PRUEBA TÉCNICA

Nombre: **Fecha**:

Cedula:

Objetivo: Evaluar los conocimientos técnicos en cuanto a Desarrollo de Software usando la plataforma JEE.

1) **Defina los siguientes conceptos:**
   - Clase: Podría definir este concepto como un marco o plantilla la cual puede definir las características y comportamiento de algún objeto o entidad específicos.
   - Objeto: Quizá se asemeja mucho a una entidad que representa las características algo en específico mediante datos.
   - Herencia: Es uno de los tantos pilares de la programación orientada a objetos (POO), el cual tiene como objetivo definir una clase padre, con la cual clases hij@s podrán heredar características propias mediante la palabra reservada **extends**.
   - Polimorfismo: En pocas palabras, este concepto hace referencia también a otros de los pilares de la (POO), con la cual, define la capacidad que tiene un objeto de adquirir más de una forma, es decir, comportarse de distintas formas dependiendo de factores adversos como los datos y/o parámetros definidos. Está muy relacionado al principio abierto-cerrado.
   - Sobrecarga: Otro de los principios de la (POO) con la cual, creamos métodos con el mismo nombre, pero con distinta cantidad y tipos de parámetros, es decir, realizar una sobrecarga del mismo método.
   - Paquete: Es una especie de carpeta con colecciona un conjunto de clases, interfaces, entidades, etc. Es el mecanismo empleado para realizar la modularización en una aplicación.
   - Framework: Se puede definir como una base codificada que puede contener clases, interfaces, bibliotecas, librerías, etc, que facilita y reduce tiempos a la hora de desarrollar.

2) **Asumiendo que existe una clase publica llamada Apuesta con métodos public int getValorApuesta() y public void set ValorApuesta(int valor)**

```
public void encimarApuesta(){
    Apuesta apuesta = null;
```

```
        apuesta = crearApuesta(apuesta);
        System.out.println(apuesta.getValorApuesta());

    }

    public Apuesta crearApuesta(Apuesta apuesta){
        apuesta = new Apuesta();
        apuesta.setValorApuesta(1500);
        return apuesta;
    }
```

El resultado de la línea en negrilla una vez ha sido ejecutado por completo el método encimarApuesta es:

a. 1500
b. 0
c. Null
d. Es Lanzada una excepción.

3) Para qué sirve el archivo MANIFEST.MF en una aplicación y donde está ubicado.

R./ Se puede definir como archivo de metadatos, el cual contiene datos de la aplicación desarrollada y puesta en marcha, datos como el nombre de la aplicación, la versión, licencia o tipo de licencia, compatibilidades y archivos o programas necesarios para su correcto funcionamiento. Este archivo siempre va ubicado dentro del archivo o artefacto .jar de la aplicación.

4) En una aplicación JSF donde se definen las reglas de navegación y Backing Bean.

R./

5) Si se requiere un filtro en la aplicación para que cada vez que se solicite una página se verifique que el usuario esté autenticado que archivos debería construir y cuales debería modificar.

R./ Si se trabaja del lado Backend con el framework SpringBoot, lo ideal sería trabajar la autenticación de usuarios con Spring Security y JWT, con lo cual se definen las reglas de acceso en el método configure de la clase MainSecurity, adicional, crear una especie de interceptor con el cual, cada petición que entre, acceda primero a las clases jwtEntrypoint, jwtprovider y jwtTokenFilter para verificar, controlar y asignar permisos de acceso mediante la información de sesión contenida dentro del token recibido.

6) Mencione los 7 tipos de datos primitivos en Java:

R./

int: Tipo de dato entero natural incluyendo números negativos.

boolean: Tipo de dato bandera, solo puede tener 2 valores o estados (true o false).

char: Tipo de dato carácter único o simple, el cual solo puede almacenar un carácter específico.

double: Representa los números decimales dobles, es decir, tiene la capacidad de almacenamiento más grande que el tipo float.

float: Tipo de dato número decimal sencillo, tiene menor capacidad de almacenamiento que el tipo double.

byte: Tipo de dato entero con capacidad de almacenamiento de 1byte.

short: Tipo de dato entero con capacidad pequeña de almacenamiento, 2 bytes.

long: Tipo de dato entero con capacidad grande de almacenamiento, 8 bytes.

String: Cadenas de caracteres, ya sean alfanuméricos o cualquier carácter especial.

7)
```
public static void main(String[] args){
    for(int i=0; i < =10; i++){
    if (i > 6) break;
    }

    System.out.println(i);

}
```

El resultado de la ejecución del código anterior es:

a- 6
b- 7
c- 10
d- Compilación Fallida.
e- Una excepción es lanzada.

8) En J2EE que es un EJB. Mencione los principales tipos.

R./

Son tipos de interfaces programadas (API) que hacen parte del esquema java Enterprise.

9) En qué tipo de EJB se puede mapear una tabla de base de datos.

R./

10) Defina y de un escenario de aplicación de los siguientes patrones de diseño:

Singletón, Fachada, Fábrica Abstracta.

R./

Singleton: Hace parte de los patrones de diseño creativos o creacionales. Con este patrón el objetivo es evitar crear más de un objeto por clase, es decir, que podemos definir únicamente un objeto por clase.

Facade: Hace parte de los patrones de diseño estructurales. Con este patrón podemos crear e implementar una clase tipo fachada, la cual sirve para llamar, usar e implementar una pequeña parte de los recursos que nos proveen bibliotecas, librerías y/o subsistemas grandes y complejos.

11) Se tienen las siguientes entidades:

Empresa(Id_empresa, nombre_empresa)
Producto(Id_producto, nombre_producto)
Vendedor(Id_vendedor, num_documento, nombre, fk_id_empresa)
Venta(Id_venta, valor_total, fk_id_vendedor, fecha, fk_id_cliente)
DetalleVenta(id_detalle_venta, valor_total, fk_id_producto, cantidad, fk_id_venta)

Realizar una consulta SQL que muestre la información de la siguiente manera:

Empresa	Producto	Cantidad	Valor Total

R./

SELECT e.nombre_empresa, p.nombre_producto, dv.cantidad, dv.valor_total FROM DetalleVenta dv INNER JOIN Venta v ON dv.fk_id_venta=v.Id_venta INNER JOIN Vendedor ve ON ve.id_vendedor=v.fk_id_vendedor INNER JOIN Empresa e ON e.id_empresa=ve.fk_id_empresa INNER JOIN Producto p ON p.id_producto=dv.fk_id_producto;

12) Escriba un procedimiento almacenado o aplicación Java que consolide los totales de venta del punto anterior, garantizando tener solo los últimos 6 meses en dicho consolidado

R./

13) Para qué sirve el entityManager y cuáles son los métodos básicos que este provee.

R./

Termino para referirnos a un contexto de persistencia de datos u objetos. Con este contexto, si fuera el caso de usar JPA, podemos gestionar la persistencia de las entidades y su ciclo de vida dentro de la aplicación.

Persist(): Es el método encargado de almacenar nuevas entidades a la base de datos.

Contains(): Nos sirve para verificar y comprobar si una entidad es gestionada por el entityManager de la aplicación.

Find(): Podemos buscar y encontrar rápidamente una entidad específica usando su llave primaria.

Remove(): Podemos eliminar una entidad específica de la base de datos.

14) Escriba 3 formas diferentes de recorrer un arreglo usando el lenguaje Java

R./

- La primera forma es la forma tradicional, usando el for tradicional o con índice y recorrer cada espacio reservado por el array en memoria:

  ```
  for(int i = 0; i < array.length; i++) {
  }
  ```

- La segunda forma es usar un for sin índice o for each:
  ```
  for(int edad: edades) {
  }
  ```
- La tercera forma es usar un ciclo while teniendo en cuenta un índice, así:
  ```
  Int índice = 0;
  while(índice < array.length) {
  }
  ```

15) Qué es un patrón de diseño.

R./

Hace referencia a lineamientos que fueron generados para darle solución a los distintos problemas que se pueden presentar en el diseño de software.

16) Construya un XML que defina una estructura jerárquica para Países, Departamentos, Ciudades y Barrios.

R./

```
<?xml version="1.0" encoding="UTF-8"?>
<barrios>
  <barrio>
    <nombre>San Jorge</nombre>
    <ciudad>
        <nombre>Girardot</nombre>
        <departamento>
                <nombre>Cundinamarca</nombre>
                <país>
                        <nombre>Colombia</nombre>
                </país>
```

```
        </departamento>
    </ciudad>
  </barrio>
</barrios>
```

17 **Se requiere de una aplicación web (Angular, JSF) por medio de la cual se puedan vender recargas en línea. Se debe poder identificar en cualquier momento la cantidad y valor de recargas discriminada por operador (Tigo, Movistar, Comcel, Uff) y persona que realiza la venta.**

a. **Implementar APIs necesarias en Spring boot (Opcional)**
b. **Crear pantalla para la venta de recargas (no se requiere diseño)**
c. **Realice un diagrama relacional, diagrama de casos de uso, diagrama de secuencia y de clases que sirva como solución para dicha implementación.**
d. **Subir la solución a un repositorio GIT**