Ejercicios

Programación orientada a objetos

Programación orientada a objetos

Antes de empezar

- Deberá asignarle a la propiedad 'Title' de la clase Console, el número de ejercicio.

  Por ejemplo:
  Console.Title = “Ejercicio Nro ##” donde ## será el número del ejercicio.

- Del mismo modo se deberán nombrar los proyectos de consola.

  Por ejemplo:
  Ejercicio_##.

- Para visualizar los valores decimales de los ejercicios, deberá dar el siguiente formato de salida al método Write/WriteLine: “#,###.00”.

I01. Creo que necesito un préstamo

Crear una aplicación de consola y una biblioteca de clases que contenga la clase `Cuenta`.

Deberá tener los atributos:

- `titular` que contendrá la razón social del titular de la cuenta.
- `cantidad` que será un número decimal que representa al monto actual de dinero en la cuenta.

Construir los siguientes métodos para la clase:

- Un constructor que permita inicializar todos los atributos.
- Un método para obtener cada atributo.
- `mostrar` retornará una cadena de texto con todos los datos de la cuenta.
- `ingresar` recibirá un monto para acreditar a la cuenta. Si el monto ingresado es negativo, no se hará nada.
- `retirar` recibirá un monto para debitar de la cuenta. La cuenta puede quedar en negativo.

En el método `Main`, simular depósitos y extracciones de dinero de la cuenta, e ir mostrando como va variando el saldo.

I02. ¿Vos cuántas primaveras tenés?

Crear una aplicación de consola y una biblioteca de clases que contenga la clase `Persona`.

Deberá tener los atributos:

- `nombre`
- `fechaDeNacimiento`
- `dni`

Deberá tener un constructor que inicialice todos los atributos.

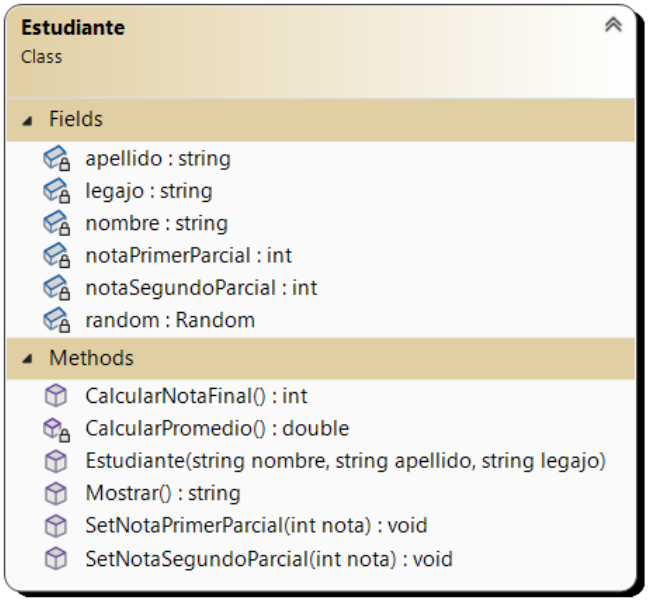
Construir los siguientes métodos para la clase:

- `CalcularEdad` será privado y retornará la edad de la persona calculándola a partir de la fecha de nacimiento.
- `Mostrar` retornará una cadena de texto con todos los datos de la persona, incluyendo la edad actual.
- `EsMayorDeEdad` si es mayor de edad devuelve el valor “Es mayor de edad", sino devuelve “es menor”.

Instanciar 3 objetos de tipo `Persona` en el método `Main`, mostrar quiénes son mayores de edad y quiénes no.

I03. El ejemplo universal

Crear una aplicación de consola y una biblioteca de clases que contenga la clase del siguiente diagrama:



La clase `Estudiante`:

- Tendrá un constructor de instancia que inicializará los atributos nombre, apellido, legajo y random.
- El método `SetNotaPrimerParcial` permitirá cambiar el valor del atributo `notaPrimerParcial`.

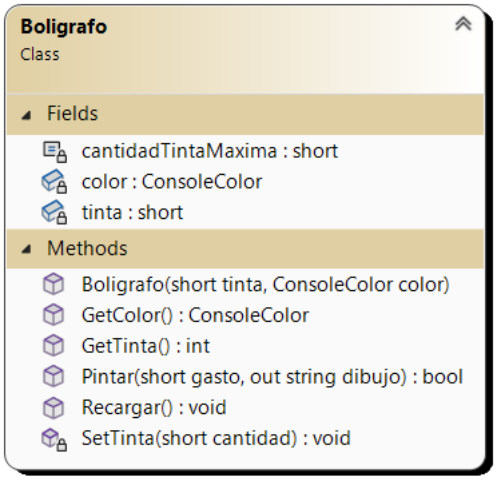
- El método `SetNotaSegundoParcial` permitirá cambiar el valor del atributo `notaSegundoParcial`.
- El método privado `CalcularPromedio` retornará el promedio de las dos notas.
- El método `CalcularNotaFinal` deberá retornar la nota del final con un número aleatorio entre 6 y 10 incluidos siempre y cuando las notas del primer y segundo parcial sean mayores o iguales a 4, caso contrario la inicializará con el valor -1.
- El método `Mostrar` utilizará `StringBuilder` para armar una cadena de texto con todos los datos de los alumnos:
  - Nombre, apellido y legajo.
  - Nota del primer y segundo parcial.
  - Promedio.
  - Nota final. Se mostrará sólo si el valor es distinto de -1, caso contrario se mostrará la leyenda *"Alumno desaprobado"*.

Crear tres instancias de la clase `Estudiante` (tres objetos) en el método `Main`. Cargar las notas del primer y segundo parcial a todos los alumnos. Dos deberán estar aprobados y uno desaprobado. Mostrar los datos de todos los alumnos.

Nota: Para darle un valor aleatorio a la nota final utilice el método de instancia Next de la clase Random.

I04. Invento argentino

En un proyecto de biblioteca de clases, crear la clase `Boligrafo` a partir del siguiente diagrama:



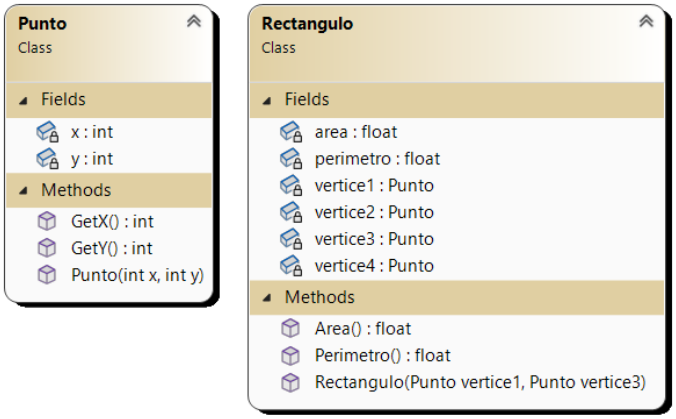
- La cantidad máxima de tinta para todos los bolígrafos será de 100. Generar una constante `cantidadTintaMaxima` en `Boligrafo` donde se guardará dicho valor.
- Generar los métodos `GetColor` y `GetTinta` para los correspondientes atributos (sólo retornarán el valor de los mismos).

- Generar un método privado `SetTinta` que valide el nivel de tinta y, si es válido, modifique el valor del atributo tinta.
  - El argumento `cantidad` contendrá la cantidad de tinta a agregar o quitar. Podrá ser positivo (cargar tinta) o negativo (gastar tinta).
  - Se deberá validar que el nivel de tinta resultante sea mayor o igual a cero y menor o igual a `cantidadTintaMaxima`. Si no es válido, no se deberá modificar el atributo ni realizar ninguna acción.
- El método `Recargar` colocará la tinta en su nivel máximo. Reutilizar código.
- El método `Pintar` restará la tinta gastada (reutilizar código). El parámetro gasto representará la cantidad de unidades de tinta a utilizar y utilizará tanta tinta como tenga disponible sin quedar en negativo. Utilizando el parámetro dibujo informará el resultado retornando tantos `*` como unidades de tinta haya gastado, por ejemplo:
  - Si no había nada de tinta retornará una cadena de texto vacía.
  - Si el nivel de tinta era 10 y la cantidad a gastar 2, entonces retornará `**`.
  - Si el nivel de tinta era 3 y la cantidad a gastar 10, entonces retornará `***`.

Agregar un proyecto de consola. En el método Main, crear un bolígrafo de tinta azul (ConsoleColor.Blue) y una cantidad inicial de tinta de 100 y otro de tinta roja (ConsoleColor.Red) y 50 de tinta. Utilizar todos los métodos y mostrar los resultados por consola. Al utilizar el método Pintar, si corresponde, se deberá dibujar por pantalla con el color de dicho bolígrafo.

I05. Prueba de geometría

En un proyecto de biblioteca de clases, crear las clases modeladas en el siguiente diagrama:



Ambas clases deberán encontrarse dentro de un espacio de nombres (namespace) llamado `Geometria`.

La clase `Punto` debe tener:

- Dos atributos privados con acceso de sólo lectura, que serán las coordenadas del punto.

- Generar los métodos `GetX` y `GetY` para los correspondientes atributos (sólo retornarán el valor de los mismos).
- Un constructor que reciba los parámetros x e y.

La clase `Rectangulo`:

- Tiene los atributos de tipo `Punto`: `vertice1`, `vertice2`, `vertice3` y `vertice4` (que corresponden a los cuatro vértices del rectángulo).
- La base de todos los rectángulos de esta clase será siempre horizontal. El constructor calculará los vértices 2 y 4 del rectángulo a partir de los vértices 1 y
- Utilizar el método `Abs` de la clase `Math` que retorna el valor absoluto de un número y será necesario para obtener la distancia entre puntos.
- El área (`base * altura`) y el perímetro (`base + altura) / 2`) se deberán calcular sólo una vez cuando se llame por primera vez. En las siguientes invocaciones de dichos métodos se deberá retornar siempre el valor calculado anteriormente.

Crear un proyecto de consola. En la clase `Program`, desarrollar un método de clase (estático) que muestre todos los datos de una instancia de `Rectángulo` que reciba como parámetro. En el método `Main` probamos las funcionalidades de las clases Punto y `Rectángulo`.

1. Instanciar un nuevo `Rectángulo`.
2. Imprimir por pantalla los valores de área y perímetro.

I06. La veterinaria

El dueño de una veterinaria nos contrató para que desarrollemos una aplicación donde pueda consultar la lista de clientes y sus mascotas.
A los clientes les interesa conocer el domicilio, el nombre y apellido y un teléfono. A un cliente se le puede asociar una sola mascota.
De las mascotas se necesita conocer su especie, su nombre y su fecha de nacimiento. Los primeros tres datos son obligatorios para dar de alta una mascota, mientras que el último arrancará vacío. De la vacuna sólo interesa conocer el nombre.

1. Crear una aplicación de consola.
2. Modelar y declarar las clases necesarias para resolver la necesidad del cliente. Hacerlo en un proyecto de biblioteca de clases.
3. Las clases deberán tener los métodos necesarios, incluyendo uno que devuelva toda la información del objeto en formato string.
4. Instanciar los siguientes objetos:
    a. Un cliente con un perro sin vacunar.
    b. Un cliente con un gato con la vacuna "Triple Felina".
    c. Un cliente con un gato sin vacunas y un perro con la vacuna contra la rabia.
5. Mostrar por consola todos los datos de los clientes instanciados y sus mascotas