Examen diagnóstico para desarrollador Java – Arquitectura de Crédito Ágil

1) Indique cuáles son y explique los 4 pilares de la programación Orientada a Objetos.

Herencia: Permite que una clase herede características de otra clase.
Polimorfismo: Permite que una interfaz sea utilizada para diferentes tipos de objetos.
Encapsulamiento: Oculta los detalles internos y expone solo lo necesario.
Abstracción: Simplifica el modelo al ocultar detalles complejos.

2) ¿Qué es una clase en Java?

Se puede decir que una clase en Java es una plantilla que estructura el comportamiento de un objeto que se crea en la misma, dentro de ella de se describen los atributos del objeto.

3) ¿Qué es un objeto y cómo se declara en Java?

Un objeto es una instancia que se crea a partir de una clase.

4) Indique los tipos de operadores y dé ejemplos de estos:

Son símbolos especiales que realizan operaciones sobre uno o más valores, existen varios tipos.

Operadores Aritméticos

```
int a = 5;
int b = 5;
int suma = a + b;  // Resultado: 10
int resto = a - b;  // Resultado: 5
```

Operadores Relacionales

```
int a = 10;
int b = 20;
boolean mayor = a > b;  // Resultado: false
```

Operadores Lógicos

```
boolean a = true;
boolean b = false;
(a > b) && (b < 0);// Resultado: true
```

Operadores de Asignación

```
int a = 5;
a += 3;  // Resultado: a = 8
```

Operadores Unarios

```
int a = 5;
a++;  // Resultado: a = 6
```

Operadores Ternarios

```
int a = 5;
int b = 10;
int max = (a > b) ? a : b;  // Resultado: max = 10
```

Operadores de Bits

```
int a = 5;  // Binario: 0101
int b = 3;  // Binario: 0011
int resultado = a & b;  // Resultado: 1 (0001 en binario)
```

Operadores de Comparación de Referencia

```
String a = new String("Hola");
String b = new String("Hola");
boolean resultado = (a == b);  // Resultado: false, porque son dos objetos diferentes
```

5) Defina qué es un modificador de acceso e indique cuáles son los que utiliza Java.

Es una palabra clave o palabra reservada que se utiliza para indicar el nivel de acceso a los métodos, clases, variables o constructores.

- **public**: Acceso sin restricciones.
- **private**: Acceso limitado solo dentro de la clase.
- **protected**: Acceso dentro del mismo paquete y en las subclases, incluso si están en otros paquetes.
- **Sin modificador (paquete o "default")**: Acceso solo dentro del mismo paquete.

6) ¿Qué es una función lambda y cómo se utiliza en Java?

Son expresiones que representa un bloque de código que se puede pasar como argumento a un método o almacenar como una variable.

Su uso sigue la siguiente sintaxis:
(parametros) -> { cuerpo_de_la_lambda }

7) ¿Qué es una clase genérica y cómo se instancia? Complemente su respuesta con 3 ejemplos de clases genéricas en Java.

Es una clase que puede manejar objetos de cualquier tipo de manera flexible, sin necesidad de especificar el tipo exacto de datos.

Instanciación:

```
// Definición de clase genérica con el tipo 'T'
public class Caja<T> {
    private T contenido;

    public void guardar(T contenido) {
        this.contenido = contenido;
    }

    public T abrir() {
        return contenido;
    }
}

//Inctanciamos Caja con String
Caja<String> cajaDeTexto = new Caja<>();
cajaDeTexto.guardar("Hola, Mundo");
System.out.println(cajaDeTexto.abrir());  // Resultado: Hola, Mundo
```

Ejemplos:
- ArrayList<T>
- HashMap<K, V>
- Optional<T>

8) Defina qué es un hilo y dé un ejemplo de cómo se declara en Java.

Un hilo es una unidad básica de ejecución dentro de un proceso. Los hilos permiten que los programas realicen múltiples tareas simultáneamente, es decir, facilitan la **concurrencia** y el **paralelismo**.

Hilo con Thread

```
class MiHilo extends Thread {
    @Override
    public void run() {
        // Cuerpo del metodo
    }
}
```

Hilo con runable.

```
class MiRunnable implements Runnable {
    @Override
    public void run() {
        // Cuerpo del metodo
    }
}
```

9) Defina qué es una excepción y explique cómo se controlan en Java.

Es un evento que interrumpe el flujo normal de ejecución de un programa cuando ocurre un error inesperado, Java permite manejar excepciones mediante los bloques **try-catch-finally.**

```
try {
    int resultado = 10 / 0; // Puede generar una excepción
} catch (ArithmeticException e) {
    System.out.println("Error: División por cero");
} finally {
    System.out.println("Esto se ejecuta siempre");
}
```

10) ¿Qué es JSON?

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos, se utiliza ampliamente para transmitir datos entre un servidor y un cliente en aplicaciones web.

11) Para proyectos de Maven, ¿cuál es el archivo en el que se declara información general del mismo como librerías, versión de Java, plugins, etc.?

En proyectos Maven, toda la información general del proyecto, como dependencias, versión de Java, plugins, etc., se declara en el archivo **pom.xml** (Project Object Model).

12) Defina los conceptos “Inversión de Control” e “Inyección de Dependencias”.

Inversión de Control (IoC): Es un principio de diseño en el cual el control del flujo de un programa es invertido, es decir, en lugar de que el código del programa controle las dependencias y el flujo, un framework o contenedor lo hace.

Inyección de Dependencias (DI): Es una técnica específica de IoC donde las dependencias (objetos requeridos por una clase) se suministran desde el exterior, ya sea a través del constructor, métodos o campos, en lugar de que la propia clase las cree.

13) ¿Qué es una interfaz?

Una interfaz en Java es un contrato que define métodos sin implementar. Las clases que la implementan deben proporcionar la lógica de esos métodos. Permite especificar comportamientos que diferentes clases deben seguir.

14) Defina qué es una anotación en Java e indique ejemplos de estás.

15) Defina qué es el control de versiones y mencione 3 ejemplos de herramientas que ayudan este proceso.

El **control de versiones** es un sistema que permite gestionar y registrar cambios en archivos y proyectos a lo largo del tiempo, facilitando la colaboración y el seguimiento de versiones anteriores.
Ejemplos de herramientas:

1. Git
2. Subversion (SVN)
3. Mercurial

16) ¿Qué es un API?

Un **API** (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y definiciones que permite a diferentes aplicaciones comunicarse entre sí, facilitando el acceso a funciones o datos de un servicio o software.

17) ¿Qué es un servicio REST?

Un **servicio REST** (Representational State Transfer) es una arquitectura que permite la comunicación entre sistemas a través de HTTP, utilizando operaciones como GET, POST, PUT, y DELETE para manipular recursos representados en formato JSON o XML.

18) ¿Qué es un Endpoint?

Es una URL específica de una API que recibe solicitudes y devuelve respuestas, permitiendo el acceso a un recurso o servicio.

19) ¿Qué es un contrato de interfaz?

Es la definición formal de cómo interactuar con un servicio o API, especificando los métodos disponibles, parámetros, y tipos de respuestas.

20) ¿Qué es un Token JWT?

Es un token de autenticación basado en JSON, que contiene información codificada sobre el usuario y su autorización, usado para validar identidades de manera segura.

21) Explique las diferencias entre una aplicación monolítica y un microservicio.

- **Monolítica**: Toda la lógica de la aplicación se implementa en una única unidad.
- **Microservicio**: La aplicación se divide en servicios independientes y pequeños, que se comunican entre sí, facilitando escalabilidad y mantenimiento.

22) ¿Cuál es la salida del siguiente bloque de código? **Justifique su respuesta.**

```
public class ExamenTest {
    public static void main(String[] args) {
        Persona p1 = new Persona("Juan",20);
        System.out.println(p1.edad);
    }
}
```

```
public class Persona {
    private String nombre;
    private Integer edad;
    public Persona(String nombre, Integer edad) {this.nombre = nombre;this.edad = edad;}
    public String getNombre(){return nombre;}
    public void setNombre(String nombre) {this.nombre = nombre;}
    public Integer getEdad() {return edad;}
    public void setEdad(Integer edad) {this.edad = edad;}

}
```

El resultado es un errorya que que el campo edad de la clase Persona es privado (private), por lo que no se puede acceder directamente a él desde fuera de la clase. En el método main, intentas acceder a p1.edad de manera directa, lo cual no es permitido.

23) ¿Cuál es la salida del siguiente bloque de código? Justifique su respuesta.

```java
public class ExamenTest {
    public static void main(String[] args) {
        String var1 = "Bienvenido", var2 = " a Banco", var3 = "Azteca";
        var1 = var1.replace("o", "0");
        var1.concat(var2);
        var1.concat(var3);
        System.out.println(var1);
    }
}
```

La salida seria: Bienvenid0

Aunque se hace una modificación a var1 usando replace("o", "0"), los métodos concat() no alteran la cadena original porque las cadenas en Java son **inmutables**. Para que el resultado de las concatenaciones se refleje, se debería reasignar el resultado a var1.

24) ¿Cuál es la salida del siguiente bloque de código? Justifique su respuesta.

```java
public class ExamenTest {
    public static void main(String[] args) {

        Integer[] arreglo= {3,9,1,8,2,7,5,8,20,10};
        List<Integer> lista= Arrays.asList(arreglo);
        lista.stream().sorted((n1,n2)->n2.compareTo(n1)).forEach(t->System.out.print(t +", "));
    }
}
```

Salida: 20, 10, 9, 8, 8, 7, 5, 3, 2, 1,
El código usa **Arrays.asList()** para crear una lista fija, luego un **stream** ordena los números en orden descendente con **sorted()**, y **forEach()** los imprime en una línea con una coma y un espacio. La lista no se modifica, solo se procesa en el stream.

25) ¿Cuál es la salida del siguiente bloque de código? Justifique su respuesta.

```java
public class ExamenTest {
    public static void main(String[] args) {
        BancoAzteca b1 = new BancoAzteca("Juan Perez", 1);
        System.out.println(b1.getNombreEmpleado());
    }
}
public abstract class BancoAzteca {
    private String nombreEmpleado;
    private int id;

    public BancoAzteca(String nombreEmpleado, int id) { this.nombreEmpleado = nombreEmpleado;   this.id = id;}
    public String getNombreEmpleado() {return nombreEmpleado;}
    public void setNombreEmpleado(String nombreEmpleado) {this.nombreEmpleado = nombreEmpleado;}
    public int getId() {return id;}
    public void setId(int id) {this.id = id;}

}
```

El código no compilará y generará un **error**.
La clase **BancoAzteca** es **abstracta**, y no se pueden crear instancias directas de una clase abstracta. En este caso, se intenta crear un objeto de la clase abstracta con new BancoAzteca("Juan Perez", 1), lo cual no está permitido.

26) ¿Cuál es la salida del siguiente bloque de código? Justifique su respuesta.

```
public class BancoAzteca {
    public static void main(String[] args) {
        String str = null;
        try {str.concat("Hola BancoAzteca");}
        catch (Exception e) {System.out.println("Exception: " + e);}
    }}
```

Salida: Exception: java.lang.NullPointerException
Se intenta llamar al método concat() sobre una referencia nula (str), lo que provoca un **NullPointerException**. El bloque catch captura esta excepción y la imprime.

27) ¿Cuál es la salida del siguiente bloque de código? Justifique su respuesta

```
package com.banco.azteca;
public class Examen {
    String banco;
public Examen(String banco) {this.banco = banco;}
public static void main (String[] args) {
    System.out.println(new Examen("Hola"));
    System.out.println(new String("Banco Azteca"));}}
```

Salida:
Examen@3b22cdd0
Banco Azteca

El primer println muestra la representación predeterminada del objeto Examen (con un hashcode), y el segundo println muestra la cadena "Banco Azteca".

Práctica: Se tiene planificado migrar una aplicación monolítica que se encarga de almacenar toda la información de un cliente para separarla en distintos microservicios, comenzando por los siguientes datos: Nombre, apellido paterno, apellido materno, fecha de nacimiento, sexo, correo, teléfono.

Tomando en cuenta el planteamiento anterior, realice lo siguiente:

- Desarrollar microservicios con Java 17, Spring Boot 2.6.6 y Maven 3.8.x (al menos 2).
- Utilizar diseño MVC.
- Utilizar JSON para mapear las peticiones del microservicio.
- Implementar **las** funcionalidades para insertar y consultar de Base de Datos la información anteriormente mencionada.
- Agregar contrato de interfaz yaml con estándar OpenAPI 2.0.
- Realizar Script para crear la BD.
- No puede haber dos personas con el mismo nombre.
- El alta de datos debe devolver un identificador del cliente.
- La búsqueda debe hacerse con el identificador y debe devolver los datos almacenados.
- Implementar loggeo (aunque sean prints).

Se evaluará:

- Que funcione.
- Legibilidad del código.
- Tiempos de respuesta.
- Pruebas Unitarias (mínimo un método/caso).

Enviar ZIP con respuestas y práctica al siguiente correo:
acamicroservicios@elektra.com.mx