Ejercicios de la UT7 (parte 1)

1) Implementación básica de una interfaz

Crea una interfaz Reproducible con un método reproducir(). Luego, crea las clases Cancion y Pelicula que implementen esta interfaz e impriman mensajes distintos.

2) Interfaz con atributos constantes

Define una interfaz OperacionesMatematicas con una constante PI y métodos sumar(int, int), restar(int, int). Implementa la interfaz en una clase Calculadora.

3) Interfaces múltiples

Crea dos interfaces:

Comestible con el método comer().
Bebible con el método beber().
Implementa ambas en una clase Gazpacho, que imprima mensajes diferentes al llamar cada método.

4) Interfaz con métodos por defecto

Define una interfaz Vehiculo con el método mover(), y un método por defecto detener(). Crea la clase Bicicleta que implemente Vehiculo.

5) Interfaz funcional con expresión lambda

Crea una interfaz funcional Operacion con un método calcular(int a, int b). Implementa una lambda que realice la multiplicación de dos números.

6) Clase abstracta y herencia

Crea una clase abstracta Cocinero con un método cocinar() que siga estos pasos:

Llamar a un método abstracto prepararIngredientes().
Llamar a un método abstracto cocinarPlato().
Imprimir "Plato listo para servir" al final.

Luego, crea dos subclases:

CocineroItaliano, que imprime "Preparando pasta" y "Cocinando a fuego lento".
CocineroJapones, que imprime "Cortando pescado" y "Preparando sushi".
En main(), crea instancias de ambos cocineros y llama a cocinar().

7) Métodos abstractos y concretos

Define una clase abstracta Figura con:

Un método abstracto calcularArea().
Un método concreto mostrarInfo() que imprima "Soy una figura".
Crea las clases Triangulo y Circulo que extiendan Figura (y sobrescriban el métodoMostrarInfo).

8) Clase abstracta con constructor

Crea una clase abstracta Empleado con un constructor que reciba nombre y salarioBase (sus atributos) y un método abstracto calcularSalario(). Implementa una subclase Gerente que agregue un atributo bono.

9) Jerarquía de clases abstractas

Crea una clase abstracta InstrumentoMusical con un método tocar(). Luego, define Guitarra y Piano que extiendan de esta clase.

10) Clases abstractas con atributos protegidos

Define una clase abstracta Electrodomestico con atributos protegidos marca y consumo.
Crea una subclase Televisor que implemente un método mostrarInfo().

11) Lambda con Predicate

Usa Predicate<String> para filtrar una lista de nombres y mostrar solo los que empiezan con "A".

12) Lambda con Function

Utiliza Function<Integer, String> para convertir una lista de números en sus representaciones en texto (ej. 1 -> "Uno").

13) Lambda con Consumer

Usa Consumer<Integer> para imprimir el doble de cada número en una lista.

14) Lambda con Supplier

Crea un Supplier<String> que devuelva un mensaje aleatorio entre "Hola", "Bienvenido", "Saludos".

15) Lambda con Comparator (ordenación básica)

Dada una lista de números, usa Comparator<Integer> para ordenarlos de mayor a menor con una expresión lambda.

16) Lambda con Comparator (ordenación de objetos)

Crea una clase Persona con nombre y edad. Ordena una lista de personas de menor a mayor edad con Comparator<Persona>.

17) Lambda con una interfaz funcional personalizada

Define una interfaz funcional Operacion con un método ejecutar(int, int). Usa una lambda para implementarla con la suma y la resta.

18) Referencias a métodos con lambda

Dada una lista de cadenas, usa List.forEach() con una referencia a método System.out::println para imprimir cada una.

19) Referencia a método estático

Crea una clase Calculadora con un método static int cuadrado(int x). Usa Function<Integer, Integer> para referenciarlo con Calculadora::cuadrado.

20) Referencia a un constructor con lambda

Crea una clase Libro con un constructor que reciba String titulo. Usa Supplier<Libro> para instanciar un objeto con Libro::new.

21) Filtrado de números primos

Dada una lista de números enteros, usa un Stream para filtrar y mostrar solo los primos. La verificación de si un número es primo debe realizarse con una expresión lambda.

22) Suma de cuadrados de los primeros N números pares

Dado un número N, genera los primeros N números pares y usa map para elevarlos al cuadrado. Finalmente, usa reduce para sumarlos.

23) Recuento de palabras en una lista

Dada una lista de cadenas de texto, usa Stream para contar cuántas palabras hay en total (sumando las palabras de cada cadena). Usa split("\\s+") para dividir palabras.

24) Nombres únicos y en orden alfabético

Dada una lista de nombres, usa Stream para eliminar duplicados y mostrar los nombres ordenados alfabéticamente.

25) Encontrar la palabra más larga

Dada una lista de cadenas, usa Stream y reduce para encontrar la palabra con más caracteres.

26) Crear una lista de cuadrados de números impares

Dada una lista de enteros, usa Stream para obtener los números impares y crear una nueva lista con sus cuadrados.

27) Concatenación de cadenas con separador

Dada una lista de nombres, usa Stream y Collectors.joining(", ") para concatenarlos en una sola cadena separada por comas.

28) Contar elementos mayores que un valor dado

Dada una lista de números y un número X, usa Stream y filter para contar cuántos elementos son mayores que X.

29) Transformar objetos con map

Dada una lista de objetos Persona(nombre, edad), usa map para transformar la lista en una lista de nombres (List<String>).

30) Agrupar palabras por su primera letra

Dada una lista de palabras, usa Stream y Collectors.groupingBy para agruparlas en un Map<Character, List<String>>, donde la clave es la primera letra de cada palabra.