Nuevos Ejercicios

Interfaces

1. **Interfaz con métodos estáticos**
   Define una interfaz Utilidades con un método estático esPar(int numero) que devuelva un booleano. Implementa esta interfaz en una clase NumeroUtil que también tenga un método de instancia sumarPares(int limite).

2. **Herencia de interfaces**
   Crea una interfaz Animal con un método moverse(). Luego, crea una interfaz Mamifero que extienda Animal y añada el método amamantar(). Implementa ambas en una clase Perro.

3. **Interfaz con múltiples métodos**
   Define una interfaz DispositivoElectronico con los métodos encender(), apagar() y reiniciar(). Crea una clase Telefono que la implemente con mensajes personalizados.

Clases Abstractas

4. **Clase abstracta con método protegido**
   Crea una clase abstracta VehiculoMotorizado con un método protegido arrancarMotor() y un método abstracto acelerar(). Implementa una subclase Moto que use el método protegido.

5. **Clase abstracta con atributos finales**
   Define una clase abstracta Producto con un atributo final id y un método abstracto calcularPrecio(). Crea subclases Libro y Electronico con precios calculados de manera distinta.

6. **Herencia múltiple con clases abstractas**
   Crea una clase abstracta Artista con un método abstracto crearObra(). Luego, define una clase abstracta Musico que extienda Artista y añada tocarInstrumento(). Implementa una clase Pianista.

Expresiones Lambda

7. **Lambda con BiFunction**
   Usa BiFunction<Integer, Integer, Integer> para crear una expresión lambda que devuelva el mayor de dos números enteros.

8. **Lambda con UnaryOperator**
   Define un UnaryOperator<String> que convierta una cadena a mayúsculas y aplícalo a una lista de nombres.

9. **Lambda con BiPredicate**
   Crea un BiPredicate<String, String> que verifique si dos cadenas tienen la misma longitud. Pruébalo con ejemplos.

10. **Lambda con Supplier de objetos**
    Usa Supplier<Random> para generar un objeto Random y obtener un número aleatorio entre 1 y 100.

Streams

11. **Filtrado de números divisibles por 3**
    Dada una lista de enteros, usa un Stream para filtrar y mostrar solo los números divisibles por 3.

12. **Promedio de números mayores a un umbral**
    Dada una lista de enteros y un valor X, usa Stream para calcular el promedio de los números mayores que X.

13. **Transformación a potencias de 2**
    Dada una lista de enteros, usa map para transformar cada número en su potencia de 2 (2^n) y recoge los resultados en una lista.

14. **Contar cadenas vacías**
    Dada una lista de cadenas, usa Stream para contar cuántas están vacías ("").

15. **Ordenación inversa de palabras por longitud**
    Dada una lista de cadenas, usa Stream para ordenarlas de mayor a menor longitud.

16. **Suma de dígitos de números**
    Dada una lista de enteros, usa Stream para calcular la suma de todos sus dígitos individuales (ej. 123 -> 1+2+3).

17. **Agrupar números por paridad**
    Dada una lista de enteros, usa Collectors.groupingBy para agruparlos en un Map<String, List<Integer>> con claves "par" e "impar".

18. **Encontrar el número más cercano a cero**
    Dada una lista de enteros, usa Stream y min con un comparador personalizado para encontrar el número más cercano a cero.

19. **Eliminar vocales de cadenas**
    Dada una lista de cadenas, usa Stream y map para eliminar todas las vocales de cada cadena.

20. **Lista de factoriales**
    Dada una lista de enteros, usa Stream para transformar cada número en su factorial y recoger los resultados en una lista.

Referencias a Métodos

21. **Referencia a método de instancia**
    Crea una clase Texto con un método de instancia contarLetras(). Usa una referencia a método con Function<String, Integer> para contar las letras de una cadena.

22. **Referencia a método con Stream**
    Dada una lista de enteros, usa map con una referencia a Math::abs para obtener una nueva lista con valores absolutos.

23. **Referencia a constructor con parámetros múltiples**
    Crea una clase Estudiante con un constructor que reciba nombre y edad. Usa BiFunction<String, Integer, Estudiante> con Estudiante::new.

Más Conceptos Avanzados

24. **Interfaz funcional personalizada con genéricos**
    Define una interfaz funcional Transformador<T, R> con un método transformar(T input). Usa una lambda para convertir una lista de enteros en cadenas.

25. **Clase abstracta con método estático**
    Crea una clase abstracta Herramienta con un método estático crearHerramienta(String tipo) y un método abstracto usar(). Implementa una subclase Martillo.

26. **Stream con flatMap**
    Dada una lista de listas de enteros (List<List<Integer>>), usa flatMap para aplanarla en una sola lista y sumar todos los elementos.

27. **Partición de números positivos y negativos**
    Dada una lista de enteros, usa Collectors.partitioningBy para separarlos en un Map<Boolean, List<Integer>> según si son positivos o no.

28. **Máximo y mínimo en un solo Stream**
    Dada una lista de enteros, usa Stream para encontrar el máximo y el mínimo en una sola pasada (pista: usa reduce o summaryStatistics).

29. **Simulación de un dado con Supplier**
    Crea un Supplier<Integer> que simule tirar un dado (valores entre 1 y 6) y úsalo para generar una lista de 10 tiradas.

30. **Combinación de Streams**
    Dadas dos listas de cadenas, usa Stream para concatenarlas, eliminar duplicados y mostrarlas en orden alfabético inverso.