EJERCICIOS FORMATO EXAMEN POO

El objetivo de estos ejercicios es que practiquéis en lo difícil para que se os haga fácil el examen. No os preocupéis si no os salen, es simplemente que veáis cosas más complejas.

- **Ejercicio 1 -POO**
  Crear un programa con la siguiente estructura que permite controlar el funcionamiento de una centralita de teléfonos Crear una clase LlamadaLocal que tenga las siguientes características:
  1. **Atributos privados**: numeroOrigen (long), numeroDestino (long), coste (double), duracion (double).
  2. **Constructor con todos los parámetros**. Se calculará automáticamente el coste teniendo en cuenta que el coste de cada segundo es de 0,15 más un establecimiento de llamada de 0,50
  3. **Métodos de acceso y modificación**: Para cada uno de los atributos (getters y setters).
  4. **Método void mostrarDatos().**

Crea una clase LlamadaNacional que extienda de LlamadaLocal. Tendrá que cumplir las siguientes características

1. **Atributo privado:** destino (char), será A,B,C
2. **Constructor con todos los parámetros**. Se calculará automáticamente el coste teniendo en cuenta que: el coste de cada segundo es de 0,40 si el destino es A,el coste de cada segundo es de 0,50 si el destino es B, el coste de cada segundo es de 0,60 si el destino es C o más un establecimiento de llamada de 0,70.
3. **Métodos de acceso y modificación**: Para cada uno de los atributos (getters y setters).
4. **Implementación del método void calcularCoste().** Con las instrucciones que he indicado en el constructor. Al final del método llamada al método setCoste() para poder modificar el coste.

Crea una clase Centralita. Tendrá que cumplir las siguientes características.

1. **Atributo privado: costeAcumulado (int)**
2. **Constructor por defecto. costeAcumulado = 0.**
3. **Método void agregarLocal(LlamadaLocal nombreatributo):** Tendrás que sumar al coste acumulado al de la llamada que se introduce por parámetros.
4. **Método void agregarNacional(LlamadaNacional nombreatributo):** Tendrás que sumar al coste acumulado al de la llamada que se introduce por parámetros.
5. **Método getCosteAcumulado()**

Clase Main: Crea un objeto Centralita. Crea dos objetos de LlamadaLocal, y llamada al método agregarLocal para los dos objetos. Crea dos objetos de LlamadaNacional, y llamada al método agregarNacional para los dos objetos.

Llama a los métodos mostrar datos para todos los objetos y al método getCosteAcumulado para ver el coste acumulado en la centralista.

- **Ejercicio 2 -POO**
  Crear un programa con la siguiente estructura que permite controlar el funcionamiento de una tienda de discos. Crear una clase Canción que tenga las siguientes características:

  1. **Atributos privados**: título, artista. Los dos String.
  2. **Constructor con todos los parámetros**.
  3. **Métodos de acceso y modificación**: Para cada uno de los atributos (getters y setters).
  4. **Método toString().**

Crea una clase CD. Tendrá que cumplir las siguientes características.

1. **Atributo privado:** Array de objetos de la clase Canción. **Array, no arraylist**. contador (int)
2. **Constructor por defecto.** Asume un tamaño fijo del array de 10 posiciones y contador inicializado en 0.
3. **Métodos int númeroCanciones().**
4. **Método Cancion dameCancion(int posicion):** devuelve una canción a través de una posición. Si no lo encuentra devolverá null
5. **Método void grabaCancion(int posición, Cancion cancion):** Método para introducir una canción en el array Canción. Mucho cuidado que no puede haber espacios en blanco.
6. **Método void agrega(Cancion cancion):** Muy parecido al método anterior pero sin pedir la posición donde se quiere insertar, se pondrá en la primera posición libre.
7. **Método void elimina(int posición):** Elimina la canción para ello guarda en la posición el valor de null.

Crea una clase main: Con un menú de opciones. Probando cada uno de los métodos de la clase CD.

- **Ejercicio: Sistema de Gestión de Biblioteca Personal (Nivel medio – difícil)**

Contexto:

Con el objetivo de organizar mejor tu colección de libros y facilitar el acceso a la información sobre ellos, decides crear un programa en Java que te permita gestionar tu biblioteca personal mediante ficheros de texto. Este sistema permitirá añadir nuevos libros, buscarlos por título o autor, y eliminarlos del registro. Cada libro se caracteriza por su título, autor y año de publicación.

Especificaciones:

Requisitos Funcionales:

1. **Agregar Libro:**
    - El sistema debe permitir añadir un nuevo libro al registro, solicitando al usuario el título, el autor y el año de publicación.
    - Cada libro debe almacenarse en un fichero de texto denominado "biblioteca.txt", donde cada línea representará un libro con sus datos separados por comas, por ejemplo: "El Señor de los Anillos, J.R.R. Tolkien, 1954".
2. **Buscar Libro:**
    - Implementa una función que permita buscar libros por título y por autor.
    - La búsqueda debe ser insensible a mayúsculas y minúsculas.
    - Si se encuentran coincidencias, el sistema deberá mostrar toda la información del o los libros encontrados.
3. **Eliminar Libro:**
    - El sistema debe ofrecer la opción de eliminar un libro del registro.
    - Para ello, el usuario introducirá el título del libro que desea eliminar.
    - Si existen varios libros con el mismo título, se solicitará al usuario que especifique el autor.
    - Confirma la eliminación antes de proceder.
4. **Listar Libros:**
    - Debe existir una opción para listar todos los libros registrados en el sistema, mostrando su título, autor y año de publicación.

Requisitos Técnicos:

- Utiliza manejo de excepciones para controlar posibles errores durante la lectura y escritura de ficheros, así como la entrada de datos por parte del usuario.
- Implementa una estructura de menú que permita al usuario seleccionar las diferentes operaciones disponibles (Agregar, Buscar, Eliminar, Listar).

- Considera la creación de una clase **Libro** que modele la información necesaria (título, autor, año de publicación) y una clase **GestorBiblioteca** que contenga la lógica para interactuar con el fichero de texto y realizar las operaciones requeridas.

Este ejercicio te permitirá practicar la manipulación de ficheros de texto en Java, el manejo de excepciones y la implementación de un sistema CRUD (Crear, Leer, Actualizar, Eliminar) sencillo.

- **Usando Arrays y arraysmultidimensionales. Ejercicio arrays nivel difícil.**

Crear un programa que lea los precios de 5 artículos y las cantidades vendidas por una empresa en sus 4 sucursales. Informar:

- Las cantidades totales de cada articulo.
- La cantidad de artículos en la sucursal 2.
- La cantidad del articulo 3 en la sucursal 1.
- La recaudación total de cada sucursal.
- La recaudación total de la empresa.
- La sucursal de mayor recaudación.