conjunto, lo hace mediante esta capa. También se comunica con la capa de datos para obtener información

existente o ingresar nuevos datos.

Recibe los datos que ingresó el usuario del sistema mediante la capa de presentación, luego los

procesa y crea objetos según lo que se necesite hacer con estos datos; esta acción se denomina

encapsulamiento.

Al encapsular los datos, el programa asegura mantener la consistencia de los mismos, así como

obtener información precisa de las bases de datos e ingresar en las mismas, solamente la información

necesaria, asegurando así no tener datos duplicados ni en las bases de datos, ni en los reportes solicitados

por el usuario.

**Capa de Datos:**

Es la encargada de realizar transacciones con bases de datos y con otros sistemas para obtener o

ingresar información al sistema.

El manejo de los datos debe realizarse de forma tal que haya consistencia en los mismos, de tal

forma los datos que se ingresan, así como los que se extraen de las bases de datos, deben ser consistentes y

precisos.

Es en esta capa donde se definen las consultas a realizar en la base de datos, tanto las consultas

simples como las consultas complejas parla generación de reportes más específicos.

Esta capa envía la información directamente a la capa de reglas de negocio para que sea procesada

e ingresada en objetos según se necesite, esta acción se denomina encapsulamiento.

**5. (2 Pto) Una palabra o frase palíndroma es aquella que se lee y escribe de igual forma al derecho y al revés, por ejemplo (arenera, oso, Ana, reconocer, amor a roma), diseñe un algoritmo (en pseudo-codigo o en el lenguaje programación de su preferencia) que reciba como parámetro una palabra o frase y retorne un mensaje informando al usuario si la palabra ingresada es palíndroma o no. Nota: No se permite el uso de funciones avanzadas como reverse de java u otras.**

**En la clase**

package palindromos;

class Palindroma {

public boolean palindroma(String palabra){

for(int i=0;i<palabra.length();i++){

palabra.toLowerCase();

palabra.replaceAll("","");

if(palabra.charAt(i) != palabra.charAt(palabra.length()-i-1)){

System.out.println("la palabra no es palindroma");

return false;

}

else{

System.out.println("la palabra es palindroma");

return true;

}

}return true;

}

}

**En el main**

package palindromos;

import java.util.Scanner;

public class Palindromos {

public static void main(String[] args) {

Scanner sc = new Scanner (System.in);

// punto #5, palabra o frase palindroma

Palindroma palabra = new Palindroma();

System.out.println("Por favor ingrese una palabra o frase para verificar si es palindrome: ");

String palabraFrase=sc.nextLine();

palabra.palindroma(palabraFrase);

}

}

**6. (1 Pto) ¿Cuál es el resultado de ejecutar el siguiente pseudo-codigo con el valor “5”?**

El resultado es 12

**7. (1 Pto) Realice un algoritmo que reciba como parámetro dos números enteros y retorne la división de ambos números.**

import static java.time.Clock.system;

import java.util.Scanner;

public class Division {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int numero1, numero2, division;

System.out.println("ingrese 2 numeros para realizar la division de los numeros ingresados");

System.out.println("Ingrese el primer numero");

numero1 = scan.nextInt();

System.out.println("Ingresa el siguiente numero");

numero2 = scan.nextInt();

division = numero1/numero2;

System.out.println("el resultado de la division es: " + division);

}

}

**8. (2 Pto) Escriba un algoritmo que imprima los números del 1 al 100. Pero para los múltiplos de 3 imprima "Fizz" en lugar del número y para los múltiplos de 5 imprima "Buzz". Para los números que son múltiplos de ambos imprima "FizzBuzz".**

public class Fizzbuzz {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

for(int i=1; i<=100; i++){

if( i % 3 ==0){

System.out.println("Fizz");

}

if(i % 5 ==0){

System.out.println("Buzz");

}

if(i % 3 ==0 && i % 5 ==0){

System.out.println("FizzBuzz");

}else{

System.out.println(i);

}

}

}

}

**9. Nombre tres tendencias actuales en el área de software.**

**-** Énfasis en la nube

- Incremento uso del Blockchain

- Evolución de la IA

**Prueba #2 bases de datos**

**1. (1 Pto) ¿Qué es una “primary key” o clave principal o primaria?**

Es el campo o conjunto de campos, que identifica de manera única los registros dentro de una tabla.

**2. (1 Pto) ¿Qué son las “foreign keys” o claves externas o foráneas?**

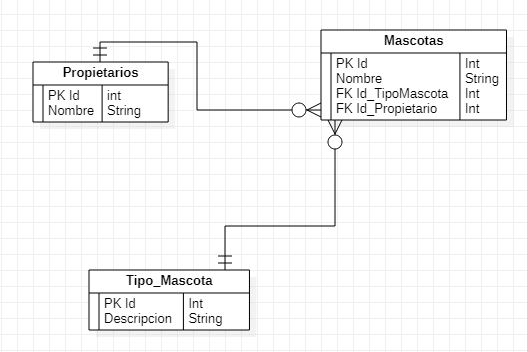
Una FOREIGN KEY es una clave (campo de una columna) que sirve para relacionar dos tablas. El campo FOREIGN KEY se relaciona o vincula con la PRIMARY KEY de otra tabla de la base de datos. La tabla secundaria es la que contiene la FOREIGN KEY y la tabla principal contiene la PRIMARY KEY.

**3. (1 Pto) ¿Qué es un stored procedure o procedimiento almacenado?**

son conjuntos de comandos SQL almacenados en el servidor, que al utilizarlos evitamos realizar estos comandos individualmente.

**4. (2 Pto) Elabore el modelo entidad relación lo más detallado posible (relaciones, tipos de datos, llaves primarias y**

**foráneas, etc…), si ud. lo considera necesario, puede adicionar más entidades o tablas.**



**5. (1 Pto) Listar todas las mascotas.**

SELECT \* FROM `mascotas`

**6. (1 Pto) Listar las mascotas que no han sido adoptadas.**

SELECT \* FROM `mascotas` WHERE Id\_propietario IS null;

**7. (1 Pto) Listar el número de mascotas por cada tipo de mascota.**

SELECT tipo\_mascota.Descripcion, COUNT(mascotas.Id\_tipoMascota) AS cantidad

FROM mascotas

INNER JOIN tipo\_mascota ON mascotas.Id\_tipoMascota=tipo\_mascota.Id

GROUP BY mascotas.Id\_tipoMascota;

**8. (1 Pto) Listar los propietarios que tengan más de una mascota.**

SELECT propietario.Nombre, COUNT(mascotas.Id) AS cantidad

FROM mascotas

INNER JOIN propietario ON mascotas.Id\_propietario=propietario.Id

GROUP BY mascotas.Id\_propietario

HAVING cantidad>1

**9. (1 Pto) Listar el número de mascotas por cada tipo de mascota y por cada propietario.**

SELECT tipo\_mascota.Descripcion, propietario.Nombre, COUNT(mascotas.Id) AS cantidad

FROM mascotas

INNER JOIN propietario ON mascotas.Id\_propietario=propietario.Id

INNER JOIN tipo\_mascota ON mascotas.Id\_tipoMascota=tipo\_mascota.Id

GROUP BY mascotas.Id\_propietario, mascotas.Id\_tipoMascota

**10. (1 Pto) Listas los propietarios que no tienen mascotas.**

SELECT propietario.Nombre, mascotas.Id FROM `propietario`

LEFT JOIN mascotas ON mascotas.Id\_propietario=propietario.Id

WHERE mascotas.Id is null

**11. (1 Pto) Cree un stored procedure o procedimiento almacenado con cualquiera de las sentencias anteriores.**

CREATE PROCEDURE `PropietariosSinMascotas`()

SELECT propietario.Nombre, mascotas.Id FROM `propietario`

LEFT JOIN mascotas ON mascotas.Id\_propietario=propietario.Id

WHERE mascotas.Id is null

**12. (2 Pto) La empresa “Mascotas & Mascotas”, además de prestar el servicio de adopción, también presta el servicio de veterinaria al público, la empresa desea conocer cuáles son las ventas mensuales por cada propietario. ¿Qué cambios le realizaría al modelo entidad relación para cumplir con este requisito?. Realice la sentencia SQL para obtener esta información.**

Crearía una nueva tabla llamada servicios\_clinicosmascotas, y sus atributos serían Id, Descripcion,Fecha,Id\_Propietario, Valor.

La sentencia para obtener el valor de los servicios por propietario es:

SELECT propietario.Nombre, SUM(servicios\_clinicosmascotas.Valor)

FROM `servicios\_clinicosmascotas`

INNER JOIN propietario ON propietario.Id = servicios\_clinicosmascotas.Id\_propietario

GROUP by propietario.Id

**13. (1 Pto) Escriba una sentencia SQL que actualice la columna “Nombre” de la tabla mascota, de modo que los nombres de todas las mascotas queden en mayúscula.**

UPDATE `mascotas` SET `Nombre`= UPPER(Nombre)

**14. (1 Pto) Escriba una sentencia SQL que elimine los propietarios que no tienen mascotas.**

DELETE propietario FROM `propietario`

LEFT JOIN mascotas ON mascotas.Id\_propietario=propietario.Id

WHERE mascotas.Id is null

**15. (1 Pto) Explique las siguientes funciones de agregación:**

**a. COUNT() :** cuenta el número de elementos en un conjunto. Incluye valores NULL y duplicados.

**b. MAX() :** es una función agregada que devuelve el valor máximo en un conjunto de registros.

**c. MIN() :** es una función agregada que te permite encontrar el valor mínimo en un conjunto de valores.