22 de mayo de 2024

Luis Alberto García Reyes

Xideral

Preguntas de Examen

Parte 2

Preguntas de examen

# 56.- Son patrones de diseño de microservicios

1. Circuit Breaker, Adaptative Lifo, MQ Strategy
2. System, Process y Client
3. Retry, Circuit Breaker, Adaptative Lifo y Bulkhead.
4. Ninguna de las anteriores

**Respuesta**: Opción C) Retry, Circuit Breaker, Adaptative Lifo y Bulkhead.

**Explicación**:

Las otras opciones no son correctas porque:

a) Circuit Breaker, Adaptative Lifo, MQ Strategy: "MQ Strategy" no es un patrón de diseño de microservicios reconocido. Esta opción mezcla algunos patrones válidos con uno incorrecto.

b) System, Process y Client: Estos no son patrones de diseño de microservicios. Son conceptos más generales relacionados con sistemas y procesos, pero no patrones específicos de microservicios.

d) Ninguna de las anteriores: Esta opción no es correcta, ya que la opción c) sí enumera patrones de diseño de microservicios válidos.

Los patrones de diseño de microservicios son estrategias específicas que abordan desafíos comunes en este tipo de arquitectura, como la tolerancia a fallas, la escalabilidad, la resiliencia y la comunicación entre servicios independientes. Los patrones mencionados en la opción c) (Retry, Circuit Breaker, Adaptative Lifo y Bulkhead) son ampliamente reconocidos y utilizados en el diseño de microservicios.

Otros patrones de microservicios populares que podrían mencionarse son: API Gateway, Service Discovery, Event Sourcing, CQRS (Command Query Responsibility Segregation), entre otros. Pero la opción c) presenta un conjunto válido y reconocido de patrones de diseño de microservicios.

# 57.- ¿Qué afirmaciones son verdaderas tanto para las clases abstractas como para las interfaces? (Elije todas las correctas)

1. Ambos pueden contener métodos estáticos.
2. Ambos se pueden ampliar con la clave extend.
3. Ambos pueden contener métodos predeterminados.
4. Ambos heredan de java.lang.Object.
5. Ninguno de los dos puede ser instanciado directamente.
6. Ambos pueden contener variables finales estáticas públicas.
7. Supone que todos los métodos dentro de ellos son abstractos.

**Respuesta**: Opciones a) b) e) y f)

**Explicación**:

Analizando detalladamente cada opción.

1. SI las **clases abstractas** pueden declarar y definir métodos estáticos. A partir de Java 8, se permite definir métodos estáticos dentro de las **interfaces**, y no hay restricciones en cuanto a la cantidad de métodos estáticos que una interfaz puede tener
2. Cuando nos referimos a ampliar es que las clases o interfaces están adquiriendo nuevos métodos de una interfaz o de una clase, sin embargo, a pesar de que **ambos pueden ampliarse con extends** las interfaces pueden heredar de múltiples interfaces, pero las clases no, solo pueden usar una herencia y debe de ser de una clase ya que para las clases si quieren heredar interfaces deben usar implements y ahí sí pueden implementar multiples interfaces.
3. El término "**método predeterminado" es específico de las interfaces** porque fue introducido para resolver un problema particular: permitir la evolución de interfaces sin romper la compatibilidad con las implementaciones existentes. En las **clases abstractas, esta necesidad no existe** porque siempre han podido tener métodos con implementación (métodos concretos). Aunque técnicamente funcionan para lo mismo por el concepto y la definición no lo son y por lo tanto esta opción no es correcta.
4. Interfaz no Extiende Object: A nivel de sintaxis, las **interfaces en Java no extienden explícitamente java.lang.Object**. No se declara nada como public interface MyInterface extends Object. Implícita Disponibilidad de Métodos de Object: Sin embargo, cualquier clase que implemente una interfaz tiene acceso a los métodos de java.lang.Object. Esto es porque todas las clases en Java (incluso las que implementan interfaces) heredan de java.lang.Object.
5. Tanto las clases abstractas como las interfaces **no pueden ser instanciadas directamente**. Las clases abstractas necesitan ser extendidas y las interfaces necesitan ser implementadas.
6. **Verdadero**. Tanto las clases abstractas como las interfaces pueden contener variables que son final, static, y public.
7. **Falso**. Las clases abstractas pueden contener métodos concretos (no abstractos). Las interfaces, desde Java 8, también pueden contener métodos predeterminados y estáticos que no son abstractos.

# 58.- ¿Cuál no es un objetivo de Maven?

1. Clean
2. Package
3. Debug
4. Install

**Respuesta**: Opción c) debug

**Explicación**:

El debug no es un objetivo directo de Maven. Maven no está diseñado para depurar aplicaciones, sino para construir y gestionar el ciclo de vida del proyecto. La depuración de aplicaciones es una tarea que se realiza típicamente dentro de un entorno de desarrollo integrado (IDE) como IntelliJ IDEA, Eclipse o NetBeans.

Ciclo de vida de Maven

* validate
* compile
* test
* package
* verify
* install
* deploy

Ciclos de vida separados

* default (o ciclo de vida de construcción)
* clean
* site

# 59.- ¿Si deseas obtener una copia de un repositorio Git existente en un servidor qué comando se utiliza?

1. git commit
2. git log
3. git clone
4. git add

**Respuesta**: Opción c) git clone

**Explicación**:

Para obtener una copia de un repositorio Git existente en un servidor, se utiliza el comando **git clone**. Este comando clona un repositorio remoto en tu máquina local, creando una copia completa del repositorio, incluyendo todos los archivos, ramas y el historial de commits.

# 60.- ¿Qué es un repositorio remoto en Git?

1. Una herramienta que se utiliza para compartir y fusionar cambios entre diferentes ramas de un repositorio.
2. Una copia local de un repositorio que se utiliza para hacer cambios en el código fuente.
3. Un servidor Git que almacena una copia central del repositorio.
4. Un archivo que contiene una instantánea del código fuente en un momento determinado.

**Respuesta**: Opción c)

**Explicación**:

Un repositorio remoto en Git es un **repositorio alojado en un servidor** o en la nube, accesible a través de la red. Permite la colaboración entre diferentes desarrolladores al proporcionar una ubicación centralizada para clonar, obtener, fusionar, tirar y empujar cambios. Los comandos más comunes para interactuar con repositorios remotos incluyen clone, fetch, pull, push, y remote.

# 61.- Dada la siguiente clase

Texto

Descripción generada automáticamente

¿Cuál de las siguientes instrucciones puede colocarse en la línea 9 para que la clase Helper compile?

1. Helper.printException(new Exception(“B”));
2. Helper. printException(new FileNotFoundException(“A”));
3. Helper.<Throwable>printException(new Exception(“C”));
4. Helper.<NullPointerException>printException(new NullPointerException (“D”));
5. Helper. printException(new Throwable(“E”));

**Respuesta**: Opciones a) b) y d)

**Explicación**:

La clase Throwable no cumple con estas restricciónes, ya que es la superclase de Exception, pero no es una subclase de Exception.

Por lo tanto, intentar pasar un objeto Throwable al método printException resultará en un error de compilación, ya que viola la restricción genérica impuesta en la declaración del método.

# 62.- ¿Cuál es la salida al ejecutar el siguiente código?

Texto

Descripción generada automáticamente

1. 51tuna
2. 5tuna
3. 5
4. 41
5. 5 tuna
6. 4 1
7. El código no compila

**Respuesta**: Opción G) el código no compila

**Explicación**:

La variable String anotherFish = numFish +1; es incompatible con los datos numéricos como int por lo tanto el compilador mandara un problema de error de incompatibilidad. Para corregir ese error habrá que castear el contenido de la variable anotherFish con Integer.toString();

# 63.- ¿Cuál de las siguientes opciones son correctas? (Elija todas las correctas)

Texto

Descripción generada automáticamente

1. s2 = s2
2. s3 = null
3. s1 = s1
4. s3 = a
5. El código no compila
6. s2 = s2b
7. s1 = a

**Respuesta**: Opciones c) d) y f)

**Explicación**:

Este es un código que utiliza tres variables de tipo StringBuilder para manipular cadenas de caracteres de manera mutable. En principio cuando en el método main se declaran e inicializan las cadenas, su contenido no muta, no es hasta que se llama al método work. El método work recibe como parámetros a s1 y s2 con sus propias variables a y b, en principio, aunque a cambia su valor esto no afecta a s1 porque está dentro del método, pero cuando b usa el método append es cuando s2 cambia su valor entonces s1 se queda con su valor original, s2 se le asigna su valor original más el carácter b y s3 se le asigna el valor obtenido del método work.

# 64.- ¿A qué hace referencia el principio de Liskov?

1. Nos indica que una clase no debe tener solo una funcionalidad sino varias para reducir el uso de objetos.
2. Este principio nos indica que dentro del programa una clase puede ser sustituida por cualquier clase que se extienda de ella sin alterar el comportamiento del programa.
3. Nos indica que cualquier clase se puede extender para agregar funcionalidad, pero no se puede modificar.
4. Este principio nos indica que dentro del programa una clase puede ser sustituida por su clase padre sin alterar el comportamiento del programa.

**Respuesta**: Opción b) Este principio nos indica que dentro del programa una clase puede ser sustituida por cualquier clase que se extienda de ella sin alterar el comportamiento del programa.

**Explicación**:

El Principio de Sustitución de Liskov (Liskov Substitution Principle, LSP) es uno de los cinco principios de diseño SOLID que proponen buenas prácticas para el desarrollo de software orientado a objetos. Formulado por la científica de la computación Barbara Liskov en 1987, este principio establece que:

*"Los objetos de una clase derivada deben poder sustituir a objetos de la clase base sin alterar el correcto funcionamiento del programa."*

En otras palabras, si S es una subclase de T, entonces los objetos de tipo T en un programa pueden ser reemplazados por objetos de tipo S sin cambiar ninguna de las propiedades deseables de ese programa (correctitud, funcionalidad, etc.).

# 65.- ¿Qué es un “code smell”?

1. Un componente de la biblioteca estándar de Java
2. Un error en tiempo de compilación que se produce en Java
3. Un indicador de que puede haber un problema en el código que puede ser dificil de detectar o que podría ser una fuente potencial de errores o problemas de mantenimiento en el futuro.
4. Una práctica de programación recomendada en Java.

**Respuesta**: Opcion c) Un indicador de que puede haber un problema en el código que puede ser dificil de detectar o que podría ser una fuente potencial de errores o problemas de mantenimiento en el futuro.

**Explicación**:

Es un término utilizado en el desarrollo de software para describir **cualquier indicio en el código que sugiere la existencia de un problema de diseño subyacente**. Aunque un code smell no es necesariamente un error que afecte el funcionamiento inmediato del programa, sí indica que hay algo en el código que puede llevar a problemas futuros, como dificultad para entender el código, aumentar la complejidad, errores ocultos, o dificultades en el mantenimiento y la extensión del código.

# 66.- ¿Qué significa el acrónimo CRUD en una API REST?

1. Code, Register, Update, Debug
2. Create, Read, Update, Delete
3. Call, Receive, Use, Debug
4. Customize, Request, Use, Debug

**Respuesta**: Opcion b) Create Read Update Delete

**Explicación**:

CRUD es un acrónimo que describe las operaciones básicas que se pueden realizar en una base de datos o una API (Application Programming Interface). CRUD representa las cuatro funciones principale: Crear (Create), Leer (Read), Actualizar (Update) y Eliminar (Delete). Estas operaciones se mapean a los métodos HTTP en una API RESTful de la siguiente manera:

* **Create** (Crear): Se utiliza el método HTTP POST para crear un nuevo recurso.
* **Read** (Leer): Se utiliza el método HTTP GET para leer o recuperar un recurso.
* **Update** (Actualizar): Se utiliza el método HTTP PUT o PATCH para actualizar un recurso existente.
* **Delete** (Eliminar): Se utiliza el método HTTP DELETE para eliminar un recurso

# 67.- ¿Para qué nos sirve utilizar un profile dentro del archivo pom.xml?

1. Etiqueta por la cual podemos definir la versiones de nuestras dependencias.
2. Es la etiqueta por la cual podemos definir las características que tendrá nuestro proyecto al ser compiladas.
3. Etiqueta por la cual definimos los parámetros de conexión a un repositorio.
4. No existe esta etiqueta en Maven.

**Respuesta**: Opción b) Es la etiqueta por la cual podemos definir las características que tendrá nuestro proyecto al ser compiladas.

**Explicación**:

Es una forma de **personalizar** la construcción de tu proyecto. Los perfiles en el archivo pom.xml permiten **definir configuraciones** de construcción alternativas que se pueden activar en función de ciertos criterios, como el entorno (desarrollo, prueba, producción), parámetros del sistema, o argumentos de la línea de comandos.

# 68.- Dadas las siguientes clases Vehicle y Car

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

¿Cuál de las siguientes instrucciones pueden colocarse en la línea 7 para que la clase Car compile correctamente? (Seleccione las que apliquen)

1. mileage = 15285;
2. Ninguna de las anteriores.
3. make = “Honda”;
4. year = 2009;
5. model = “Pilot”;

**Respuesta**: Opciones c) y e)

**Explicación**:

El código nos pregunta que respuestas podemos colocar en la clase Car para que compile correctamente dándonos 5 opciones, tenemos que fijarnos en el nivel de alcance de las variables de la clase Vehicle. Entonces la opción a no es posible porque la variable mileage tiene por defecto el alcance de default entonces solo puede compartir su información con clases del mismo paquete, la opción b no es correcta porque si hay opciones disponibles, la opción c es correcta porque el alcance de la variable make es público, la opción d no es posible porque year es private y finalmente model es posible porque aunque es protected puede ser utilizado por clases de otros paquetes siempre que se hereden correctamente de la clase Vehicle.

# 69.- Enumere cuatro interfaces de la API colecciones

1. List, Map, Set, Queue.
2. ArrayList, Map, Set, Queue.
3. List, HashMap, HashSet, PriorityQueue.
4. List, Map, HashSet, PriorityQueue.

**Respuesta**: Opcion a) List Map Set Queue

**Explicación**:

Las interfaces registradas para Collections son:

1. Collection<E>
2. **List**<E>
3. **Set**<E>
4. SortedSet<E>
5. NavigableSet<E>
6. **Queue**<E>
7. Deque<E>
8. **Map**<K, V>
9. SortedMap<K, V>
10. NavigableMap<K, V>

# 70.- Selecciona la respuesta correcta con respecto al resultado del bloque de código.

Texto

Descripción generada automáticamente

1. Head  
   Object  
   Tail  
   Side
2. No compila
3. Side  
   Object  
   Tail  
   Side
4. Head  
   Head  
   Tail  
   Tail
5. Side  
   Head  
   Tail  
   Side

**Respuesta**: Opcion c)

**Explicación**:

Este código muestra cómo diferentes tipos de referencias afectan a qué método sobrecargado se llama en Java.

Análisis

Llamada Coin.overload(primerIntento);:

* primerIntento es de tipo **Side** y referencia un objeto de tipo Head.
* El método overload(Side side) es el que mejor coincide.
* Salida: Head 3.

Llamada Coin.overload((Object)segundoIntento);:

* segundoIntento es de tipo Tail pero se convierte a **Object**.
* El método overload(Object side) es el que mejor coincide.
* Salida: Object.

Llamada Coin.overload(segundoIntento);:

* segundoIntento es de tipo **Tail**.
* El método overload(Tail side) es el que mejor coincide.
* Salida: Tail 2.

Llamada Coin.overload((Side)primerIntento);:

* primerIntento es de tipo **Side** y referencia un objeto de tipo Head, pero está explícitamente convertido a Side.
* El método overload(Side side) es el que mejor coincide.
* Salida: Head 3.

# 71.- ¿Cuál es la salida al ejecutar el siguiente código?

Texto, Carta

Descripción generada automáticamente

1. roar roar!!!
2. roar!!! Roar
3. Se lanza una excepción
4. roar!!! roar!!!
5. roar roar
6. El código no compila

**Respuesta**: Opcion a) roar roar!!!

**Explicación**:

El ejercicio aborda temas sobre los Strings y StringBuilder y cómo funcionan sus respectivos métodos. concat() y .append(). Como String es inmutable, pero StringBuilder es mutable entonces roar1 mantiene su valor a pesar del método concat porque no modifica la cadena original y el método roar, roar1.concat("!!!") crea una nueva cadena "roar!!!" pero no asigna este nuevo valor a roar1. Por lo tanto, roar1 sigue apuntando a la cadena original "roar".

A diferencia de roar1, roar 2 si cambia porque es mutable, precisamente en el método roar, roar2.append("!!!") modifica el objeto roar2 directamente, añadiendo "!!!" al final de la cadena que contiene.

# 72.- ¿Cuál de los siguientes es cierto acerca de una subclase concreta?

1. Una subclase concreta no se puede marcar como final.
2. Una subclase concreta debe implementar todos los métodos definidos en una interfaz heredada.
3. Una subclase concreta debe implementar todos los métodos abstractos heredados.
4. Una subclase concreta puede declararse como abstracta.
5. Los métodos abstractos no pueden ser anulados por una subclase concreta.

**Respuesta**: Opción c) Una subclase concreta debe implementar todos los métodos abstractos heredados.

**Explicación**:

* Opcion a) Esto es incorrecto. Una subclase concreta puede ser marcada como final, lo que significa que no puede tener subclases.
* Opcion b) Incorrecto: Si una clase es abstracta, no puede ser concreta. Una clase abstracta no implementa todos los métodos abstractos heredados, por lo que no puede ser instanciada directamente.
* Opcion c) Esto es correcto. Una subclase concreta debe proporcionar implementaciones para todos los métodos abstractos heredados de sus clases base abstractas.
* Opcion d) Esto es incorrecto. Si una clase es declarada como abstracta, no puede ser concreta porque una clase abstracta puede contener métodos abstractos (sin implementación), mientras que una subclase concreta debe proporcionar implementaciones para todos los métodos abstractos heredados.
* Opcion e) Esto es incorrecto. Los métodos abstractos son precisamente aquellos que deben ser implementados (anulados) por una subclase concreta.

# 73.- ¿Cuál es la salida del siguiente código?

Una captura de pantalla de una computadora

Descripción generada automáticamente con confianza media

1. Error de compilación línea 12
2. Error compilación línea 16
3. Caught string
4. Error compilación línea 2
5. Caught Object

**Respuesta**: opción e) Caught Object

**Explicación**:

El método que se ejecuta es catchAnObject(Object x) porque es la implementación que coincide con el tipo de argumento java.util.Date pasado a él. El método catchAnObject(String s) no se considera en este contexto.

# 74.- Seleccione la respuesta que considere correcta, dado el siguiente bloque de código.

Texto

Descripción generada automáticamente

1. 3
2. 9
3. 14
4. 6

**Respuesta**: opción d) 6

**Explicación**:

El código crea una lista de números, filtra los números pares y luego suma esos números, imprimiendo el resultado. En este caso, imprimirá la suma de los números pares del 2 al 4, que es 6.

* Crea una lista de números enteros llamada numbers utilizando el método Arrays.asList() para inicializarla con los valores 1, 2, 3, 4 y 5.
* Utiliza el método stream() sobre la lista numbers para obtener un stream de elementos.
* Llama al método filter() del stream para filtrar los números pares, es decir, aquellos números que al dividirlos por 2 su residuo es 0.
* Utiliza el método reduce() para reducir los elementos del stream a un solo valor. El valor inicial para la reducción es 0, y la operación de reducción suma los elementos (a + b).

# 75.- ¿QUÉ DECLARACIÓN REPRESENTA UNA DECLARACIÓN VÁLIDA QUE PERMITIRÁ LA INCLUSIÓN DE CLASES DEL PAQUETE JAVA?util?

1. #include java.util.\*;
2. #include java.util;
3. import java.util.\*;
4. import java.util;

**Respuesta**: Opción c) import java.util.\*;

**Explicación**:

La declaración correcta para la importación de paquetes en java no comienza con # además debemos de especificar si estamos importando todo el paquete o parte de el, ya sea con punto y asterisco para todo, o punto y la clase especifica.

# 76.- ¿Qué es la cobertura de código?

1. La cantidad de veces que se ejecuta una línea de código.
2. La cantidad de errores detectados por una prueba.
3. La cantidad de código que se ejecuta durante una prueba.
4. La cantidad de tiempo que tarda una prueba en ejecutarse.

**Respuesta**: Opción c) La cantidad de código que se ejecuta durante una prueba.

**Explicación**:

La cobertura de código, en el contexto del desarrollo de software, es una medida que indica **qué porcentaje del código fuente ha sido ejecutado durante la ejecución de pruebas automatizadas**. En otras palabras, la cobertura de código indica qué tan completa es la suite de pruebas en términos de qué parte del código ha sido ejecutada y qué parte no.

# 77.- ¿Cuál es el formato correcto para hacer un commit en Git?

1. Descripción breve del cambio y nombre del autor.
2. Tipo de cambio, descripción breve, cuerpo opcional y notas de pie de página.
3. Solo se necesita una breve descripción del cambio.
4. Nombre de la rama, descripción detallada del cambio y fecha.

**Respuesta**: Opción b)

**Explicación**:

El formato típicamente consiste en dos partes: el encabezado y el cuerpo, opcionalmente separados por una línea en blanco.

<tipo>(<área>): <descripción>

<Cuerpo opcional>

# 78.- ¿Qué es el patrón de diseño Singleton y cómo se implementa en Java 8?

1. El patrón de diseño Singleton es un patrón que se utiliza para garantizar que una clase tenga una única instancia en todo el sistema. Se implementa utilizando una variable estática y un constructor privado.
2. El patrón de diseño Singleton es un patrón que se utiliza para abstraer la capa de infraestructura en una aplicación. Se implementa utilizando excepciones y bloques try-catch.
3. El patrón de diseño Singleton es un patrón que se utiliza para abstraer la capa de presentación en una aplicación. Se implementa con interfaces y clases concretas.
4. El patrón de diseño Singleton es un patrón que se utiliza para abstraer la capa de negocios en una aplicación. Se implementa utilizando clases abstractas y métodos estáticos.

**Respuesta**: Opcion a) El patrón de diseño Singleton es un patrón que se utiliza para garantizar que una clase tenga una única instancia en todo el sistema. Se implementa utilizando una variable estática y un constructor privado.

**Explicación**:

El patrón de diseño Singleton es un patrón de creación que **garantiza que una clase tenga solo una instancia y proporciona un punto global de acceso a esa instancia**. Es útil cuando solo se necesita una instancia de una clase para **coordinar acciones en todo el sistema**.

La implementación del patrón Singleton garantiza que solo haya una **instancia** de la clase Singleton en todo el programa y proporciona un método estático para acceder a esa instancia

# 79.- En los verbos REST ¿Cuál es la diferencia en el uso de PATCH y PUT?

1. Son exactamente iguales, no hay diferencia de uso.
2. PATCH requiere se le envíe la entidad completa mientras que PUT solo los atributos a modificar.
3. PUT requiere se le envíe la entidad completa mientras que PATCH solo los atributos a modificar.
4. PATCH es un verbo deprecado sustituido por PUT.

**Respuesta**: Opción PUT requiere se le envíe la entidad completa mientras que PATCH solo los atributos a modificar.

**Explicación**:

La diferencia principal entre los métodos PUT y PATCH en el contexto de REST radica en cómo manejan las actualizaciones de recursos:

PUT: Cuando se utiliza el método PUT, se espera que el **cliente envíe una representación completa del recurso que se desea actualizar al servidor**. Esta representación completa reemplaza por completo el recurso existente en el servidor. Por lo tanto, PUT se utiliza para actualizar el recurso completo.

PATCH: Por otro lado, cuando se utiliza el método PATCH, el cliente **solo necesita enviar los atributos que desea modificar en el recurso,** junto con los nuevos valores. No es necesario enviar una representación completa del recurso. El servidor aplicará los cambios especificados en la solicitud PATCH al recurso existente, modificando solo los campos indicados. PATCH se utiliza para realizar actualizaciones parciales en un recurso.

# 80.- ¿Cuál es la diferencia entre las anotaciones: @RestController, @Component, @Service y @Repository?

1. @Controller es una anotación que nos ayuda a construir una API REST mientras que @Service, @Component y @Repository solo marcan las clases que se deben de inicializar.
2. @Controller, @Component son anotaciones que crean bean y exponen la serialización de las clases mientras que @Service y @Repository requieren de una inicialización manual.
3. No existe diferencia funcional entre ellas sino semántica, las 4 son anotaciones de Spring que crean un bean y lo agregan al contexto de Spring.
4. @Service y @Repository son anotaciones que crean un bean y exponen la serialización de las clases mientras que @Controller. @Component requiere de una inicialización manual.

**Respuesta**: Opción c) No existe diferencia funcional entre ellas sino semántica, las 4 son anotaciones de Spring que crean un bean y lo agregan al contexto de Spring.

**Explicación**:

Todas estas anotaciones se utilizan en el marco de trabajo Spring para marcar clases como beans de Spring y hacer que se inicialicen automáticamente en el **contexto de la aplicación**. Sin embargo, tienen **diferentes significados semánticos** que indican el propósito general de la clase anotada.

* @**Component**: Es una anotación genérica para marcar una clase como un componente de Spring. Se utiliza para clases que no encajan en ninguna de las otras categorías más específicas.
* @**Service**: Se utiliza para marcar una clase como un servicio de la capa de negocio. Semánticamente, indica que la clase representa un servicio de la lógica de negocio de la aplicación.
* @**Repository**: Se utiliza para marcar una clase como un repositorio de acceso a datos. Semánticamente, indica que la clase se encarga de interactuar con la capa de persistencia de datos.
* @**RestController**: Es una anotación específica de Spring MVC que combina las anotaciones @**Controller** y @**ResponseBody**. Se utiliza para marcar una clase como un controlador web que maneja solicitudes HTTP y devuelve respuestas en formato JSON o XML.

Aunque estas anotaciones tienen diferentes significados semánticos, funcionalmente todas ellas hacen que Spring cree un bean a partir de la clase anotada y lo agregue al contexto de la aplicación. **La elección de qué anotación utilizar depende del propósito y la semántica de la clase en cuestión.**

# 81.- ¿Cuál es una buena práctica al escribir pruebas unitarias?

1. Ejecutar pruebas con poca frecuencia.
2. Asegurarse de que las pruebas sean claras y concisas.
3. Probar solo una pequeña parte de una función.
4. Hacer que las pruebas dependan de otras pruebas.

**Respuesta**: Opción b) Asegurarse que las pruebas sean claras y concisas.

**Explicación**:

Escribir pruebas unitarias claras y concisas es una buena práctica por varias razones:

* Legibilidad
* Mantenibilidad
* Enfoque
* Reutilización

Las otras opciones no son buenas prácticas:

# 82.- ¿Cuál es la ventaja de usar APIs REST sobre otros tipos de servicios web?

1. Mayor seguridad.
2. Mayor facilidad de implementación.
3. Mayor velocidad de transferencia de datos.
4. Mayor compatibilidad con diferentes plataformas.

Respuesta: Opción b)

Explicación:

Una de las principales ventajas de utilizar APIs RESTful (Representational State Transfer) sobre otros tipos de servicios web es su **capacidad de ser independientes de cualquier tecnología o plataforma en particular en este caso a la plataforma y al lenguaje de programación.**

Sin embargo, una ventaja de REST es que es más fácil de utilizar e implementar con respecto a SOAP su predecesor, sin embargo podemos tener en cuenta la habilidad del programador o no, para un principiante por más fácil que sea podría ser difícil aun asi.

# 83.- Selecciona la respuesta correcta con respecto al resultado del bloque de código.

Texto

Descripción generada automáticamente

1. No compila
2. [25, 7, 25, 67]

[67, 7, 25]

[7, 25, 67]

[67, 7, 25]

[7, 25, 67]

1. [25, 7, 67]  
   [67, 7, 25]  
   [7, 25, 67]  
   [67,7, 25]  
   [7, 25, 67]
2. [67, 7, 25]  
   [67, 7, 25]  
   [67, 7, 25]  
   [67, 7, 25]  
   [67, 7, 25]
3. [25, 7, 25, 67]  
   [7, 25, 67]  
   [67, 7, 25]  
   [7, 25, 67]  
   [67, 7, 25]

**Respuesta**: Opción b)

**Explicación**:

El ejercicio consiste en los ArrayList y sus métodos correspondientes:

* **HashSet** es adecuado cuando no se requiere un orden específico y se necesita un alto rendimiento.
* **TreeSet** es útil cuando se necesita un conjunto ordenado y operaciones basadas en rangos.
* **ConcurrentSkipListSet** es una buena opción para aplicaciones concurrentes que requieren un conjunto ordenado y operaciones seguras para subprocesos.

# 84.- ¿Cuáles son los 4 pilares de la programación orientada a objetos?

1. Polimorfismo, Coerción, Herencia y Encapsulamiento.
2. Encapsulamiento, Coerción, Polimorfismo y Abstracción.
3. Polimorfismo, Herencia, Encapsulamiento y Sincronía.
4. Polimorfismo, Abstracción, Herencia y Encapsulamiento.

**Respuesta**: opción Polimorfismo, Abstracción, Herencia y Encapsulamiento.

**Explicación**: Los cuatro pilares de la Programación Orientada a Objetos de Java son:

* Encapsulación (Encapsulation)
* Abstracción (Abstraction)
* Herencia (Inheritance)
* Polimorfismo (Polymorphism)

# 85.- ¿Qué utilidad de línea de comandos basada en MS Windows le permitirá ejecutar el intérprete de Java sin abrir la ventana de la consola?

1. jconsole
2. javaw
3. interpw
4. java -wo

**Respuesta**: opción b) javaw

**Explicación**:

**javaw** es un comando en sistemas Windows que ejecuta programas Java en un entorno de ventana sin abrir una ventana de consola. Es similar al comando java, pero javaw se utiliza específicamente para **aplicaciones que no necesitan interacción directa con la consola de comandos**.

# 86.- ¿Qué es un endpoint en una API REST?

1. Un endpoint es un objeto que se utiliza para almacenar datos en una API REST.
2. Un endpoint es un método que se utiliza para procesar datos en una API REST.
3. Un endpoint es un controlador que se utiliza para administrar una API REST.
4. Un endpoint es la URL que se utiliza para acceder a una API REST.

**Respuesta**: Opción d) Un endpoint es la URL que se utiliza para acceder a una API REST.

**Explicación**:

Un "endpoint" se refiere a un punto de acceso específico o una URL a través de la cual los clientes pueden interactuar con el servicio web para realizar una operación particular.

Cada endpoint en una API REST representa una funcionalidad específica o una acción que puede ser ejecutada sobre los recursos disponibles en el servidor. Estos endpoints son básicamente URLs que se mapean a funciones o métodos en el servidor que realizan operaciones como leer, crear, actualizar o eliminar datos.

# 87.- ¿Cuál es el valor de x e y al final el programa?

Pantalla de computadora con letras

Descripción generada automáticamente

1. X=9 y=10
2. X=10 y=9
3. X=10 y=10
4. X=9 y=9

**Respuesta**: Opción c) x=10 y=10

**Explicación**:

El ejercicio consiste en el uso de los loops do-while y while.

En el primer caso x entra valiendo 0, se imprime y después se post incrementa en uno, para entonces x ya vale 1, se ejecuta la condición si x es menor a 10 y como es verdad se regresa al loop, entonces cuando x llega a 10 x sale del loop y entra el segundo loop while con y declarada e inicializada en 0 comienza la condición ¿y es menor a 10?, como es verdad entonces entra al ciclo se imprime y al igual que x se post incrementa en uno, se evalúa la condición nuevamente y como aun es menor a diez continua la iteración. Finalmente sale del loop cuando y=10.

# 88.- ¿Dado el siguiente enum y clase cuál es la opción que puede ir en el espacio en blanco para que el código compile?

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

1. Ninguno de los anteriores
2. case SUMMER ->
3. case Season.Winter:
4. case FALL:
5. case Winter, Spring:
6. case SUMMER | WINTER:

**Respuesta**: Opción a) Ninguna de las anteriores

**Explicación**:

En Java, enum es una palabra clave que se utiliza para definir un tipo de dato especial que representa un conjunto fijo de constantes predefinidas. Un enum, abreviatura de enumeración, es una forma de definir un conjunto de valores que representan opciones o casos específicos.

En el código causa un error porque lejos de usar el enum si el default ya fue declarado no podemos colocar un case dentro de él y todas las opciones menos la a) implican un case.

# 89.- ¿Cuál es el resultado de compilar y ejecutar el siguiente programa’

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. False
2. no se produce salida
3. true
4. error de compilación

**Respuesta**: Opción a) False

**Explicación**:

La variable stmt1 nos da true porque está comparando dos cadenas iguales y dentro del pool de string estas apuntan al mismo objeto, entonces esto es true para stmt1

Para la variable stmt2 nos da false, recordemos que el stringBuilder construye su cadena afuera del pool de string entonces no apunta al mismo lugar y es false

Por ultimo la variable stmt3 tambien es false porque como stringBuilder contruye afuera del pool de string no va a ocupar las variables que estén afuera, mas bien construye una nueva entonces es false porque no apunte a la misma cadena

Entonces la evaluación queda (true && false || false) con que haya un false por el or nos da false

# 90.- ¿Cómo se manejan las excepciones en Java?

1. Las excepciones el manejan con bloques switch case en Java. La excepción trae with Resources es una forma de lanzar una excepción en un método.
2. Las excepciones se manejan con bloques while en Java. La excepción trae with Resources es una forma de manejar excepciones de compilación
3. las excepciones se manejan con bloques if else en Java. La excepción trae with resource es una forma de manejar las excepciones en tiempo de ejecución.
4. Las excepciones se manejan con bloques try catch finally en Java. La excepción trae with Resources es una forma de cerrar automáticamente los recursos abiertos en un bloque try.

**Respuesta:** opción d) Las excepciones se manejan con bloques try catch finally en Java. La excepción try with Resources es una forma de cerrar automáticamente los recursos abiertos en un bloque try.

**Explicación**:

En Java, las excepciones se manejan **mediante el uso de bloques try, catch, finally y la palabra clave throw**.

Try-with-resources y el multi-catch son características introducidas en versiones posteriores de Java que simplifican y mejoran la legibilidad del manejo de excepciones.

Try-with-resources**: Esta característica permite que los recursos que se abren en un bloque try se cierren automáticamente al final del bloque, sin necesidad de un bloque finally**

Multi-catch: Esta característica permite manejar múltiples excepciones en un solo bloque catch, lo que mejora la legibilidad del código.

# 91.- ¿Qué clase del paquete java.io permite leer y escribir archivos en ubicaciones específicas dentro de un archivo?

1. File
2. filename filter
3. file descriptor
4. RandomAccessFile

**Respuesta**: Opcion d) Random AccesFile

**Explicación**:

La clase del paquete java.io que permite leer y escribir archivos en ubicaciones específicas dentro de un archivo se llama **RandomAccessFile**. Esta clase proporciona métodos para realizar operaciones de lectura y escritura en cualquier posición dentro de un archivo, lo que permite acceder y manipular datos en ubicaciones específicas del archivo.

# 92.- Todas las siguientes definiciones de clases my School classroom y my City School ¿qué números de línea en el método main generan un error de compilación? (Elija todas las opciones correctas)

Texto, Carta

Descripción generada automáticamente

1. Ninguna, el código compila bien
2. línea 6
3. línea 9
4. línea 7
5. línea 8
6. línea 5

Texto

Descripción generada automáticamente

**Aquí dejo el código que muestra las líneas que tienen error.**

**Respuesta**: Opciones b) c) d) y f)

**Explicación**:

La línea 5 no compila porque globalKey no es publica por lo que no se puede accesar a ella, la línea 6 no compila por lo mismo, no se puede accesar a esa clase fuera de ese paquete, la línea 7 no compila porque rootNumber es privado y la línea 9 no compila porque teacherName es protected y la clase school no hereda de classroom

# 93.- ¿Qué es una expresión lambda en Java?

1. Una instancia de una clase que implementa una interfaz funcional
2. Una instancia de una clase abstracta que se utiliza para implementar métodos anónimos
3. Una forma concisa de representar una función anónima que se puede pasar como argumento 🡨 Es correcto
4. Un método que no tiene cuerpo

**Repuesta**: Opción C) Una forma concisa de representar una función anónima que se puede pasar como argumento

**Explicación**:

Una expresión lambda en Java es una forma de **crear funciones anónimas**, que **pueden ser tratadas como objetos y pasadas como argumentos de métodos**, almacenadas en variables, o devueltas como resultados de métodos

# 94.- ¿Qué hace el siguiente código fuente?

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

1. Muestra los números del 1 al 10
2. muestra un 10
3. se queda en un bucle infinito
4. muestra los números del 0 al 9

**Respuesta**: Opción c) Se queda en un bucle infinito

**Explicación**:

El código declara e inicializa dos variable una de tipo int llamada x con un valor de cero y una de tipo boolean llamada flag con valor false. Entonces entra a un ciclo While donde la comáracion es la siguiente

(x<10) || !flag

Esta sentencia nos dice que mientras x sea menor que diez o flag se diferente de false, osea true, el código va a imprimir a x, después realiza un Post-incremento a x donde evalúa la expresión con el valor actual de la variable, y luego incrementa la variable en 1. Entonces el código completa el primer loop y continua asi sin embargo no tiene un limite debido que la condición es que se cumpla x<10 o que flag sea diferente de false, pero **como no hay operador que modifique el valor de flag** entonces continuara en un ciclo infinito.

# 95.- ¿Qué son las anotaciones en java?

1. Una forma de declarar variables en Java.
2. Una forma de declarar métodos abstractos en Java.
3. Un mecanismo para etiquetar y procesar código de forma especial.
4. Comentarios especiales que se utilizan para documentar el código.

**Respuesta**: Opción C) Un mecanismo para etiquetar y procesar código de forma especial

**Explicación**:

Las anotaciones son una forma de añadir metadatos a las clases, métodos, variables y otros elementos del código. Estos metadatos pueden ser utilizados por herramientas y frameworks para realizar diversas tareas, como configuración, generación de código, validación y más.

**Nota**:

Las anotaciones en Java proporcionan una infraestructura básica para agregar metadatos al código, mientras que las anotaciones de Spring extienden esta capacidad para simplificar y automatizar la configuración y gestión de aplicaciones dentro del ecosistema Spring. **Las anotaciones de Spring se basan en el mecanismo de anotaciones de Java, pero están optimizadas para tareas específicas en el desarrollo de aplicaciones Spring**.

# 96.- ¿Qué son las expresiones regulares en Java?

1. Un mecanismo para validar y manipular fechas y horas.
2. Un mecanismo para validar y manipula cadenas de caracteres.
3. Un mecanismo para validar y manipular números enteros.
4. Un mecanismo para validar y manipular números decimales.

**Respuesta**: Opción b) Un mecanismo para validar y manipular cadenas de caracteres

**Explicación**:

Las expresiones regulares en Java son una poderosa herramienta utilizada para buscar, **coincidir** y **manipular patrones de texto**. Se basan en el paquete java.util.regex, que proporciona clases como Pattern y Matcher para trabajar con ellas. Las expresiones regulares permiten definir patrones complejos de búsqueda y sustitución en cadenas de texto, facilitando tareas como la validación de formatos, extracción de datos y transformación de texto.

**Ejemplo**:

String emailPattern = "^[\\w.-]+@[\\w.-]+\\.[a-zA-Z]{2,6}$";

# 97.- ¿Qué es una anotación en Spring?

1. Una biblioteca de terceros que se utiliza para extender la funcionalidad de Spring
2. Una clase que se utiliza para definir la estructura de una tabla de base de datos.
3. Una etiqueta que se utiliza para anotar una clase o un método y proporcionar información adicional al contenedor de Spring.
4. Un archivo de configuración XML que se utiliza para configurar una aplicación de Spring.

**Respuesta**: Opción C) Una etiqueta que se utiliza para anotar una clase o un método y proporcionar información adicional al contenedor de Spring

**Explicación**:

Antes de las anotaciones, el comportamiento de Spring Framework se controlaba en gran medida a través de la configuración XML. Hoy, el uso de anotaciones nos proporciona capacidades enormes en la forma en que configuramos los comportamientos de Spring Framework.

Una anotación es una forma de proporcionar **metadatos** sobre el código. Las anotaciones en Spring se utilizan para configurar y gestionar los componentes de la aplicación de manera declarativa, lo que reduce la necesidad de configuración explícita en archivos XML.

Utilizamos las anotaciones (a veces también llamadas "**etiquetas**") para **simplificar y centralizar la configuración de nuestras aplicaciones**. Las anotaciones permiten **declarar el comportamiento y la configuración de los componentes directamente en el código fuente, eliminando la necesidad de configuraciones externas extensivas** (como archivos XML).

**Ejemplos**:

@Required, @Autowired, @Qualifier, @Configuration, @ComponentScan, @Bean, @Lazy. @Value, @Repository, @Component, @RequestMapping(@GetMapping, @PostMapping, @PutMapping, @PatchMapping @DeleteMapping), @Controller, @CoockieValue

# 98.- ¿Cuál es la salida?

Texto

Descripción generada automáticamente

1. Ninguna de las anteriores
2. 1 3
3. 1 1
4. El código no compila en la línea x
5. 2 3

**Respuesta**: Opcion b) 1 3

**Explicación**:

* Se importa la clase ArrayList del paquete java.util.
* Se crea una nueva instancia de ArrayList de tipo String utilizando la palabra clave var para la inferencia de tipos.
* Se agregan tres elementos a la lista: "Austin", "Boston" y "San Francisco".
* Se crea un Stream a partir de la lista utilizando el método stream().
* Se utiliza el método filter() en el Stream para filtrar los elementos cuya longitud sea mayor que 10. La expresión lambda a -> a.length() > 10 es la condición de filtrado.
* Se cuenta la cantidad de elementos que cumplen con la condición de filtrado utilizando el método count().
* Se imprime la cantidad de elementos filtrados (c) y el tamaño total de la lista (list.size()), separados por un espacio.

Dado que solo hay un elemento en la lista con una longitud mayor que 10 ("San Francisco"), el valor de c será 1. Y el tamaño total de la lista es 3.

# 99.- ¿Cuál es la salida de la siguiente aplicación?

Texto

Descripción generada automáticamente

1. 5
2. 7
3. El código compila sin problemas la salida no se puede determinar hasta el tiempo de ejecución.
4. Ninguna de las anteriores.
5. El código no compila

**Respuesta:** Opcion e) el código no compila

**Explicación:** no compila debido a un conflicto de métodos default en las interfaces Speak y Sing. Ambas interfaces definen un método default con la misma firma talk(), pero con diferentes implementaciones.

Cuando una clase implementa múltiples interfaces que definen el mismo método default con la misma firma, se produce un **conflicto de métodos**, y Java no puede determinar qué implementación debe utilizar.

Para resolver este conflicto, la clase que implementa las interfaces debe proporcionar su propia implementación del método talk(), sobrescribiéndolo explícitamente. Sin embargo, en el código proporcionado, la clase Performance define un método talk(String... x) con una firma diferente, lo que no resuelve el conflicto. Para corregir el problema, debes proporcionar una implementación del método talk() sin argumentos en la clase Performance, resolviendo así el conflicto.

# 100.- ¿Qué conjunto de modificadores, cuando son agregados a un método default dentro de una interfaz, evitan que sea sobrescrito por la clase que lo implementa?

1. private
2. const
3. final
4. private static
5. Ninguna de las anteriores
6. Static

**Respuesta**: Opción c) final

**Explicación**:

En Java, cuando se define un método default dentro de una interfaz, se puede evitar que sea sobrescrito por la clase que implementa dicha interfaz mediante el uso del modificador **final**.

Es importante tener en cuenta que los métodos default se introdujeron en Java 8 y permiten proporcionar una implementación predeterminada en la interfaz. Sin embargo, las clases que implementan la interfaz aún pueden sobrescribir estos métodos, a menos que se marquen como **final**.

# 101.- ¿Qué tipo de excepción se produce cuando se intenta realizar una operación incompatible con el tipo de datos en Java?

1. ArrayIndexOutOfBoundsException
2. ArithmeticException
3. IllegalArgumntException
4. ClassCastException

**Respuesta:** Opcion d) ClassCastException

**Explicación:**

En Java, cuando se intenta realizar una **operación incompatible con el tipo de datos**, se produce una excepción de tipo java.lang.ClassCastException.

Por ejemplo, si se intenta hacer un casting de un objeto de una clase a otra clase incompatible, se lanzará una ClassCastException.

En general, las excepciones de tipo ClassCastException y ArrayIndexOutOfBoundsException se lanzan cuando se intenta realizar operaciones que no son válidas para los tipos de datos involucrados.

# 102.- ¿Qué es un starter?

1. Es una herramienta que nos facilita la creación y configurar un proyecto cargando las dependencias necesarias para un objetivo.
2. Es el inicializador de un proyecto en Java.
3. Componente encargado de realizar las operaciones de balanceador.

**Respuesta:** Opción a) Es una herramienta que nos facilita la creación y configurar un proyecto cargando las dependencias necesarias para un objetivo.

**Explicación:**

Un starter es una herramienta de software **que simplifica la creación y configuración de un proyecto**, especialmente al **inicializarlo con las dependencias y configuraciones básicas** necesarias para cumplir con un objetivo específico.

Los starters funcionan como plantillas o kits predefinidos que incluyen los elementos esenciales para comenzar a desarrollar un proyecto sin tener que configurar todo desde cero.exclamation Esto ahorra tiempo y esfuerzo, especialmente para proyectos que siguen patrones o arquitecturas comunes.

**Ejemplos de Starters**:

1. Spring Initializr
2. Maven Archetype
3. G radle Init
4. JHipster
5. Quarkus Maven Plugin
6. Micronaut
7. Dropwizard

# 103.- Selecciona la respuesta correcta con respecto al resultado del siguiente bloque de código.

Texto

Descripción generada automáticamente

**Respuesta:** No compila

**Explicación:**

* No es válido usar **protected** dentro de un método para declarar una variable local. Los modificadores de acceso (protected, public, private) solo pueden aplicarse a miembros de clase (atributos y métodos), no a variables locales dentro de métodos.
* La clase Test2 contiene **dos métodos main**, lo cual no es válido en Java. Solo puede haber un método main en una clase.

**Recomendación:** El código puede ser confuso dado que intenta manejar hilos y este es un tema más avanzado para el nivel de este examen, por lo tanto, hay que enfocarse en la parte básica de Java

# 104.- Comando Docker que se utiliza para construir la imagen a partir del Dockerfile

1. docker run
2. docker build
3. docker create
4. docker image

**Respuesta:** Opción b) docker build

**Explicación:** El comando **docker build** se utiliza para construir una imagen Docker a partir de un Dockerfile.

# 105.- ¿Seleccione la respuesta que considere correcta, dado el siguiente bloque de código?

1. [ [1], [2], [3], [4], [5], [7], [8] ]
2. [ [1, 2 ]], [ [ 3, 4]], [[ 5, 6]], [ [ 7, 8]]
3. [ [ 1, 2], [ 3,4], [ 5,6], [ 7,8]]

**Respuesta**: Opcion C) [ [ 1, 2], [ 3,4], [ 5,6], [ 7,8]]

**Explicación**:

El código es un ejemplo de cómo se puede aplanar una matriz tridimensional en una matriz bidimensional

## Notas

Existe una incongruencia en esta pregunta, debido a que el código está mal escrito porque los métodos. flapMapToIn no existe en java y la clase Array no tiene como método el método String.

Si este es el caso entonces el código no podría compilarse y correr y la opción sería la opción D) no compila.

Sin embargo, la opción D no es información que nos proporcionaron, solo es una suposición, sin la opción D entonces la respuesta correcta seria la C ya que no nos da la opción de que falle el programa.