Examen diagnóstico para desarrollador Java – Arquitectura de Crédito Ágil

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

**Abstracción**: proceso de identificar las características esenciales de un objeto.

**Encapsulamiento**: proceso de ocultar los detalles internos de un objeto y solo mostrar operaciones públicas. Se encapsula teniendo a los atributos (variables de un objeto) como privados y los métodos que la controlan como públicos.

**Herencia**: mecanismo que permite que una clase (subclase) herede propiedades y comportamiento de otra clase (superclase). Esto se logra usando la palabra clave "extends" para clases o "implements" para interfaces.

**Polimorfismo**: permite que los objetos de diferentes clases sean tratados de manera uniforme si comparten una interfaz común. Donde los objetos pueden responder de manera diferente a la misma llamada del método.

**2) ¿Qué es una clase en Java?**

Es un modelo o plano para crear objetos. Define los datos (atributos) y el comportamiento (métodos) común a todos los objetos de ese tipo. Una clase encapsula los datos para el objeto y proporciona métodos para operar con esos datos.

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

Un objeto es una instancia de una clase, que tiene sus propios datos y puede realizar acciones específicas definidas por los métodos de la clase. Se declara creando una nueva instancia de la clase utilizando la palabra clave ‘new’, seguida del constructor de la clase, y luego se accede a los campos y métodos del objeto utilizando la notación de punto.

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

Operadores aritméticos: int resultado = 10 / 2;

Operadores de asignación: int x = 5;

Operadores de comparación: boolean resultado = (10 >= 8);

Operadores lógicos: boolean resultado = (true && false);

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

Un modificador de acceso es una palabra clave que se utiliza para controlar el acceso a los miembros (campos y métodos) de una clase desde otras partes del programa.

public: son accesibles desde cualquier clase. No hay restricciones en su acceso.

protected: son accesibles dentro de la misma clase, dentro del mismo paquete y por subclases (incluso si están en paquetes diferentes).

default (o package-private): sólo es accedido dentro del mismo paquete.

private: son accesibles sólo dentro de la misma clase. No son accesibles desde otras clases, incluso si pertenecen al mismo paquete o son subclases.

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

Las expresiones lambda son funciones anónimas que se utilizan para crear instancias de interfaces funcionales.  
Las funciones lambdas fueron agregadas en Java en su versión 8.

Una función lambda se define donde se usa.

**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 administrar tipos de datos distintos. Lo que permite que la clase sea reutilizable con diferentes tipos de datos, lo que proporciona flexibilidad y seguridad de tipos en tiempo de compilación.

|  |  |  |
| --- | --- | --- |
| Clase Lista genérica Nota: puede usar Integer o String | Clase ContenedorUnico genérico Nota: puede usar Integer o String | Clase ApiResponse genérico Nota: puede devolver objeto success o error |
| import java.util.ArrayList;  public 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);      }      public int tamaño() {          return elementos.size();      }  } | import java.util.HashSet;  import java.util.Set;  public class ContenedorUnico<T> {      private Set<T> elementos;      public ContenedorUnico() {          this.elementos = new HashSet<>();      }      public void agregarElemento(T elemento) {          elementos.add(elemento);      }      public boolean contieneElemento(T elemento) {          return elementos.contains(elemento);      }  } | public static class ApiResponse<T> {          private boolean success;          private T data;          public ApiResponse(boolean success, T data) {              this.success = success;              this.data = data;          }          public boolean isSuccess() {              return success;          }          public T getData() {              return data;          }      } |

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

Un hilo (thread) en Java es una secuencia de ejecución independiente dentro de un programa que puede ejecutarse simultáneamente con otros hilos. Los hilos permiten que un programa realice múltiples tareas simultáneamente y aproveche los recursos de procesamiento de manera más eficiente.

En Java, los hilos se pueden crear extendiendo la clase Thread o implementando la interfaz Runnable.

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

Una excepción en un evento que interrumpe el flujo normal de ejecución de un programa debido a una situación excepcional o un error. Se controlan usando bloques de Try-Catch.

A nivel de la aplicación o global se controlan usando el componente interceptor @ControllerAdvice o @RestControllerAdvice de Spring Boot.

Las excepciones personalizadas se manejan usando @ExceptionHandler y permite devolver un mensaje específico y hasta cambiar el estado de la respuesta.

**10) ¿Qué es JSON?**

Es un formato ligero de intercambio de datos que se utiliza para transmitir información estructurada entre un servidor y un cliente en la 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.?**

El archivo es el **build.gradle**.

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

La **Inversión de Control** es un principio de diseño de software que promueve el desacoplamiento y la modularidad al invertir el control del flujo de ejecución.   
En lugar de que cada componente invoque directamente sus dependencias, se utiliza un contenedor de IoC para gestionar la creación y gestión de las dependencias, haciendo uso del Constructor para la inyección de la dependencia.  
La **Inyección de Dependencias** es una técnica específica para implementar la Inversión de Control, permitiendo que las dependencias sean inyectadas en lugar de ser instanciadas dentro de los componentes.

**13) ¿Qué es una interfaz?**

Es una interfaz es una colección de métodos abstractos (sin implementación) que define un conjunto de comportamientos que una clase debe implementar.  
Las interfaces proporcionan un mecanismo para definir un contrato que las clases deben cumplir.

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

Es una forma de agregar metadatos, las anotaciones proporcionan datos sobre el código sin cambiar su comportamiento, y se utilizan para proporcionar información adicional, se definen con el símbolo **‘@’** seguido de su nombre, y pueden tener parámetros que proporcionan más detalles sobre la anotación.  
Entre las más usadas:  
@Override: se utiliza para indicar que un método está sobrescribiendo un método de la clase padre.  
@Autowired: se utiliza en Spring Framework para realizar la inyección de dependencias automáticamente  
@Value: para inyectar valores de algún fichero yml o properties  
@RequestMapping: para mapear peticiones HTTP a métodos de controlador.  
@ RequestBody: para acceder al cuerpo de una petición Rest.  
@RequestHeader: para acceder a las cabeceras HTTP.  
@Valid: para validar campos de entrada

@RequestParam: para recibir parámetros filtrados mediante URL

@PathVariable: para recibir parámetros fijos por URL

@RestController: es una especialización de controller en la capa de presentación para peticiones de tipo REST

@RequestMapping: usada para adicionar una ruta base para todos los métodos del controlador

@ControllerAdvice: para manejar las excepciones que se lanzan desde cualquier controlador.

**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 y gestiona los cambios realizados en el código y otros archivos, lo que permite a los equipos de desarrollo colaborar de manera efectiva y rastrear el historial de cambios.

Las herramientas mas conocidas son: Git, Subversion(SVN), Mercurial.

**16) ¿Qué es un API?**

Es una interfaz que define cómo interactuar con un software o servicio específico, permitiendo que otros sistemas o programas accedan a sus funciones y datos de manera controlada y segura.

**17) ¿Qué es un servicio REST?**

REST (Representational State Transfer) es una arquitectura para diseñar servicios web que utiliza el protocolo HTTP para la comunicación entre sistemas. Se comunican mediante verbos: GET, POST, PUT, PATCH, DELETE.

**18) ¿Qué es un Endpoint?**

Un endpoint es un punto final de comunicación en un servicio web o API que se utiliza para acceder a recursos o ejecutar operaciones específicas, por ejemplo a los recursos para gestionar usuarios: **GET /users**: (Para recuperar la lista de todos los usuarios).

**19) ¿Qué es un contrato de interfaz?**

Un contrato de interfaz es un acuerdo formal sobre cómo interactuar con un servicio o API. En el contexto de las API RESTful, un contrato de interfaz define los endpoints disponibles, los parámetros que aceptan y los formatos de datos esperados en las solicitudes y respuestas. Lo ideal es usar OpenAPI es una especificación que se utiliza para describir y documentar APIs de manera estándar. Define un formato para describir los endpoints disponibles, los parámetros que aceptan y los formatos de datos esperados en las solicitudes y respuestas.

**20) ¿Qué es un Token JWT?**

Es un estándar abierto que define un formato compacto y seguro para transmitir información entre dos partes. Utiliza un token se para la autenticación y la autorización en aplicaciones web y servicios API.

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

**Arquitectura**:

* Monolítica: Todos los componentes están integrados en una sola unidad.
* Microservicios: La aplicación se divide en servicios independientes que funcionan de forma separada.

**Despliegue**:

* Monolítica: Se despliega como una sola entidad.
* Microservicios: Cada servicio puede ser desplegado de manera independiente.

**Tecnología**:

* Monolítica: Todos los componentes comparten la misma tecnología y lenguaje de programación.
* Microservicios: Cada servicio puede utilizar diferentes tecnologías y lenguajes de programación según sus necesidades.

**Escalabilidad**:

* Monolítica: Escalar significa replicar toda la aplicación.
* Microservicios: Puedes escalar servicios específicos según la demanda.

**Mantenimiento**:

* Monolítica: Los cambios pueden ser más complejos debido a la interdependencia de los componentes.
* Microservicios: Los cambios pueden ser más fáciles de implementar ya que los servicios son independientes.

**Acoplamiento**:

* Monolítica: Alta cohesión y acoplamiento entre los componentes.
* Microservicios: Menor acoplamiento entre los servicios debido a su independencia.

**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 salida es:** 20

Porque aquí solo se está imprimiendo la edad de la persona

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

Texto

Descripción generada automáticamente

**La salida es**: Bienvenid0

El motivo porque no muestra la concatenación completa es porque las llamadas a concat no se están asignando a ninguna variable.

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

Texto

Descripción generada automáticamente

**La salida es**:

20 ,

10 ,

9 ,

8 ,

8 ,

7 ,

5 ,

3 ,

2 ,

1 ,

La salida del bloque de código es una lista de números ordenados de forma descendente. La lista está compuesta por los números del arreglo arreglo, ordenados 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

**La salida es**: Juan Perez

Se debe a que se creo una instancia de la clase BancoAzteca donde el empleado con el nombre Juan Perez es el que se solicita imprimir.

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

Texto

Descripción generada automáticamente

**La salida es**: Exception: java.lang.NullPointerException

Porque cuando se intenta llamar al método concat en esta cadena null, se produce una excepción NullPointerException porque no se puede invocar un método en una referencia nula.

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

Texto

Descripción generada automáticamente

**La salida es**:

com.example.demo.Examen@39fb3ab6

Banco Azteca

Como no se ha sobrescrito el método toString() en la clase Examen, se imprime el nombre completo de la clase seguido del código hash del objeto.  
En comparación con la siguiente línea que crea una instancia de String con el valor "Banco Azteca". La clase String tiene un método toString() sobrescrito que devuelve el contenido de la cadena.

**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)