Segunda evaluación: Programación Orientada a Objetos.

**Para obtener SA, deberá implementar las opciones del 1-7 y 11. Deberá tener también todas las prácticas.**

**Para obtener DE, deberá implementar la opcion 8**

**Para obtener AU, deberá implementar las opciones 9 y 10**

Realice una clase llamada Principal, deberá tener un menú con las opciones siguientes, debe considerar que luego de elegir las opciones de 1-10 deberá regresar a éste menú principal:

1- Calficaciones

2- Serie

3- Perfectos

4- Viñedo

5- Estadio

6- Viñedo con objetos

7- Estadio con objetos.

8- Centro de Servicio Automotriz

9- Relación composición.

10- Documentación

11- Salir

Deberá permitir leer desde teclado la opción elegida, y realizará:

Opción 1:

Realice un programa utilizando el paradigma estructurado donde permita leer por teclado las notas de N alumnos que están matriculados en M asignaturas, si un alumno no está matriculado de una asignatura o no se presenta al examen se debe introducir como nota -1. El programa debe calcular y escribir el número de alumnos presentados a cada asignatura, la nota media, el número de no acreditados, el número de aprobados, el número de destacados y el número de autónomos.

Opción 2:

Realice un programa utilizando el paradigma estructurado que permita leer por teclado un número el cual será el tope de una serie con las siguientes características:

1/2+1/5+1/7+1/10+1/12 …. Hasta llegar al número leído por teclado.

Opción 3:

Realice un programa utilizando el paradigma estructurado que imprima los números perfectos menores o iguales a N. Un número es perfecto si “la suma de sus divisores excepto el mismo es igual al número propio”.

Opción 4:

Realice un programa utilizando el paradigma estructurado. En una bodega se tiene información sobre las cantidades producidas de cada tipo de vino, a lo largo de los últimos años. Calcule e imprima lo siguiente:

1. El total producido de cada tipo de vino (son 5 tipos) a lo largo de los N años.
2. El total producido de vino por año.
3. Año en que se produjo la mayor cantidad de litros de vino del tipo 2. Imprimir también la cantidad de litros.
4. Verificar si hubo algún año en el cual no se produjo el vino tipo 3. Si existe dicho año imprimirlo.

**Opción 5:**

Realice un programa utilizando el paradigma orientado a objetos. En un estadio se tienen 5 tipos diferentes de localidades, las cuales se identifican por una clave numérica que es un valor comprendido entre 1 y 5. Los precios de cada localidad y los datos referentes a las ventas de boletos para el próximo juego se proporciona como sigue:

1. Lea los precios.
2. Lea los datos de las ventas de boletos.
3. Imprima de cada venta, la clave, cantidad y el importe total de los boletos vendidos en esta venta.
4. Calcule e imprima la cantidad de boletos vendidos de cada tipo.
5. Calcule e imprima la recaudación total del estadio.

NOTA: Considere que en una venta, solo se venden boletos de un solo tipo.

**Opción 6:**

Deberán impletar la opción 4 utilizando el paradigma orientado a objetos.

**Opción 7:**

Un cliente quiere que construyamos un programa utilizando el paradigma orientado a objetos para manejar las reservas de un vuelo. Se sabe que el avión tiene 50 sillas, de las cuales 8 son de [clase](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#clase) ejecutiva y las demás de [clase](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#clase) económica. Las sillas ejecutivas se acomodan en filas de cuatro, separadas en el medio por el corredor. Las sillas económicas se acomodan en filas de seis, tres a cada lado del corredor.

Cuando un pasajero llega a solicitar una silla, indica sus datos personales y sus preferencias con respecto a la posición de la silla en el avión. Los datos del pasajero que le interesan a la aerolínea son el nombre y la cédula. Para dar la ubicación deseada, el pasajero indica la [clase](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#clase) y la ubicación de la silla. Esta puede ser, en el caso de las ejecutivas, [ventana](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#ventana) y pasillo, y en el de las económicas, [ventana](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#ventana), pasillo y centro. La [asignación](file:///C:\Users\anaestrada\Documents\PROGRAMACIÓN%20OO\GLOSSARY.html#asignación) de la silla en el avión se hace en orden de llegada, tomando en cuenta las preferencias anteriores y las disponibilidades.

Deberá permitir realizar lo siguiente:

* Asignar una silla a un pasajero según sus preferencias. Estas son [clase](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#clase) (Ejecutiva o Económica) y ubicación ([Ventana](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#ventana), Centro o Pasillo). Se marca como asignada una de las sillas disponibles en el avión, dependiendo de la [clase](https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/GLOSSARY.html#clase) y ubicación elegida. En caso de que todas las sillas estén asignadas, se muestra un mensaje de error.

Opción 8:

Realice lo siguiente utilizando el paradigma orientado a objetos.

Se debe escribir una o varias clases para crear un Centro de Servicio Automotriz que pueda ingresar vehículos a servicio. Los vehículos pueden ser coches o motos. Cada vehículo debe tener:

* Atributos para guardar matrícula (placa), marca y kilometraje. Además, cada vehículo debe tener un atributo booleano para saber si cuenta con verificación de contaminantes. Los coches tendrán número de pasajeros y las motos, cilindrada.
* Métodos getters y setters para cada atributo, excepto setter para marca y getter para Verificación.
* Constructor que reciba como parámetros: matrícula, marca y kilometraje. Los coches deben tener en su constructor el número de pasajeros y las motos, la cilindrada.
* Método **definirServicio**() que no reciba parámetros y que devuelva un String con lo siguiente (el método no debe imprimir NADA en consola):
  + “CAMBIO DE ACEITE” si tiene un kilometraje mayor o igual a 10 mil y menor a 20 mil kilómetros.
  + “CAMBIO DE FILTROS” si tiene un kilometraje entre 20 y 50 mil kilómetros.
  + “CAMBIO DE LLANTAS” si tiene un kilometraje mayor a 50 mil kilómetros.
  + Si tiene menos de 10 mil kilómetros regresará una cadena vacía.
* Método **estaVerificado()** que devuelva un valor booleano si el vehículo tiene verdadero su atributo de verificación.

Se debe crear una clase “CentroServicio” que contenga un arreglo de vehículos para almacenarlos. Debe contener:

* Constructor que reciba un parámetro “int” que defina el tamaño del arreglo de vehículos.
* Un método **ingresar()** que reciba un vehículo y lo asigne al arreglo. En caso de no existir espacio disponible debe devolver un valor “false” y mostrar un mensaje en consola. Si logra ingresarlo exitosamente, el método debe devolver un “true”.
* Un método **retirar()** que reciba como parámetro un vehículo, verifique que exista en el arreglo y lo elimine lógicamente. Debe regresar un “true” si el borrado fue exitoso.
* Un método **imprimir()** que imprima en consola los datos de los vehículos ingresados, el servicio que requiere cada uno y número de vehículos que ya estén verificados.

Se debe crear una clase “Principal” que contenga el método **main()** de la aplicación, que cree un Centro de Servicio, un coche y una moto, que ingrese ambos vehículos al Centro de Servicio y que imprima el reporte de vehículos. En éste caso, la clase principal es la que contiene el menú. Debe implementar herencia.

Opción 9:

Implemente un ejemplo usando la opción 7 sobre una relación de composición.

Opción 10:

En ésta opción se hará una revisón del código para verificar que se encuentre documentado.

Opción 11:

Finaliza el programa.