UTEC
Universidad Tecnológica

Diseño de clases e implementación de Objetos

Ejercicio 1: Creación de una Clase `Libro`

1. Clase `Libro`:
   - Diseña una clase llamada `Libro` con los siguientes atributos:
     - Título
     - Autor
     - Año de publicación
     - Número de páginas

2. Constructor:
   - Crea un constructor que permita inicializar todos los atributos al crear un objeto `Libro`.

3. Método `MostrarInformacion()`:
   - Implementa un método llamado `mostrarInformacion()` que imprima en la consola todos los detalles del libro.

4. Uso de la Clase:
   - Crea instancias de la clase `Libro` y muestra su información utilizando el método `mostrarInformacion()`.

Ejercicio 2: Creación de una Clase `Estudiante`

1. Clase `Estudiante`:
   - Diseña una clase llamada `Estudiante` con los siguientes atributos:
     - Nombre
     - Edad
     - Carrera

2. Constructor:
   - Crea un constructor que permita inicializar todos los atributos al crear un objeto `Estudiante`.

3. Método `CambiarCarrera()`:
   - Implementa un método llamado `cambiarCarrera()` que permita cambiar la carrera del estudiante.

UTEC
Universidad Tecnológica

4. Uso de la Clase:
   - Crea instancias de la clase `Estudiante` y realiza cambios en la carrera utilizando el método `cambiarCarrera()`.

Ejercicio 3: Creación de una Clase `Círculo`

1. Clase `Círculo`:
   - Diseña una clase llamada `Circulo` con el atributo radio.

2. Constructor:
   - Crea un constructor que permita inicializar el radio al crear un objeto `Circulo`.

3. Método `CalcularArea()`:
   - Implementa un método llamado `calcularArea()` que calcule y devuelva el área del círculo.

4. Método `CalcularPerimetro()`:
   - Implementa un método llamado `calcularPerimetro()` que calcule y devuelva el perímetro del círculo.

5. Uso de la Clase:
   - Crea instancias de la clase `Circulo` y realiza cálculos de área y perímetro utilizando los métodos correspondientes.

Relaciones entre clases, herencia y polimorfismo

Ejercicio 1:

Supongamos que tenemos las clases `Vehiculo` y `Automovil`. El `Automovil` es un tipo específico de `Vehiculo`. Define las clases y muestra cómo el automóvil hereda de vehículo.

Ejercicio 2:

Crea una clase base `Figura` con métodos para calcular área y perímetro. Luego, define clases derivadas como `Circulo` y `Cuadrado`. Utiliza el polimorfismo para calcular el área y el perímetro de diferentes figuras sin conocer su tipo exacto.

UTEC
Universidad Tecnológica

Ejercicio 3:

Imagina una clase `Persona` y otra clase `Direccion`. Modifica la clase `Persona` para que contenga un objeto `Direccion`, estableciendo así una relación de composición.

Ejercicio 4

Define una clase `Animal` con un método `hacerSonido()`. Luego, crea clases derivadas como `Perro` y `Gato` que heredan de `Animal` y sobrescriben el método `hacerSonido()` con sonidos específicos para cada animal.

Ejercicio 5

Crea una interfaz `Imprimible` con un método `imprimir()`. Luego, define clases como `Documento`, `Foto` y `Video` que implementan la interfaz. Utiliza el polimorfismo para imprimir diferentes tipos de objetos sin conocer su tipo exacto.

Sobrecarga de métodos y constructores

Ejercicio 1: Sobrecarga de Métodos

Define una clase llamada `Operaciones` que contenga varios métodos llamados `sumar`. Cada método `sumar` debe aceptar un número diferente de parámetros (por ejemplo, dos, tres y cuatro) y realizar la suma correspondiente. Demuestra la sobrecarga de métodos utilizando estos métodos `sumar` en un programa principal.

Ejercicio 2: Sobrecarga de Constructores y Herencia (Pájaros)

Crea una clase base llamada `Ave` con los siguientes atributos:

- `nombre` (String): el nombre del ave.
- `color` (String): el color del plumaje.

La clase debe tener un constructor que acepte el nombre y el color del ave.

Luego, crea dos clases derivadas, `Canario` y `Aguila`, que hereden de la clase `Ave`. Ambas clases deben tener atributos adicionales y métodos específicos:

1. Canario:
    - Atributo adicional: `canto` (String): descripción del canto del canario.
    - Método adicional: `cantar()`, que imprime el canto del canario.
2. Águila:
    - Atributo adicional: `envergadura` (double): envergadura del águila en metros.
    - Método adicional: `volarAlto()`, que imprime un mensaje indicando que el águila está volando a gran altura.

Realiza la sobrecarga de constructores en las clases derivadas para permitir la creación de instancias con información adicional (por ejemplo, canto para el canario y envergadura para el águila). Luego, en el método `main`, crea instancias de `Canario` y `Aguila` y muestra sus atributos y llama a sus métodos específicos.

Ejercicio 3: Sobrecarga de Constructores y Herencia (Personajes de Juego)

Crea una clase base llamada `Personaje` con los siguientes atributos:

- `nombre` (String): el nombre del personaje.
- `puntosSalud` (int): los puntos de salud del personaje.

La clase debe tener un constructor que acepte el nombre y los puntos de salud del personaje.

Luego, crea dos clases derivadas, `Guerrero` y `Mago`, que hereden de la clase `Personaje`. Ambas clases deben tener atributos adicionales y métodos específicos:

1. Guerrero:
   - Atributo adicional: `puntosAtaque` (int): los puntos de ataque del guerrero.
   - Método adicional: `atacar()`, que imprime un mensaje indicando que el guerrero está atacando.

2. Mago:
   - Atributo adicional: `puntosMagia` (int): los puntos de magia del mago.
   - Método adicional: `lanzarHechizo()`, que imprime un mensaje indicando que el mago está lanzando un hechizo.

Luego, crea instancias de las clases `Guerrero` y `Mago` con diferentes configuraciones y muestra cómo funcionan sus métodos específicos. Utiliza la sobrecarga de constructores para permitir la creación de personajes con información adicional según su tipo.

UTEC
Universidad Tecnológica

Ejercicio 4: Sistema de Registro de Empleados

En el contexto de un sistema de registro de empleados para una empresa, se desea implementar la clase `Empleado` con funcionalidades diversas. La clase debe tener un constructor que permita inicializar el nombre y el salario base de un empleado.

1. Método para Calcular Salario Neto:
   - Implementar un método para calcular el salario neto del empleado. Este cálculo debe considerar deducciones estándar.

2. Método Sobrecargado para Calcular Salario Neto con Deducciones Adicionales:
   - Sobrecargar el método de cálculo de salario neto para permitir deducciones adicionales. Esto brinda flexibilidad al sistema para manejar casos específicos de deducciones.

3. Método para Imprimir Información Detallada del Empleado:
   - Desarrollar un método que imprima información detallada del empleado, incluyendo su nombre, salario base y otros detalles relevantes.

4. Método Sobrecargado para Gestionar Beneficios Adicionales:
   - Implementar un método que permita gestionar beneficios adicionales para el empleado. Pueden ser beneficios especiales o personalizados que no estén contemplados en el salario base.

5. Explorar Más Opciones de Sobrecarga:
   - Analizar otras operaciones relacionadas con empleados que podrían beneficiarse de la sobrecarga de métodos. Por ejemplo, podrías considerar la posibilidad de añadir métodos para gestionar bonificaciones, evaluar el desempeño, entre otros.