Parcial I

Programación Orientada a Objetos
27/09/2020

Parte I (30)

1. ¿Qué es la programación orientada a objetos?
2. Nombre tres características/propiedades de la programación orientada a objetos.
3. ¿Qué es un objeto?
4. ¿Qué es un constructor?
5. ¿Qué es una clase?

Parte II (30)

Considerando los siguientes casos y bloques de código, indique cuáles funcionarían y cuáles no, justificando los casos incorrectos y qué cambios podrían hacerse para que funcione correctamente junto con las decisiones de diseño tomadas (éste cambio por tal motivo).

```
public interface IVehiculo {
	double getVelocidadMaximaKmh();
}

public class Vehiculo : IVehiculo {
	public string _marca;
	public string _modelo;
	private int _año;
	private double _velocidadMaxima; //ms

	public Vehiculo(
		string xMarca,
		string xModelo,
		int xAño
	)
	{
		this._marca = xMarca;
		this.xModelo = _modelo;
		this._año = xAño;
	}

	public double getVelocidadMaxima() {
		return this._velocidadMaxima;
	}
}
```

```
public class Camioneta : Vehiculo {
      private double _tamañoCaja // m3;
      private bool _cajaCerrada;

      public Camioneta(
            string xMarca,
            string xModelo,
            int xAño,
            double xVelocidadMaxima,
            double xTamañoCaja,
            bool xCajaCerrada
      )
      {
            this._marca = xMarca;
            this.xModelo = _modelo;
            this._año = xAño;
            this._tamañoCaja = xTamañoCaja;
            this._velocidadMaxima = xVelocidadMaxima;
            this._cajaCerrada = this.xCajaCerrada;
      }

      public void getTamañoCaja() {
            return this._tamañoCaja;
      }

      private void setCajaCerrada(string xCajaCerrada) {
            this._cajaCerrada = this.xCajaCerrada;
      }
}

CASO 01 // IVehiculo objVehiculo = new IVehiculo("Toyota",
"Hilux", 2016);
CASO 02 // Vehiculo objVehiculo2 = new Vehiculo("Fiat", "Strada",
2013, 200.05);
CASO 03 // Vehiculo objVehiculo3 = new Vehiculo("Suzuki",
"Maruti", 2013);
Caso 04 // objVehiculo3.modelo = "Alto";
Caso 05 // objVehiculo3.setModelo("Alto 800");
Caso 06 // objVehiculo3.setModelo("Alto 800");
Caso 07 // Camioneta objCamioneta = new Vehiculo("Volwkswagen",
"Suran", 2014, 240, 40, true);
Caso 08 // Vehiculo objCamioneta2 = new Camioneta("Chevrolet
", "Captiva", 2016, 250, 20, true);
Caso 09 // objCamioneta2 .getTamañoCaja();
Caso 10 // objCamioneta2 .setTamañoCaja(25);
Caso 11 // string vKmh = objVehiculo3.getVelocidadMaximaKmh();
Caso 12 // objVehiculo3.setTamañoCaja(10);
Caso 13 // objCamioneta2 .getVelocidadMaxima();
```

Parte III (40)

Se desea un programa para un taller mecánico que permita ingresar una lista de **clientes**, junto con sus respectivos **vehículos** e información acerca de éstos.

Cada uno de los clientes posee un `identificador` (cédula de identidad), `nombre` y `apellido`.

Cada cliente tiene asociado uno o más vehículos, identificados por su `matrícula`. Los vehículos poseen como atributos `modelo`, `marca`, `año`, f`echa de último mantenimiento`, y `fecha de próximo mantenimiento`.

En base a ese modelo, el programa debe poder:

- Ingresar clientes.
- Asociar uno o más vehículos a un cliente.
- Buscar un cliente en particular (puede ser un método estático) e imprimir sus datos.
- Listar los vehículos asociados a un cliente en particular (puede ser un método estático) junto con todos los datos del cliente y su vehículo.
- Modificar la fecha de último mantenimiento para un vehículo (enviando número de matrícula), modificando así la fecha de próximo mantenimiento para dentro de cinco meses (fecha próxima = fecha último mantenimiento + 5).

Crear una clase `VehiculoAntiguo`, que posea el mismo comportamiento que `Vehiculo` pero que al momento de modificar la fecha de último mantenimiento, la fecha de próximo mantenimiento sea para dentro de dos meses (fecha próxima = fecha último mantenimiento + 2).

Agregar el atributo color y que el programa siga corriendo.

No es necesario ingresar los datos de forma dinámica (aunque sí recomendado).

Justificar las decisiones de diseño.