**Prueba técnica teórica java unificada**

**(Duración 1 hora)**

|  |  |
| --- | --- |
| Nombre candidato | Eduardo Montaño Cardenas |
| Teléfono fijo |  |
| Celular | 3225921678 |
| Fecha | 02/04/2021 |
| C.C. | 1053840500 |

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

**La estructura Map** es una interface que nos proporciona una estructura en forma de <clave, valor> en donde esta clave es el identificador de los valores en esta estructura por la cual estas claves deben de ser únicas y se asigna a cualquier tipo de valor, también denominado diccionario

**Mientras que HashMap** es una implementación de la estructura **Map** con tablas hash el cual almacena datos e esta estructura clave valor garantizando que los elementos que vamos agregando con **put** estén ordenados, este trabaja asignando con el método **hashcode()** y **equals()** para así comparar de una forma mas rápida

* 1. ¿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

Las excepciones chequeadas son errores que pueden presentarse en la aplicación por **razones externas** a la aplicación como por ejemplo los problemas al tratar archivos,

Problemas de red, bases de datos estas implementan de la clase **excepction**, en estos ti**pos de excepciones el sistema puede recuperarse de detenerse repentinamente.**

Mientras que las **excepciones no chequeadas** son lanzadas por la aplicación y puede tener **errores en el tiempo de ejecucion** que suceden por aspectos internos entre los más comunes son los valores nulos, desbordes de listas el cual **detiene abruptamente el sistema si no son controlados** implementadas por **RuntimeException**

* 1. Escriba el código que permita realizar el casting del atributo persona de tipo “Usuario” al tipo “Cliente”.

// Cliente extends Usuario

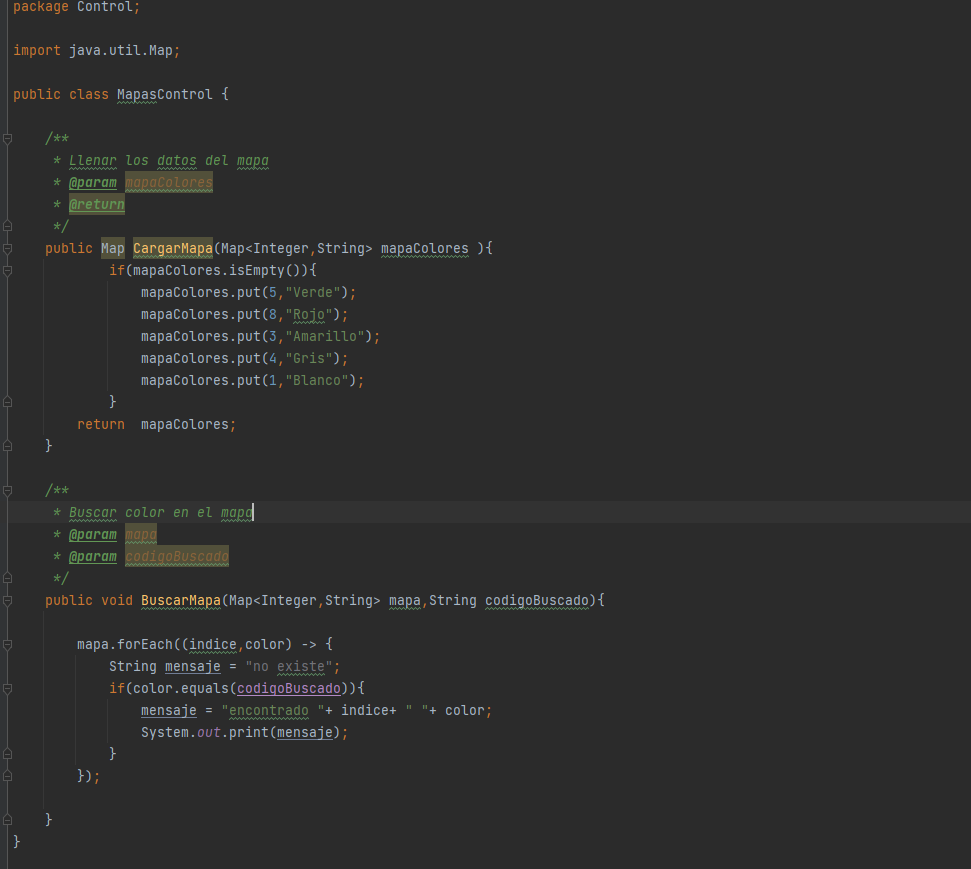
//throws ClassNotFoundException

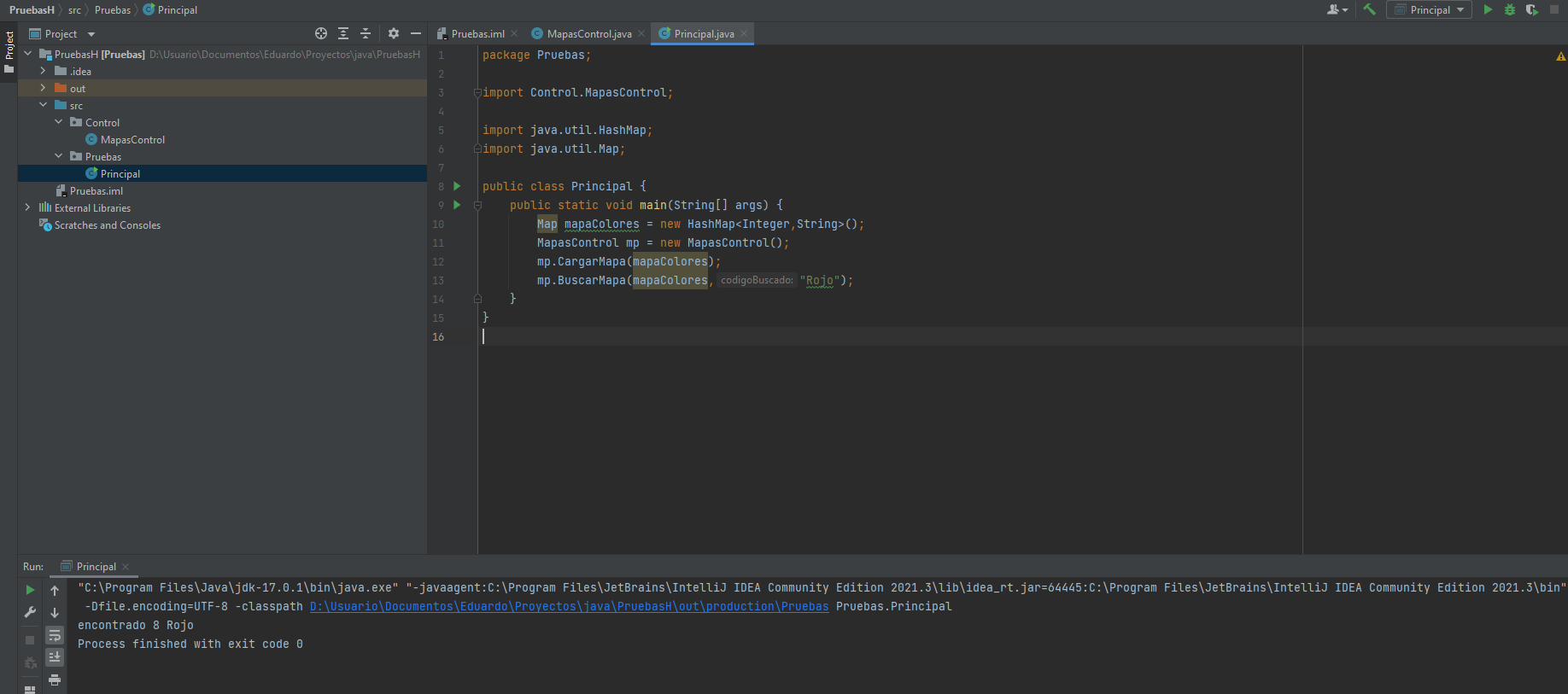
If (Class.forName(Usuario).isInstance(Cliente))

Usuario.SetPersona((Usuario) Cliente.persona);

* 1. 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 |  | |  |





**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 |

* 1. Escriba la consulta SQL que cuenta los registros de la tabla “Ordenes”

**SELECT COUNT(Id) as NroRegistros FROM Ordenes**;

* 1. 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 Precio, Fecha FROM Ordenes WHERE Precio>500 ORDER BY Fecha DESC;**

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

**SELECT \* FROM Ordenes WHERE LOWER(Cliente) LIKE ‘j%’**

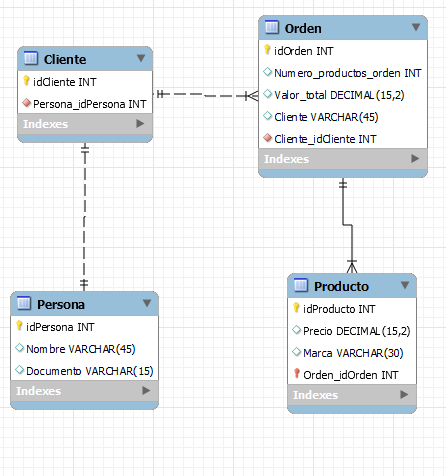
* 1. Escriba la consulta SQL que trae los siguientes resultados

|  |  |
| --- | --- |
| **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**

* 1. 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).



### Parte 3. Conocimientos JEE y JPA (20 puntos)

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

EJB es Enterprise Java Beans son interfaces de programación de aplicaciones que proporcionan un modelo del lado del servidor para **extraer problemas generales de la programación** como temas de seguridad, persistencia entre otros

Existen 3 tipos de EJB

**Beans de sesión** en donde se almacena los datos en la sesión del cliente en donde se clasifica según su estado donde se diferencia en que estando con estado se borra los datos de la sesión y sin estado solo se pude acceder a estos

**Beans de entidad** Se guardan objetos en el servidos de forma persistente

**Mensajes** Brinda la posibilidad de mandar mensajes, notificaciones constantemente

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?

@Entity

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

Eager: Arranca todas las relaciones al iniciar el contenedor el cual no es recomendable por el abuso de datos

Lazy: Las entidades que no tienen dependencias solo inician cuando son llamadas así para evitar sobrecargar el sistema Nota debe cerrarse controladamente