Presentación: Temario ASO

Introducción

- Importancia del conocimiento integral en el desarrollo de software moderno.
- Objetivo: Repasar los conceptos clave y tecnologías vistas.

I. Lenguaje de Programación Java

A. Conceptos Fundamentales

- Máquina Virtual de Java (JVM)
- ¿Qué es un “code smell”? - Introducción a la calidad del código.

B. Manejo de Excepciones

- Tipos: RuntimeException vs. Checked Exception.
- Mecanismos: try-catch-finally.
- Creación: throw y excepciones personalizadas.
- Información: getMessage().

C. Sintaxis y Semántica

- Programación funcional en Java (Introducción a Lambdas y Streams).
- Referencias en Java

D. Programación Orientada a Objetos (POO)

- Pilares: Polimorfismo, Abstracción, Herencia, Encapsulamiento.
- Polimorfismo en acción.
- Compilación (javac) y ejecución (java).
- Operador :: (Referencia a métodos) en Java 8.
- Subclases concretas y declaraciones import.

E. Clases Abstractas e Interfaces

- Diferencias clave y cuándo usar cada una.
- Métodos static y default en interfaces (Java 8+).

F. Características Modernas de Java (Java 8+)

- Expresiones Lambda: Sintaxis y uso.
- API de Streams: Procesamiento de colecciones.
- Operadores de cortocircuito (&&, ||).
- La palabra clave final.
- Generics: Tipado seguro.

G. API de Colecciones

- Interfaces principales: List, Map, Set, Queue.
- Comparativa: ArrayList vs. LinkedList.

H. Declaraciones

- Asignación, Assertion.

I. Otros Conceptos Clave

- Archivos JAR: Empaquetado de aplicaciones.
- Paquetes GUI: javax.swing, java.awt (Mención breve).
- Entrada/Salida: Lectura y escritura de archivos (java.io).
- Excepciones comunes de la JVM: ArrayIndexOutOfBounds, NumberFormat, NullPointer, IOException, ExceptionInInitializerError.
- Utilidad javaw en Windows.

II. Spring Framework

- ¿Qué es Spring? Ecosistema y módulos principales.
- **Beans de Spring:**
  - Concepto de Bean y Contenedor IoC.
  - Anotaciones: @RestController, @Component, @Service, @Repository.
- **Inyección de Dependencias (DI):** Concepto y beneficios.
- **APIs REST con Spring:** Uso de @RestController.

III. APIs REST

- **Conceptos Fundamentales:**
  - ¿Qué es REST? Principios.
  - CRUD (Create, Read, Update, Delete) y su mapeo a verbos HTTP.
  - Endpoint: Definición y ejemplos.
  - Diferencia entre PUT y PATCH.
  - Formato de datos: JSON.
- **Ventajas:** Comparación con otros enfoques (SOAP).

IV. Patrones de Diseño de Software

- **Importancia:** Reusabilidad, mantenibilidad, comunicación.
- **Patrones Estructurales:** Adapter, Proxy, Bridge, Composite (Breve descripción y caso de uso).
- **Patrones de Creación:** Builder, Singleton, Prototype, Abstract Factory (Breve descripción y caso de uso).
  - Implementación del Singleton en Java 8.
- **Patrones Específicos:**

    - DAO (Data Access Object): Implementación en Java.
- **Patrones de Microservicios:** Circuit Breaker, Retry, Bulkhead (Introducción).

V. Herramientas de Construcción (Build Tools)

- **Maven:**
    - Propósito: Gestión de dependencias y ciclo de vida del build.
    - pom.xml: Estructura y elementos clave (dependencies, build, profiles).
    - Ciclo de vida y Objetivos (Goals): clean, package, install, debug.
    - Directorio target.
    - Archivos de configuración.

VI. Sistemas de Control de Versiones

- **Git:**
    - ¿Qué es Git? Control de versiones distribuido.
    - Conceptos: Repositorio local vs. remoto, commit, branch, merge.
    - **Comandos Esenciales:**
        - clone, add, commit, push, pull.
        - status, log, diff.
        - branch, checkout, merge.
        - reset, revert, amend (para deshacer/modificar).
    - **Buenas Prácticas:**
        - Formato de mensajes de commit.
        - Flujo de trabajo básico (branching, merging).
        - Actualizar rama local (git pull).

VII. Pruebas de Software

- **Importancia:** Calidad, detección temprana de errores, refactorización segura.
- **Pruebas Unitarias:**
    - Propósito: Probar unidades aisladas de código.
    - Buenas prácticas.
    - Anotación @Ignore (o similar según el framework, ej. @Disabled en JUnit 5).
- **Pruebas de Integración:** Probar la interacción entre componentes.
- **Cobertura de Código:** Métrica para evaluar qué porcentaje del código está cubierto por pruebas.

VIII. Microservicios

- **¿Qué son?** Arquitectura de software basada en servicios pequeños e independientes.
- Ventajas y Desafíos.

- **Patrones de Diseño:** Circuit Breaker, Retry, Bulkhead (Relación con la resiliencia).

IX. Principios de Diseño

- **Principio de Sustitución de Liskov (LSP):**
  - Parte de los principios SOLID.
  - Definición y ejemplo.
  - Importancia para la herencia y el polimorfismo correctos.