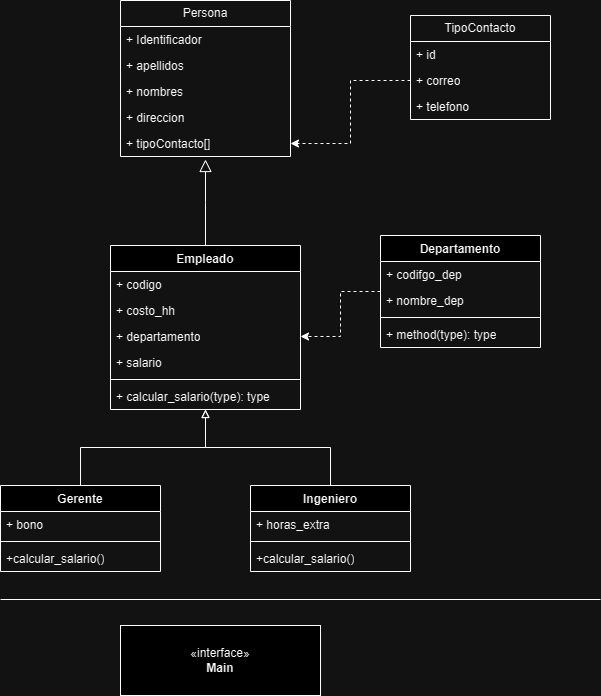
PRACTICA CALIFICADA 02

Instrucciones: crear una carpeta “apellidos\_nombres\_practica02”, no se calificarán los ejercicios que no cuenten con esta estructura.

Se cuenta con el siguiente diagrama de clases:



1. Determine la clase abstracta
2. Mencione las clases involucradas en herencia
3. Mencione el tipo de relación de las clases
4. Mencione las clases involucradas con el concepto de Polimorfismo
5. Implemente las clases y sus métodos necesarios
6. Mencionar de una clase el concepto de encapsulamiento
7. Implemente la sobrecarga de métodos
8. Explique la jerarquía de clases a partir del diagrama
9. Implemente la clase principal (main), que cumpla las siguientes condiciones:
   1. Lista de 5 personas
   2. 4 personas son ingenieros
   3. 1 persona es gerente
   4. Determinar el salario de las 5 personas, para los ingenieros se determina 14 horas extra, y para el gerente el bono es de 1500
   5. Mostrar los datos y su salario de los ingenieros: apellidos, nombres, departamento, dirección, salario final
   6. Mostrar los datos del gerente: apellidos, nombres, departamento, dirección, salario final
   7. Cambiar el bono del gerente: 2500

a) Clase Abstracta:

La clase abstracta en este caso sería la clase Empleado, ya que tiene un método virtual puro (CalcularSalario ( ) ) que se implementa en las clases derivadas (Gerente e Ingeniero). Las clases abstractas proporcionan una interfaz común para sus clases derivadas.

b) Clases involucradas en herencia:

Gerente y Ingeniero heredan de la clase abstracta Empleado.

TipoContacto se utiliza dentro de la clase Persona.

c) Tipo de relación de las clases:

Relación de herencia: Gerente e Ingeniero heredan de Empleado.

Asociación: Persona tiene una asociación con TipoContacto.

d) Clases involucradas con el concepto de Polimorfismo:

Empleado, Gerente, e Ingeniero están involucradas en el polimorfismo debido al método virtual puro CalcularSalario().

e) Implementación de las clases y sus métodos necesarios:

He proporcionado la implementación anterior de las clases y métodos.

f) Clase con el concepto de encapsulamiento:

Todas las clases (Persona, TipoContacto, Departamento, Empleado, Gerente, Ingeniero) tienen variables miembro privadas y métodos públicos/miembro para acceder a ellas. Esto sigue el concepto de encapsulamiento, donde los detalles internos de una clase están ocultos y solo se accede a ellos a través de métodos públicos.

g) Implementación de la sobrecarga de métodos:

No hay ejemplos claros de sobrecarga de métodos en el código proporcionado. Sin embargo, podrías sobrecargar métodos en una clase al definir varias versiones del mismo método con diferentes parámetros o tipos de retorno.

h) Jerarquía de clases a partir del diagrama:

* La jerarquía de clases es la siguiente:
* Persona
  1. TipoContacto
* Empleado
  1. Gerente
  2. Ingeniero
* Departamento

1. Implementación de la clase principal (main):

int main() {

// personas

Persona persona1(1001, "Apellidos1", "Nombres1", "Dirección1");

Persona persona2(1002, "Apellidos2", "Nombres2", "Dirección2");

Persona persona3(1003, "Apellidos3", "Nombres3", "Dirección3");

Persona persona4(1004, "Apellidos4", "Nombres4", "Dirección4");

Persona persona5(1005, "Apellidos5", "Nombres5", "Dirección5");

// TipoContacto para las personas

TipoContacto contacto1(1, "correo1@example.com", "123456789");

TipoContacto contacto2(2, "correo2@example.com", "987654321");

persona1.agregarTipoContacto(contacto1);

persona2.agregarTipoContacto(contacto2);

persona3.agregarTipoContacto(contacto1);

persona4.agregarTipoContacto(contacto2);

persona5.agregarTipoContacto(contacto1);

// Departamento

Departamento departamento(101, "Ventas");

// Empleados

Gerente gerente(2001, 20.0, departamento, 1500.0);

Ingeniero ingeniero1(2002, 15.0, departamento, 14);

Ingeniero ingeniero2(2003, 15.0, departamento, 14);

Ingeniero ingeniero3(2004, 15.0, departamento, 14);

Ingeniero ingeniero4(2005, 15.0, departamento, 14);

// Asignar empleados a personas

persona1.mostrarInformacion();

cout << "Salario final: " << ingeniero1.CalcularSalario() << "\n\n";

persona2.mostrarInformacion();

cout << "Salario final: " << ingeniero2.CalcularSalario() << "\n\n";

persona3.mostrarInformacion();

cout << "Salario final: " << ingeniero3.CalcularSalario() << "\n\n";

persona4.mostrarInformacion();

cout << "Salario final: " << ingeniero4.CalcularSalario() << "\n\n";

persona5.mostrarInformacion();

cout << "Salario final del Gerente: " << gerente.CalcularSalario() << "\n\n";

// Cambiar el bono del gerente

gerente.CambiarBono(2500.0);

cout << "Nuevo salario final del Gerente: " << gerente.CalcularSalario() << "\n";

return 0;

}