Examen diagnóstico para desarrollador Java – Arquitectura de Crédito Ágil

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

Los 4 pilares de la POO son los siguientes:

1. Abstracción: La abstracción es la propiedad que considera los aspectos más significativos o notables de un problema y expresa una solución en esos términos. La abstracción posee diversos grados o niveles de abstracción, los cuales ayudan a estructurar la complejidad intrínseca que poseen los sistemas del mundo real. La abstracción encarada desde el punto de vista de la programación orientada a objetos es el mecanismo por el cual se proveen los límites conceptuales de los objetos y se expresan sus características esenciales, dejando de lado sus características no esenciales.
2. Encapsulamiento: Significa reunir en una cierta estructura a todos los elementos que a un cierto nivel de abstracción se pueden considerar pertenecientes a una misma entidad, y es el proceso de agrupamiento de datos y operaciones relacionadas bajo una misma unidad de programación, lo que permite aumentar la cohesión de los componentes del sistema. El encapsulamiento oculta lo que hace un objeto de lo que hacen otros objetos y del mundo exterior por lo que se denomina también ocultación de datos. Un objeto tiene que presentar “una cara” al mundo exterior de modo que se puedan iniciar sus operaciones.
3. Herencia: La herencia es una relación fuerte entre dos clases donde una clase es padre de otra. La herencia es un pilar importante de la POO. Es el mecanismo mediante el cual una clase es capaz de heredar todas las características (atributos y métodos) de otra clase. Las propiedades comunes se definen en la superclase (clase padre/madre) y las subclases heredan estas propiedades (Clase hija/o). En esta relación, la frase “Un objeto es un-tipo-de una superclase” debe tener sentido, por ejemplo: un perro es un tipo de animal, o también, una heladera es un tipo de electrodoméstico.
4. Polimorfismo: Este término se utiliza en POO para referirse a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos, es decir, que la misma operación se realiza en las clases de diferente forma. Estas operaciones tienen el mismo significado y comportamiento, pero internamente, cada operación se realiza de diferente forma. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía. Esto hace referencia a la idea de que podemos tener un método definido en la superclase y que las subclases tengan el mismo método, pero con distintas funcionalidades.

2) ¿Qué es una clase en Java?

Una clase en Java es una plantilla o modelo que define los atributos y métodos que tendrán los objetos creados a partir de ella. En el paradigma de la programación orientada a objetos, los programas se construyen en términos de objetos y sus relaciones, y las clases sirven para modelar esos objetos. Una clase puede contener variables (atributos), métodos, constructores, bloques estáticos y otros elementos que determinan el comportamiento y las propiedades de los objetos que se crearán a partir de dicha clase.

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

Un objeto en la programación es la forma que tenemos de llevar a código, la representación más fidedigna posible de un objeto de la vida real. Un objeto tendrá características (atributos) que son comunes a todos ellos, pero pueden variar entre sí. Pensemos también a las personas como objetos. Tenemos ciertas características que son uniformes a todos. Todos tenemos un nombre, un número de identificación, una fecha de nacimiento. Estas características variarían de una persona a otra, pero al compartir esta información podemos elaborar código que represente y nos permita identificar a una o varias personas.

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

Los tipos de operadores son los siguientes

1. Aritméticos: Encargados de realizar las operaciones matemáticas.

int suma = 5 + 3;

int resta = 5 - 2;

int multiplicacion = 4 \* 3;

1. Relacionales: Se encargan de comparar los valores y devuelven True o False.

boolean mayor = 8 > 5; (True)

boolean menor = 7 < 3; (False)  
boolean igual = 6 == 6; (True)

1. Lógicos: Se encargan de operar sobre los valores booleanos.

boolean resultado1 = (5 > 3) && (8 < 10); (True) boolean resultado2 = (5 > 3) || (2 > 3); (True)

1. Asignación: Se encarga de asignarles valores a las variables.  
   int x = 5;

x += 3;

x = 8;

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

Para lograr el uso correcto del encapsulamiento vamos a utilizar los modificadores de acceso, estos, van a dejarnos elegir como se accede a los datos y a través de que se accede a dichos datos. Todas las clases poseen diferentes niveles de acceso en función del modificador de acceso.

1. Public: Este modificador permite acceder a los elementos desde cualquier clase, independientemente de que esta pertenezca o no al paquete en que se encuentra el elemento.
2. Private: Es el modificador más restrictivo y especifica que los elementos que lo utilizan sólo pueden ser accedidos desde la clase en la que se encuentran. Este modificador sólo puede utilizarse sobre los atributos de una clase y sobre interfaces y clases internas, no sobre clases o interfaces de primer nivel, dado que esto no tendría sentido. Es importante destacar también que el modificador private convierte los elementos en privados para otras clases, no para otras instancias de la clase. Es decir, un objeto de una determinada clase puede acceder a los atributos privados de otro objeto de la misma clase.
3. Protected: Este modificador indica que los elementos sólo pueden ser accedidos desde su mismo paquete y desde cualquier clase que extienda la clase en que se encuentra, independientemente de si esta se encuentra en el mismo paquete o no. Este modificador, como private, no tiene sentido a nivel de clases o interfaces no internas.

Si no especificamos ningún modificador de acceso se utiliza el nivel de acceso por defecto, que consiste en que el elemento puede ser accedido sólo desde las clases que pertenezcan al mismo paquete.

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

Es una expresión que representa una función anónima, es decir, una función sin nombre que puede ser tratada como un valor. Las lambdas permiten una forma más concisa de definir comportamientos o funciones y son útiles principalmente para implementar interfaces funcionales, que son aquellas que tienen un solo método abstracto. Las lambdas simplifican el código eliminando la necesidad de clases internas anónimas y se utilizan comúnmente en expresiones de orden superior, como en colecciones y flujos.

Ejemplo:  
List<Integer> numeros = Arrays.asList(1, 2, 3, 4, 5); numeros.forEach(n -> System.out.println(n));

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

Las clases genéricas permiten crear estructuras de datos y métodos que pueden trabajar con cualquier tipo de objeto, lo que promueve la reutilización del código y la seguridad de tipo. Al instanciar una clase genérica, puedes especificar el tipo de datos que deseas utilizar, lo que te da flexibilidad al definir tus clases.

Ejemplo 1:  
class Caja<T> {

private T valor;

public void setValor(T valor) {

this.valor = valor;

}

public T getValor() {

return valor;

}

}

Ejemplo 2:  
class Contenedor<T> {

private T contenido;

public void setContenido(T contenido) {

this.contenido = contenido;

}

public T getContenido() {

return contenido;

}

}

Ejemplo 3:  
class Lista<T> {

private ArrayList<T> elementos = new ArrayList<>();

public void agregar(T elemento) {

elementos.add(elemento);

}

public T obtener(int indice) {

return elementos.get(indice);

}

}

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

Un hilo es una unidad de ejecución dentro de un proceso. Los hilos permiten que un programa realice múltiples tareas simultáneamente. En Java, se pueden crear hilos extendiendo la clase Thread o implementando la interfaz Runnable.

Ejemplo Thread:  
class MiHilo extends Thread {

public void run() {

System.out.println("Hilo en ejecución");

}

}

MiHilo hilo = new MiHilo();

hilo.start();

Ejemplo Runnable:  
class MiRunnable implements Runnable {

public void run() {

System.out.println("Hilo con Runnable en ejecución");

}

}

Thread hilo = new Thread(new MiRunnable());

hilo.start();

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

Una excepción es un evento anómalo que ocurre durante la ejecución de un programa, interrumpiendo el flujo normal de la ejecución. Las excepciones pueden ser causadas por errores en la lógica del programa, condiciones externas (como fallos en la red) o entradas de usuario inválidas. Java proporciona un mecanismo para manejar estas excepciones de manera controlada y evitar que el programa se detenga inesperadamente.

Se utilizan bloques try, catch y finally para manejar excepciones.

1. Try: Bloque donde se coloca el código que puede lanzar una excepción.
2. Catch: Bloque que captura y maneja la excepción. Puede haber múltiples bloques catch para manejar diferentes tipos de excepciones.
3. Finally: Bloque que se ejecuta siempre, independientemente de si ocurrió una excepción o no, ideal para liberar recursos.

10) ¿Qué es JSON?

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos, fácil de leer y escribir para humanos y fácil de analizar y generar para las máquinas. Se utiliza comúnmente para enviar y recibir datos en aplicaciones web, especialmente entre el cliente y el servidor.

Ejemplo:  
{

"nombre": "Eduardo",

"edad": 31,

"ciudad": "Buenos Aires",

"intereses": ["programación", "música", "deportes", "viajes"]

}

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 de Maven, la información general del proyecto se declara en el archivo pom.xml (Project Object Model). Este archivo contiene información sobre el proyecto, sus dependencias, plugins, y otras configuraciones.

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

1. Inversión de Control (IoC): Es un principio en el que el flujo de control de un programa es invertido. En lugar de que el programa controle el flujo, se delega a un contenedor o framework que gestiona la creación y la interacción de objetos. Esto facilita la gestión de la complejidad y promueve un diseño desacoplado.
2. Inyección de Dependencias (DI): Es un patrón que implementa la Inversión de Control, donde las dependencias de una clase se inyectan desde el exterior en lugar de que la propia clase las cree. Esto permite cambiar las dependencias sin modificar la clase.

13) ¿Qué es una interfaz?

Una interfaz en Java es una clase que contiene todos sus métodos abstractos y sus atributos son sólo constantes. En las interfaces se especifica qué se debe hacer, pero no su implementación o funcionalidad. Las clases que implementen una interface serán las responsables de describir la lógica del comportamiento de los métodos, las interfaces se implementan usando la palabra implements. La principal diferencia entre interface y abstract es que una interface proporciona un mecanismo de encapsulación de los métodos sin forzar a la utilización de una herencia.

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

Una anotación en Java es una forma de agregar metadatos a clases, métodos, atributos, etc. Las anotaciones no afectan la lógica del programa pero pueden ser utilizadas por herramientas de desarrollo o en tiempo de ejecución para modificar el comportamiento del código.

Ejemplos de anotaciones en Java:

1. @Override: Indica que un método está sobrescribiendo un método de la superclase.
2. @Deprecated: Indica que un método, clase o atributo está obsoleto y no debe usarse.
3. @SuppressWarnings: Indica que se deben suprimir ciertas advertencias de compilación.

class MiClase {

@Deprecated

public void metodoEjemplo() {

System.out.println("Este método es de Ejemplo.");

}

@Override

public String toString() {

return "MiClase";

}

}

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 registra los cambios en un conjunto de archivos o código a lo largo del tiempo, permitiendo revertir a versiones anteriores y colaborar con múltiples desarrolladores. Es fundamental en el desarrollo de software, ya que ayuda a mantener un historial del proyecto y facilita la colaboración.

1. Git: Una de las herramientas más populares para el control de versiones. Permite crear repositorios locales y remotos, realizar commits, ramas, merges, etc.
2. Subversion (SVN): Un sistema de control de versiones centralizado que permite gestionar archivos y directorios a lo largo del tiempo.
3. Mercurial: Un sistema de control de versiones distribuido similar a Git, que se enfoca en la eficiencia y simplicidad.

16) ¿Qué es un API?

Un API (Application Programming Interface) es un conjunto de definiciones y protocolos que permiten la comunicación entre diferentes aplicaciones o componentes de software. Un API define cómo se pueden hacer solicitudes y cómo se estructuran las respuestas. Facilita la integración y la interacción entre diferentes sistemas, aplicaciones o servicios.

17) ¿Qué es un servicio REST?

Un servicio REST (Representational State Transfer) es un estilo arquitectónico que permite la comunicación entre sistemas a través de HTTP. Los servicios REST utilizan métodos HTTP (GET, POST, PUT, DELETE) para realizar operaciones sobre recursos, que son representaciones de datos en formato JSON, XML, entre otros.

Ejemplo de un servicio RESTful:

1. GET /api/usuarios: Obtener una lista de usuarios.
2. POST /api/usuarios: Crear un nuevo usuario.
3. PUT /api/usuarios/1: Actualizar el usuario con ID 1.
4. DELETE /api/usuarios/1: Eliminar el usuario con ID

18) ¿Qué es un Endpoint?

Un endpoint es una URL que representa un recurso específico en un servicio web. Es el punto de acceso donde los clientes pueden interactuar con el servicio, enviando solicitudes y recibiendo respuestas.

Ejemplo de endpoints en un servicio RESTful:

1. /api/productos: Para obtener la lista de productos.
2. /api/productos/{id}: Para obtener, actualizar o eliminar un producto específico.

19) ¿Qué es un contrato de interfaz?

Un contrato de interfaz se refiere a las reglas que definen cómo los objetos de diferentes clases pueden interactuar entre sí a través de una interfaz. El contrato especifica los métodos que deben implementarse, pero no cómo se implementan. Esto permite que diferentes clases puedan ser utilizadas de manera intercambiable siempre que cumplan con el contrato de la interfaz.

20) ¿Qué es un Token JWT?

Un Token JWT (JSON Web Token) es un estándar abierto que define un formato compacto y autónomo para transmitir información entre partes como un objeto JSON. Se utiliza comúnmente para la autenticación y autorización en aplicaciones web. Un JWT se compone de tres partes: header, payload y signature.

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

1. Aplicación Monolítica: Es una única unidad de software que contiene todos los componentes (interfaz de usuario, lógica de negocio, acceso a datos) en un solo código base. Las actualizaciones o cambios en una parte de la aplicación requieren la reconstrucción y el redeployment de toda la aplicación.
2. Ventajas: Simplicidad en el desarrollo y despliegue. Facilidad de pruebas.
3. Desventajas: Escalabilidad limitada. Dificultad para realizar cambios, ya que el código está fuertemente acoplado.
4. Microservicio: Es un enfoque arquitectónico que divide una aplicación en servicios pequeños e independientes que pueden ser desarrollados, desplegados y escalados de manera independiente. Cada microservicio está diseñado para cumplir una función específica y se comunica con otros servicios a través de APIs.
5. Ventajas: Escalabilidad y flexibilidad. Cada servicio puede ser desarrollado en diferentes tecnologías. Mejor mantenimiento y despliegue.
6. Desventajas: Complejidad en la gestión y orquestación de servicios. Necesidad de establecer un protocolo de comunicación entre servicios.

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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La clase “Persona” cuenta con 2 atributos privados ‘nombre’ y ‘edad’. Estos atributos solo pueden ser accedidos o modificados desde dentro de la propia clase y para ello se proporcionan los métodos getter y setter.  
  
En la clase “ExamenTest” se crea el objeto ‘Persona’ llamado ‘p1’, con el nombre de ‘Juan’ y la edad de ‘20’.  
  
Sin embargo se intenta imprimir directamente el atributo edad de ‘p1’ el cual genera el siguiente error “  
  
Exception in thread "main" java.lang.RuntimeException: Uncompilable code - edad has private access in com.mycompany.test.arquitectura.de.credito.agil.Persona   
  
Esto se debe ya que el atributo “edad” cuenta con modificador de acceso “Private”, lo que significa que solo puede ser accedido desde dentro de la clase “Persona”, y no desde otras clases, en este caso la clase “ExamenTest”.  
  
Por lo tanto, el código no compila debido a un intento de acceso directo a un campo privado. La solución sería utilizar el método getEdad() para obtener el valor de la edad de forma segura.

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

Texto

Descripción generada automáticamente

La salida del código es la siguiente:  
“Bienvenid0”  
  
‘var1’ empieza con el texto “Bienvenido”.  
Luego, ‘var1.replace(“o”,”0”)’ cambia la “o” por un “0”, así que ahora ‘var1’ es “Bienvenid0”.  
Cuando uso concat para agregar ‘var2’ y ‘var3’, no se guardan en ‘var1’, por lo que no cambian su valor, concat simplemente crea una nueva cadena pero no le asigna ninguna variable.  
Por eso, cuando imprimo ‘var1’, solo se visualiza “Bienvenid0”.

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

Texto

Descripción generada automáticamente

El arreglo arreglo se inicializa con los valores {3, 9, 1, 8, 2, 7, 5, 8, 20, 10}.  
Luego utilizando “Arrays.asList(arreglo) se convierte a una lista. Se utiliza “stream()” para crear un flujo de datos de una lista.  
Se aplica el método “sorted” con una expresión lambda que compara los números de manera que el orden resultante sea descendente. “n2.compareTo(n1)” significa que el número más grande vendrá primero.  
Finalmente, se usa “forEach” para imprimir cada número de la lista ordenada, seguido de una coma y un espacio.

El código toma un conjunto de números desordenados y los ordena de mayor a menor.

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

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

“BancoAzteca” es una clase abstracta que define constructor y métodos getter y setter para sus atributos ‘nombreEmpleado’ e ‘id’. Como es una clase abstracta, no puede ser instanciada directamente.  
En “ExamenTest” se intenta crear un objeto ‘b1’ de la clase “BancoAzteca”, con el código:  
  
“BancoAzteca b1 = new BancoAzteca("Juan Perez", 1);”  
  
Pero es incorrecto porque no se puede instanciar una clase abstracta de esta manera, para que el código funcione correctamente se deben agregar llaves “{}” después de la creación del objeto para definir una clase anónima que extienda “BancoAzteca”, como se muestra a continuación:  
  
BancoAzteca b1 = new BancoAzteca("Juan Perez", 1) {};  
  
Con esta modificación el código ahora puede compilar y se puede crear un objeto de la clase. De este modo, luego se puede llamar al método “getNombreEmpleado()” de “b1”, que devuelve el valor del atributo, en este caso “Juan Perez”.

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

Texto

Descripción generada automáticamente

La salida del bloque de código es:

“Exceptionjava.lang.NullPointerException: Cannot invoke "String.concat(String)" because "str" is null”

La variable ‘str’ se inicializa como ‘null’, se intenta concatenar la cadena “Hola BancoAzteca” a la variable, sin embargo, dado que ‘str’ es ‘null’, no se puede invocar al método “concat()” en un objeto ‘null’.  
  
La excepción se captura en el bloque catch, donde se imprime el mensaje, esto indica que ha ocurrido una excepción, seguida del nombre de la excepción, que en este caso es “NullPointerException”.

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

Texto

Descripción generada automáticamente

Primera línea de salida (System.out.println(new Examen("Hola"));):

Se crea una nueva instancia de la clase Examen con el valor "Hola" para el atributo ‘banco’. Sin embargo, la clase Examen no sobrescribe el método toString().

Por defecto, el método toString() de la clase Object imprime el nombre de la clase seguido de un @ y el hashcode del objeto en hexadecimal. Por lo tanto, la salida es “com.banco.azteca.Examen@4517d93”.   
  
Segunda línea de salida (System.out.println(new String("Banco Azteca"));):

Aquí se crea una nueva instancia de String con el valor "Banco Azteca". La clase String en Java sobrescribe el método toString(), por lo que cuando imprimes un objeto String, se imprimirá su contenido directamente. La salida es “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](mailto:acamicroservicios@elektra.com.mx)