Prueba técnica teórica java unificada
(Duración 1 hora)

Nombre candidato	William Steve Rodriguez Villamizar
Teléfono fijo	NA
Celular	300 438 1810
Fecha	1/06/2021 10:20pm
C.C.	1098685961

Las preguntas a continuación hacen parte del proceso de selección para el perfil de ingeniero de desarrollo JAVA.

Responda con base en sus conocimientos y experiencia, ya que de acuerdo con sus respuestas se realizará la categorización dentro del perfil. Las preguntas no contestadas no restan puntos.

Parte 1 – Conocimientos java (20 puntos)

1.1 ¿Qué es un “HashMap”? ¿Qué es un “Map” ?

HashMap es un ArrayList pero en modo diccionario de python o tipo Json, donde existe una key para acceder a un valor (u objeto), pero no se puede cambiar el key, solo el valor.

Map: es tambien un diccionario key, value ocmo Hashmap, pero las keys son unicas pero si values duplicados

Aunque son parecidos, es mas facil cambiar el value en el Hashmap que en Map

1.2 ¿Cuál es la diferencia entre excepciones chequeadas y excepciones no chequeadas? Mencione la clase Java que implementa cada una de ellas y un ejemplo de uso

1.3 Escriba el código que permita realizar el casting del atributo persona de tipo "Usuario" al tipo "Cliente".

```
Usuario user =  new Usuario();
Cliente client = new Cliente();
cliente.persona = client.persona;
```

1.4 Escriba el código java que permite imprimir por consola el objeto con valor "rojo" sabiendo que se tiene del siguiente Collection de objetos de tipo "Referencia" donde no es posible determinar el orden de sus elementos. El código debe evidenciar como se recorre la colección.

Referencia		Referencia		Referencia		Referencia		Referencia	
5	verde	8	rojo	3	amarillo	4	gris	1	blanco

Objeto referencia:

```
public class Referencia {
	public int id;
	public String color;

	public Referencia() {
		super();
	}
	public Referencia(int id, String color) {
		super();
		this.id = id;
		this.color = color;
	}
}
```

Principal code

```
import java.util.ArrayList;
import java.util.Iterator;

public class test_1 {
	public static void main(String[] args) {
		ArrayList<Referencia> list=new ArrayList<Referencia>();
		list.add( new Referencia(5, "verde"));
		list.add( new Referencia(8, "rojo"));
		list.add( new Referencia(3, "amarillo"));
		list.add( new Referencia(4, "gris"));
		list.add( new Referencia(1, "blanco"));

		Referencia find = Search(list, "rojo");

		System.out.println("Id: " + find.id + " - color: " + find.color);
  }

	public static Referencia Search(ArrayList<Referencia> list, String color) {
		Referencia ref = new Referencia();
		for (Iterator<Referencia> i=list.iterator(); i.hasNext(); ) {
			Referencia temp = i.next();
			if(temp.color.equals(color)) {
				ref = temp;
				break;
			}
		}
		return ref;
	}
}
```

Parte 2. Conocimientos SQL y Base de datos (20 puntos)

Para las siguientes preguntas asuma la siguiente tabla de "Ordenes":

Id	Fecha	Precio	Cliente
1	2020/11/12	1000	Pedro
2	2020/10/23	1600	Jairo
3	2020/09/02	700	Pedro
4	2020/09/03	300	Pedro
5	2020/08/30	2000	Juan
6	2020/10/04	100	Jairo

2.1 Escriba la consulta SQL que cuenta los registros de la tabla “Ordenes”

```
SELECT COUNT(id) AS cantidad_registros FROM Ordenes
```

2.2 Escriba la consulta SQL que trae las ordenes con precio mayor a “500”, y que estén ordenadas por el campo fecha (de la más reciente hasta la menos reciente)

```
SELECT id, Fecha, Precio, Cliente FROM Ordenes WHERE Precio > 500 ORDER BY Fecha DESC
```

2.3 Escriba la consulta SQL que trae las ordenes de los clientes que comienzan por “J”

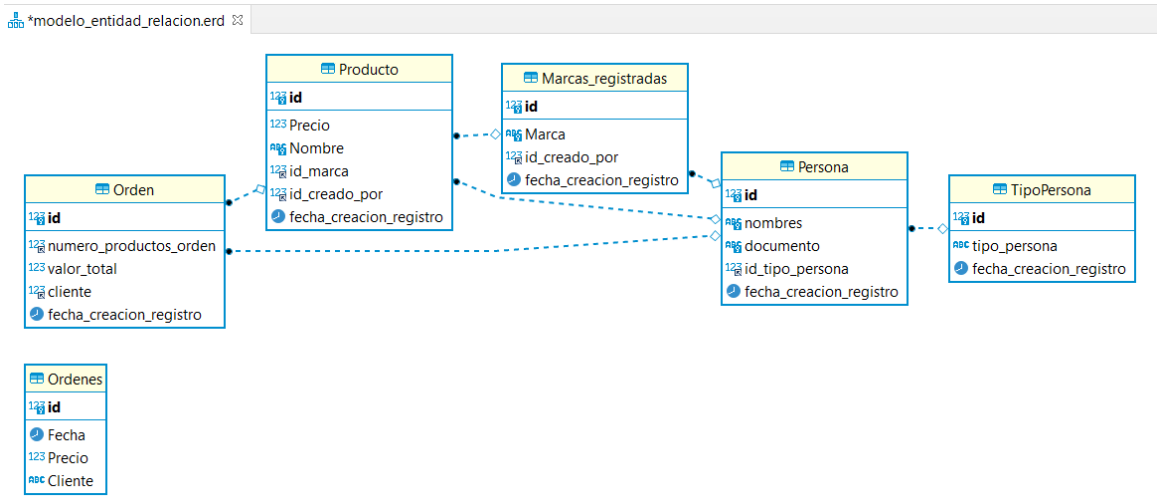
```
SELECT * FROM Ordenes WHERE Cliente LIKE "J%"
```

2.4 Escriba la consulta SQL que trae los siguientes resultados

Cliente	**Suma de “Precio”**
Pedro	2000
Jairo	1700
Juan	2000

```
SELECT Cliente, SUM(Precio) AS "Suma de Precio" FROM Ordenes GROUP BY Cliente ORDER BY Cliente DESC
```

2.5 Haga un modelo “Entidad-Relación” de las entidades “Persona” (nombres, documento,) , “Producto” (Precio, nombre, marca) , “Orden” (N productos dentro de la orden, valor total, cliente).



Nota: Luego de tener todas las tablas montadas, se puede llegar a la tabla original mostrada en el punto 2 de este test, con la siguiente consulta:

```
SELECT O.fecha_creacion_registro AS Fecha, SUM(O.valor_total) AS Precio,
Pe.nombres AS Cliente  FROM Orden AS O
INNER JOIN Producto AS P ON O.numero_productos_orden = P.id
INNER JOIN Persona AS Pe ON O.cliente = Pe.id
GROUP BY O.fecha_creacion_registro
ORDER BY O.fecha_creacion_registro
```

Parte 3. Conocimientos JEE y JPA (20 puntos)

3.1 ¿Qué es un EJB?. Enumere y explique los tipos de EJBs que existen.

EJB = Enterprise JavaBeans

Este tiene tres tipos:

1) Entidad: guarda todos los objetos en el servidor, aunque esto es obsoleto y esta por desaparecer.

2) Sesion: De dos sabores

2,1) Estado: Se almacenan datos en la sesion del cliente y estos se borran al terminar la sesion.

2,2) No estado: No se almacenan datos, pero se puede acceder a datos almacenados.

3) Mensajes: Son asincronos, funcionan parecido al mqtt, se subscribe a topics y el mensaje

llega a los subscritos.

3.2 Con cual anotación JEE / JPA se obtiene una instancia del "EntityManager" para trabajar con la persistencia de objetos en la base de datos?

Persistence.EntityManagerFactory

3.3 Explique qué significan las relaciones EAGER y LAZY entre entidades JPA (JEE5). ¿En qué casos se usa cada una de ellas?

LAZY: cuando existen relaciones entres entidades, las que no tienen dependencia no se inicializan hasta ser llamadas, para evitar una sobrecarga de red o sistema, asi solo se invoca lo que es extrictamente necesario, lo demás queda en espera hasta ser pedido, el problema es que si no se cierra controladamente puede reportar un error en el log.

EAGER = no lazy: todas las relaciones se inician a penas se inicia el contenedor, pero no es bueno abusar de esto porque seria un desperdicio cargar datos que no se necesitan.