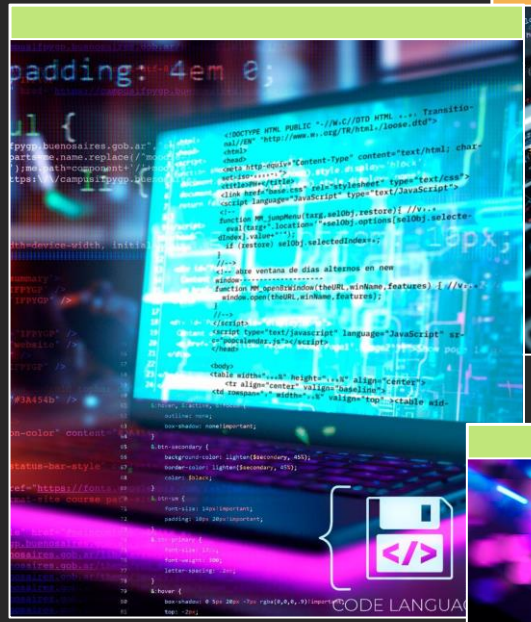
Programacion II [PRACTICA]

Programación Orientada a Objetos [P.O.O]



1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0

1 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0

OBJETOS & CLASE

Objetos que son entidades que combinan estado (propiedades o datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto).

Las clases son plantillas que agrupan comportamiento (métodos) y estados (atributos) de los futuros objetos.

ATRIBUTOS

- **Nombre:** Wade Winston Wilson
- **Alias:** Deadpool
- **Sexo:** Masculino
- **Estatura:** 1,88m
- **Peso:** 81,5 Kgr
- **Nacionalidad:** Canada
- **Raza:** Humano Mutante

METODOS

- Caminar
- Correr
- Atacar
- Defenderse
- Regenerarse
- Uso de Armas de Fuego
- Uso de Katanas

OBJETOS & CLASE

SUPERHEORES

Nombre	Fuerza
Edad	Resistencia
Estatura	
Caminar()	Defenderse()
Correr ()	
Atacar()	

→ **Clases**

Atributos

Son las características de un objeto

Métodos

Son las acciones que el objeto puede realizar

VEHICULOS

Nombre	Aceleración
Modelo	VelActual
VelMax	
Arrancar()	GirarIzq()
Frenar()	GirarDerecha()
Acelerar()	AvanzarAdelante()
	Reversa()

→ **Clases**

Atributos

Métodos

1 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0

CLASE

```
1  class LibroCalificaciones
2  {
3      // Miembros públicos:
4  public:
5      // Atributos:
6      char NombreCurso[100]="";
7      int NAlumnos=0;
8      // Método
9      void mostrarMensaje()
10         {
11             cout << "Bienvenido al Libro de calificaciones del Curso " << NombreCurso << endl;
12             cout << "Numero de Alumnos:  " << NAlumnos << endl;
13         }
14 };
```

CLASE

```
int main()
{
    // Instanciamiento:
    LibroCalificaciones ProgramacionII;
    // Asignación de valores a los atributos del objeto
    strcpy(ProgramacionII.NombreCurso, "Programacion II");
    ProgramacionII.NAlumnos=65;
    // Llamada a un método:
    ProgramacionII.mostrarMensaje();
    system("Pause");
}
```

REPASO DE CONCETOS

Instanciamiento: es crear un objeto a partir de una clase.

Abstracción consiste en ocultar los detalles complejos y mostrar solo lo esencial.

Encapsulamiento es la práctica de mantener los atributos y métodos internos de un objeto protegidos del acceso externo no autorizado.

Los miembros públicos son accesibles desde fuera de la clase, **los privados** son accesibles solo desde dentro de la clase, y los protegidos son accesibles desde la clase y sus subclases.

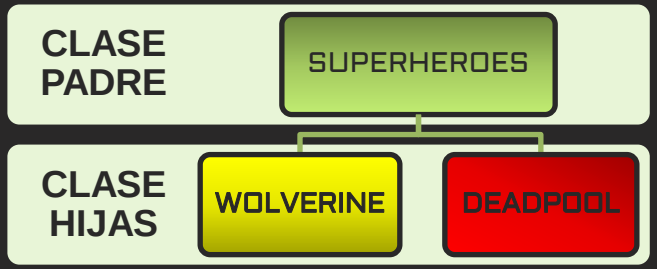
CLASE

```
class LibroCalificaciones
{
// Miembros privado:
private:
// Atributos Privados:
    char nombreCurso[100]="";
    int NAlumnos=0;

// Miembros públicos:
public:
    // Método
    void EstablecerNombreCurso()
    {
        cout << "Ingrese el Nombre del Curso: " << endl;
        cin << nombreCurso;
    }
    void EstablecerNumeroAlumnos()
    {
        cout << "Cantidad de Alumnos: " << endl;
        cin << NAlumnos;
    }
    void mostrarMensaje()
        {
            cout << "Bienvenido al Libro de calificaciones del Curso " << NombreCurso << endl;
            cout << "Numero de Alumnos:  " << NAlumnos << endl;
        }
};
```

- Los miembros públicos son accesibles desde fuera de la clase.
- Los miembros privados son accesibles solo desde dentro de la clase.

HERENCIA



La herencia permite crear clases que reutilizan, extienden y modifican el comportamiento definido en otras clases. La clase cuyos miembros se heredan se denomina clase PADRE y la clase que hereda esos miembros se denomina clase HIJAS.

WOLVERINE

Nombre	Fuerza
Edad	Resistencia
Estatura	Nivel de Energia
Caminar()	Defenderse()
Correr ()	
Atacar()	

DEADPOOL

Nombre	Fuerza
Edad	Resistencia
Estatura	Nivel de Energia
Caminar()	Defenderse()
Correr ()	
Atacar()	

HERENCIA

```cpp
class LibroCalificaciones
{
private:
    // Atributos Privados:
    char nombreCurso[100] = "";
    int NAlumnos = 0;
protected:
    // Atributo Protected:
    char semestre[10] = "N/A";
public:
    // Métodos Públicos:
    void EstablecerNombreCurso()
    {
        cout << "Ingrese el Nombre del Curso: ";
        cin >> nombreCurso;
    }
    void EstablecerNumeroAlumnos()
    {
        cout << "Cantidad de Alumnos: ";
        cin >> NAlumnos;
    }
    void EstablecerSemestre()
    {
        cout << "Ingrese el Semestre del Curso: ";
        cin >> semestre;
    }
    void mostrarMensaje()
    {
        cout << "Bienvenido al Libro de calificaciones del Curso " << nombreCurso << endl;
        cout << "Numero de Alumnos: " << NAlumnos << endl;
        cout << "Semestre: " << semestre << endl;
    }
    // Método para obtener el nombre del curso (necesario para la clase derivada)
    const char* ObtenerNombreCurso() const
    {
        return nombreCurso;
    }
};
```

```cpp
class LibroCalificacionesPracticos : public LibroCalificaciones
{
private:
    // Nuevos atributos para la clase derivada
    char nombreProfesor[100] = "";
    char ubicacionClase[100] = "";

public:
    // Métodos para establecer los nuevos atributos
    void EstablecerNombreProfesor()
    {
        cout << "Ingrese el Nombre del Profesor: ";
        cin >> nombreProfesor;
    }

    void EstablecerUbicacionClase()
    {
        cout << "Ingrese la Ubicación de la Clase: ";
        cin >> ubicacionClase;
    }

    // Sobrescribiendo el método mostrarMensaje para incluir la nueva información
    void mostrarMensajeDetallado()
    {
        // Llamada al método de la clase base
        mostrarMensaje();
        cout << "Profesor: " << nombreProfesor << endl;
        cout << "Ubicación de la Clase: " << ubicacionClase << endl;
    }

    // Método para cambiar el semestre desde la clase derivada
    void CambiarSemestre(const char* nuevoSemestre)
    {
        strcpy(semestre, nuevoSemestre);
    }
};
```

1 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0

Retos de Programacion

01 Clase Vehiculos

Crea un programa que tenga una clase llamada «Vehículo». El programa debe simular el funcionamiento de los Vehículos (Arrancar, Frenar, Acelerar, Mostrar Estado Actual)

REQUISITOS:

El usuario puede determinar en cada vehículo:

- Nombre
- Modelo
- Velocidad Maxima
- Velocidad Actual
- Aceleracion

ACCIONES:

- Mostrar los datos del vehículo.
- Mostrar Cuanto se demora de ir a una distancia determinada.
- Mostrar cuanto se demora en llegar a un 100 km/h en una recta sin obstaculos (ideal)

02 Sistema de Gestión de Libros

Implementa una clase Libro que represente un libro en una biblioteca. La clase debe incluir atributos para almacenar el título del libro, el autor, el número de páginas, y si el libro está prestado o disponible.

Atributos:

- **Titulo:** Un string que almacena el título del libro.
- **Autor:** Un string que almacena el nombre del autor.
- **Paginas:** Un entero que almacena el número de páginas del libro.
- **Prestado:** Un booleano que indica si el libro está prestado o no.

Métodos:

- **MostrarInfo():** Muestra la información completa del libro (título, autor, número de páginas, estado de préstamo).
- **Prestar():** Cambia el estado del libro a "prestado" si no lo está ya, y muestra un mensaje correspondiente.
- **Devolver():** Cambia el estado del libro a "disponible" si está prestado, y muestra un mensaje correspondiente.

Tareas:

1. Implementa la clase Libro con los atributos y métodos descritos.
2. En la función main(), crea un objeto Libro, muestra su información, presta el libro, y luego devuelve el libro. Finalmente, vuelve a mostrar la información del libro.

03 Gestión de Empleados

Crea una clase Empleado que represente a un empleado en una empresa. La clase debe gestionar la información básica del empleado y permitir calcular su salario mensual en función de su salario por hora y las horas trabajadas.

Atributos:

- **nombre:** Un string que almacena el nombre del empleado.
- **puesto:** Un string que almacena el puesto del empleado.
- **salarioPorHora:** Un float que almacena el salario por hora del empleado.
- **horasTrabajadas:** Un float que almacena el número de horas trabajadas en un mes.

Métodos:

- **CalcularSalario():** Calcula y devuelve el salario mensual del empleado (salarioPorHora * horasTrabajadas).
- **MostrarInfo():** Muestra la información del empleado (nombre, puesto, salario por hora, horas trabajadas, y salario mensual).

Tareas:

1. Implementa la clase Empleado con los atributos y métodos descritos.
2. En la función main(), crea un objeto Empleado, establece los valores de los atributos, muestra la información del empleado, y calcula su salario mensual.

04 Sistema de Gestión de Productos

Desarrolla una clase Producto que represente un producto en una tienda. La clase debe incluir métodos para gestionar el inventario, es decir, permitir la compra y venta de productos, y mostrar el estado actual del inventario.

Atributos:

- **nombre**: Un string que almacena el nombre del producto.
- **precio**: Un float que almacena el precio del producto.
- **stock**: Un entero que almacena la cantidad disponible en el inventario.

Métodos:

- **Vender(int cantidad)**: Reduce el stock en la cantidad especificada si hay suficiente stock disponible. Muestra un mensaje confirmando la venta o indicando que no hay suficiente stock.
- **Reabastecer(int cantidad)**: Aumenta el stock en la cantidad especificada.
- **MostrarInfo()**: Muestra la información del producto (nombre, precio, stock).

Tareas:

- Implementa la clase Producto con los atributos y métodos descritos.
- En la función main(), crea un objeto Producto, muestra su información, realiza algunas ventas y reabastecimientos, y vuelve a mostrar la información del producto.

05 Deadpool vs Wolverine

Crea un programa que simule la pelea y determine un ganador. El programa simula un combate por turno, donde cada protagonista posee unos puntos de vida iniciales, un daño de ataque variable y diferentes cualidades de regeneración y evasión de ataque.

REQUISITOS:

1. El Usuario debe determinar la vida inicial de cada protagonista
2. Cada personaje puede impartir un daño aleatorio:
    - **Deadpool**: *Entre 10 y 100*
    - **Wolverine**: *Entre 10 y 120*
3. Si el daño es el máximo, el persona que lo recibe ni ataca en el siguiente turno, ya que tiene que regenerarse (pero no aumenta vida)
4. Cada Personaje puede evitar el ataque contrario:
    - **Deadpool**: *25% de posibilidad*
    - **Wolverine**: *20% de posibilidad*
5. Un personaje pierde si sus puntos de vida llegan a cero o menos.

ACCIONES:

- Simula una batalla
- Muestra el numero del turno (pausa de 1 segundo entre turno)
- Muestre que pasa en cada Turno.
- Muestre la vida en cada Turno
- Muestre el resultado final