**Ejercicios de programación de clases y objetos en C# para principiantes**

**Por Valentín Torassa**

**Link al GitHub con las soluciones en la descripcion del video de youtube**

**Ejercicio 1**

**Object Oriented Piracy**

Eres el líder de una pequeña tripulación pirata. Y tienes un plan. Con la ayuda de OOP, desea crear un sistema bastante eficiente para identificar barcos con un gran botín a bordo.

Desafortunadamente para ti, la gente pesa mucho en estos días, entonces, ¿cómo sabes si un barco está lleno de oro y no de personas?

Comienza escribiendo una clase/estructura de barco genérica:

public class Barco

{

Atributos

public int Cargamento;

public int Tripulacion;

public Barco(int cargamento, int tripulacion)

{

Tripulacion = tripulacion;

Cargamento = cargamento;

}

Cada vez que tus espías vean un nuevo barco entrar al muelle, crearán un nuevo objeto de barco basado en sus observaciones:

Cargamento - una estimación del peso del barco basado en qué tan bajo está en el agua.

Tripulación - el número de tripulantes a bordo.

Barcos a evaluar

Barco titanic = new Ship(20, 10);

Barco perlaNegra = new Ship(50, 7);

**Tarea del programador**

Cada miembro de la Tripulación añade 2 unidades al peso del barco. Si después de eliminar el peso de la tripulación, el Cargamento sigue siendo superior a 20, entonces vale la pena saquear el barco. ¡Cualquier barco que pese tanto debe tener mucho botín!

**Agregar el método**

public bool valeLaPena() para decidir si el barco es digno de saquear.

Si lo vale debe devolver True de otra forma devolverá False.

Esta es la forma de evaluar si el código funciona en el main:

Console.WriteLine(titanic.valeLaPena());

**Ejercicio 2**

**Crear Atributos y Constructor de la Clase Persona**

Tu tarea es crear la clase Persona en C#, que tendrá los siguientes atributos:

nombre: cadena de caracteres que almacena el nombre de la persona.

edad: entero que almacena la edad de la persona.

genero: cadena de caracteres que almacena el género de la persona (puede ser "hombre" o "mujer").

Además, la clase Persona debe tener un constructor que inicialice los atributos con los valores que se le pasen como parámetros.

Por último, la clase Persona debe tener un método hablar() que muestre en pantalla la frase "nombre, un genero de edad años está hablando", utilizando los valores de los atributos.

Te entregaremos la estructura de código de la clase Persona, en la que solo debes completar el código de los atributos y el constructor. Tu tarea es escribir el código necesario para que la clase Persona funcione correctamente.

Recuerda que tu código debe cumplir con los estándares de codificación y debes asegurarte de que el programa compile y ejecute correctamente.

class Persona

{

// Atributos

// Constructor

public Persona()

{

// Todo el código necesario para inicializar los atributos va aquí

}

// Método hablar()

public void hablar()

{

Console.WriteLine("{0}, un {1} de {2} años está hablando.", nombre, genero, edad);

}

}

**y esto es lo que va dentro del main:**

//creo un objeto de la clase persona llamada persona1 y paso los parametros al constructor

Persona persona1 = new Persona("Juan", 30, "Hombre");

//creo otro objeto de la clase persona llamada persona2 y paso los parametros al constructor

Persona persona2 = new Persona("María", 25, "Mujer");

persona1.Hablar(); //invoco el metodo hablar para la persona1

**Ejercicio 3**

**Clase Tanque**

Tu tarea es crear la clase Tanque en C#, que tendrá los siguientes atributos:

nombre: cadena de caracteres que almacena el nombre del tanque.

país: cadena de caracteres que almacena el país de origen del tanque.

año: entero que almacena el año de fabricación del tanque.

peso: entero que almacena el peso del tanque en toneladas.

Además, la clase Tanque debe tener un constructor que inicialice los atributos con los valores que se le pasen como parámetros.

Por último, la clase Tanque debe tener un método información() que muestre en pantalla toda la información del tanque, utilizando la siguiente frase: "El tanque {nombre} fue fabricado en {país} en el año {año} y pesa {peso} toneladas." Utiliza los valores de los atributos para completar la frase.

Tu tarea es escribir el código completo de la clase Tanque y asegurarte de que el programa compile y ejecute correctamente.

**Ejercicio 4**

**Crear Clase Bloque y métodos**

Tu tarea es crear la clase Bloque en C#, que tendrá los siguientes atributos:

largo: entero largo que almacena el largo del bloque.

ancho: entero que almacena el ancho del bloque.

alto: entero que almacena el alto del bloque.

Además, la clase Bloque debe tener un constructor que inicialice los atributos con los valores que se le pasen como parámetros.

Por último, la clase Bloque debe tener los siguientes métodos:

obtenerLargo(): que devuelve el valor del atributo largo.

obtenerAncho(): que devuelve el valor del atributo ancho.

obtenerAlto(): que devuelve el valor del atributo alto.

obtenerVolumen(): que devuelve el volumen del bloque, que se calcula como largo \* ancho \* alto.

obtenerAreaSuperficie(): que devuelve el área de la superficie del bloque, que se calcula como 2 \* ((largo \* ancho) + (ancho \* alto) + (largo \* alto)).

Tu tarea es escribir el código completo de la clase Bloque y asegurarte de que el programa compile y ejecute correctamente.

**Ejercicio 5**

**Tanques: Agregar método Batalla()**

En el desafío 3, creamos la clase Tanque en C#

Ahora, tu tarea es mejorar la clase Tanque agregando un nuevo método batalla() que simula una batalla entre dos tanques. Este método debe tener un parámetros: un objeto de la clase Tanque, que representará al tanque enemigo.

El método batalla() debe mostrar en pantalla la información de ambos tanques y luego simular la batalla comparando los atributos de cada uno. Si el tanque que invoca el método es más nuevo que el tanque enemigo, debe mostrar en pantalla un mensaje indicando que ha ganado la batalla.

Si son del mismo año, debe comparar sus pesos y mostrar un mensaje indicando quién ha ganado según el peso. En caso de que el tanque enemigo sea más nuevo o tenga el mismo año y un peso mayor, debe mostrar un mensaje indicando que se ha perdido.

Recuerda que debes escribir el código necesario para que el método batalla() funcione correctamente y asegurarse de que el programa compile y ejecute correctamente.

**Ejercicio 6 (Tarea)**

**Tanques: Países aliados**

En el desafío anterior, mejoraste la clase Tanque en C# agregando un nuevo método batalla() que simulaba una batalla entre dos tanques. Este método comparaba los atributos de cada tanque para determinar quién ganaba el combate.

Ahora, tu tarea es mejorar la clase Tanque para incluir la posibilidad de que haya países aliados entre sí. Esto significa que dos tanques de países aliados no deberían poder pelear entre sí.

Para ello, deberás modificar el método batalla() para que, si los tanques son de países aliados, muestre un mensaje en pantalla indicando que no pueden pelear entre sí.

En caso contrario, el método deberá seguir comparando los atributos de los tanques para determinar quién gana el combate, tal y como lo hacía en la versión anterior.

Recuerda que debes escribir el código necesario para que el método batalla() funcione correctamente y asegurarte de que el programa compile y ejecute correctamente.

Para incluir la información de los países aliados, puedes agregar un nuevo atributo aliados a la clase Tanque, que sea una lista de países aliados del país del tanque en cuestión. Además, puedes crear un método sonAliados() que reciba como parámetro otro objeto de la clase Tanque y devuelva true si ambos tanques son de países aliados o false en caso contrario.

Una vez que hayas modificado el método batalla() para incluir la comprobación de los países aliados, asegúrate de probar el programa con diferentes combinaciones de tanques para asegurarte de que funciona correctamente.