**TEST JAVA**

**Nombre\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1.- ¿Define que es Javac?**

**2.- ¿Define que es una clase?**

**3.- ¿Define que es un objeto?**

**4.- ¿Define que es Herencia?**

**5.- ¿Define que es polimorfismo?**

**6.- ¿Qué es el bytecode en Java?**

**7.- ¿Qué significa sobrecargar un método?**

**8.- ¿Que hace el siguiente código fuente?**

Package test;

import java.applet.Applet;

import java.awt.Graphics;

public class MiApplet extends Applet{

public void Paint(Graphic g){

g.drawString(“Hola Mundo”,40,80);

}

}

1.- Pinta en consola el texto “Hola Mundo”

2.- Genera un Applet con el texto “Hola Mundo”

3- Crea una ventana que contiene el texto “Hola Mundo”

4.- Ninguna de las respuestas anteriores es correcta.

**9.- ¿Crear la clase “Cliente” y además se necesitan crear los métodos Setter’s y Getter’s por cada atributo de dicha clase a partir de las siguientes variables.**

private Integer idcliente;

private String nombre;

private String apellidos;

private String dni;

private String direccion;

**public** **class** Cliente {

**private** Integer idcliente;

**private** String nombre;

**private** String apellidos;

**private** String dni;

**private** String direccion;

/\*\*

\* **@return** the idcliente

\*/

**public** Integer getIdcliente() {

**return** idcliente;

}

/\*\*

\* **@param** idcliente the idcliente to set

\*/

**public** **void** setIdcliente(Integer idcliente) {

**this**.idcliente = idcliente;

}

/\*\*

\* **@return** the nombre

\*/

**public** String getNombre() {

**return** nombre;

}

/\*\*

\* **@param** nombre the nombre to set

\*/

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

}

/\*\*

\* **@return** the apellidos

\*/

**public** String getApellidos() {

**return** apellidos;

}

/\*\*

\* **@param** apellidos the apellidos to set

\*/

**public** **void** setApellidos(String apellidos) {

**this**.apellidos = apellidos;

}

/\*\*

\* **@return** the dni

\*/

**public** String getDni() {

**return** dni;

}

/\*\*

\* **@param** dni the dni to set

\*/

**public** **void** setDni(String dni) {

**this**.dni = dni;

}

/\*\*

\* **@return** the direccion

\*/

**public** String getDireccion() {

**return** direccion;

}

/\*\*

\* **@param** direccion the direccion to set

\*/

**public** **void** setDireccion(String direccion) {

**this**.direccion = direccion;

}

}

**10.- Escribir un programa que lea por teclado un año, que además calcule y muestre si es bisiesto. Para realizar el cálculo utiliza un método llamado ‘esBisiesto’.**

**En la llamada al método bisiesto, el valor de la variable año se copia en la variable “a” y el método trabaja con esta variable. El valor true ó false devuelto por return pasa a ser el valor de la condición.**

**public** **static** **boolean** esBisiesto(**int** anio){

**return** ((anio % 4 == 0) && ((anio % 100 != 0) || (anio % 400 == 0)));

}

**public** **static** **void** bisiesto(**int** a){

**if**(*esBisiesto*(a))

System.***out***.println("El anio:"+a+" es bisiesto");

**else**{

System.***out***.println("El anio:"+a+" no es bisiesto");

}

}

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

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

*bisiesto*(entrada.nextInt());

}

**11.- Escribir un programa que imprima los N primeros números de la serie Fibonacci.**

**El primer número de la serie es 1, el segundo número es 1 y cada uno de los siguientes es la suma de los dos anteriores.**

**1,1,2,3,5,8,13……,N.**

**Ingresar número por teclado que sea mayor que 1.**

**public** **static** **void** fibonacci(**int** a, **int** b,**int** cnt){

**if**(cnt<1)

**return**;

System.***out***.println(a+b);

*fibonacci*(b, a+b, --cnt);

}

**public** **static** **void** startFibonacci(**int** n){

**if**(n==1){

System.***out***.println("1");

**return**;

}

System.***out***.println("1");

*fibonacci*(0, 1, --n);

}

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

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

**int** n=entrada.nextInt();

**if**(n>1)

*startFibonacci*(n);

}

**12.- Asumiendo que se cuenta con la clase “Conexion” y la tabla “Cliente” en la base de datos ACME, ahora se requiere escribir un método “registrar” de tipo public void que permita guardar datos en la tabla “Cliente”.**

**Cliente**

rfc

nombre

apellido\_p

apellido\_m

edad

sexo

direccion

**//Clase Conexion**

package Controlador;

import java.sql.\*;

public class Conexion {

public Connection conn;

public Statement stmt = null;

public boolean a;

public Conexion(){

try {

//Class.forName("oracle.jdbc.driver.OracleDriver");

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

System.out.println("Conectando a la Base de Datos...");

conn = DriverManager.getConnection("jdbc:oracle:thin:@host:port:nom\_db","user\_db","pass\_db");

System.out.println("Conexion establecida...");

stmt = conn.createStatement();

a = true;

} catch (Exception e) {

System.out.println("Error al conectarse: " + e.getLocalizedMessage());

a = false;

}finally{

System.out.println("Base de Datos conectada...");

}

}

public Statement getConexion(){

return stmt;

}

public void cerrarConexion()throws SQLException{

try {

conn.close();

} catch (Exception e) {

System.out.println("Error al cerrar conexion: " + e.getLocalizedMessage());

}

}

}

**public** **void** registrar(String rfc,String nombre,String apellidoP,String apellidoM,**int** edad,String sexo,String direccion) **throws** SQLException{

Conexion c= **new** Conexion();

Statement stm=c.getConexion();

String query="insert into ACME.Cliente values ('"+rfc+"','"+nombre+"','"+apellidoP+"','"+apellidoM+"',"+edad+",'"+sexo+"','"+direccion+"')";

stm.executeUpdate(query);

}

**13.- Escribir un método “baja” de tipo public void que elimine un registro de la tabla “Cliente” de la base de datos ACME a partir su RFC del cliente, tomando en cuenta la clase ‘Conexión’ del punto #11.**

**public** **void** baja(String rfc) **throws** SQLException{

Conexion c= **new** Conexion();

Statement stm=c.getConexion();

String query="delete from ACME.Cliente where rfc='"+rfc+"'";

stm.executeUpdate(query);

}

**14.- Escribir un método “actualizar” de tipo public void que modifique cualquier dato de la tabla “Cliente” de la base de datos ACME a partir de su RFC del cliente, tomando en cuenta la clase ‘Conexión’ del punto #11.**

**public** **void** update (String rfc,String campo, String value) **throws** SQLException{

Conexion c= **new** Conexion();

Statement stm=c.getConexion();

String query="update ACME.Cliente set "+campo+"='"+value+"' where rfc='"+rfc+"'";

stm.executeUpdate(query);

}

**15.- Desarrollar un programa en java que permita escribir a un archivo de texto plano en la siguiente ruta “c:/Users/Sicam/Documents/test.txt", debe aceptar cualquier cadena (Alfanumérico) capturado desde teclado.**

**Ocupar el siguiente cuerpo del programa.**

package test;

public class EscribirArchivo {

public static void main(String[] args){

}

}

**16.- Desarrollar un programa en java que permita leer el contenido del archivo ‘c:/Users/Sicam/Documents/test.txt’ que ha sido creado anteriormente en el punto #14.**

**Utilizar el siguiente cuerpo del programa.**

package test;

public class LeerArchivo {

public static void main(String [] arg) {

}

}

package test;

public class LeerArchivo {

**public** **static** **void** main(String a[]) **throws** IOException{

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

String s=entrada.nextLine();

BufferedWriter bw=Files.*newBufferedWriter*(Paths.*get*("**c:/Users/Sicam/Documents/test.txt**"), Charset.*defaultCharset*());

bw.write(s);

bw.close();

}

}

**17.- Manejando Threads/Hilos en Java, desarrollar un programa que en primer lugar imprima el nombre del Thread de la ejecución del método main (). Este hilo es asignado por la JVM (Java Virtual Machine). Entonces se pone en marcha 10 hilos y darles a todos un número como nombre ("" + i). Cada hilo luego imprime su nombre, y luego se detiene la ejecución.**

**public** **static** **void** main(String a[]) **throws** IOException{

System.***out***.println(Thread.*currentThread*().getName());

**for**(**int** i=0;i<10;++i){

Thread t= **new** Thread(**new** Runnable() {

@Override

**public** **void** run() {

System.***out***.println(Thread.*currentThread*().getName());

}

});

t.setName(""+i);

t.start();

}

}