Ayudantia n°9

Repaso Herencia

Diego Mena

Herencia

La herencia se refiere al proceso de crear una nueva clase tomando como base otra clase. Esta nueva clase, conocida como clase derivada, hereda los atributos y funciones de la clase base. Al usar herencia, la clase derivada puede acceder y utilizar las variables y funciones de la clase base, además de definir sus propios atributos y métodos específicos.

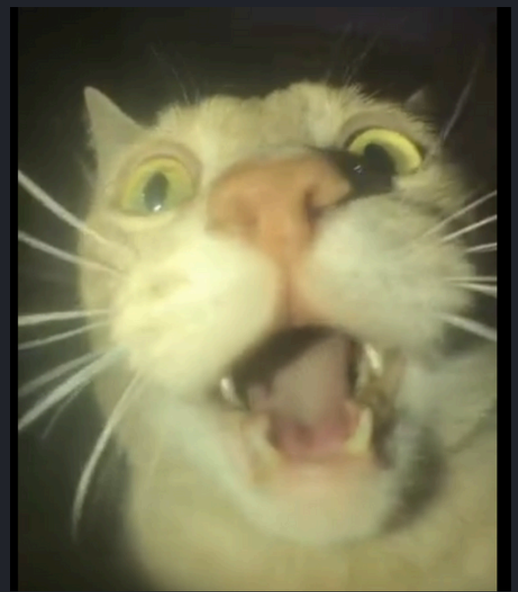
```cpp
#include <iostream>
using namespace std;
class clase1
{
    protected:
        string nombre;
    public:
        clase1(string nombre)
        {
            this->nombre=nombre;
        }
        void Imprimir()
        {
            cout<<"El nombre es: "<<nombre<<endl;
        }
};
class clase2 : public clase1
{
    private:
        int edad;
    public:
        clase2(string nombre, int edad) : clase1(nombre)
        {
            this->edad=edad;
        }
    void Imprimir2()
    {
        Imprimir();
        cout<<"La edad es: "<<edad<<endl;
    }
};

int main()
{
    clase2 * objeto1= new clase2("Seba", 12);
    objeto1->Imprimir2();
    cout<<"-----------------------------------------------"<<endl;
    objeto1->Imprimir();

}
```

Sintaxis y Ejemplo

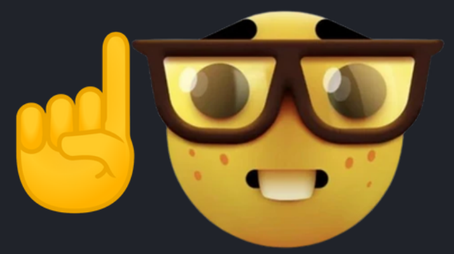
El protected sirve para poder usar las variables en las otras clases, debido a que si ponemos private no podremos hacer uso de estas.



Polimorfismo

Gracias al polimorfismo, podemos hacer que dos objetos de diferentes clases, que están relacionadas por herencia, respondan de manera distinta al mismo comando. Esto se logra mediante el uso de la palabra clave virtual (que se coloca antes del tipo de retorno, como void o int, según la función). Al declarar un método como virtual, permitimos que el objeto responda de forma diferente cuando se llama a dicho método, dependiendo a que clase paertenesca al objeto.

```
Class Animal
{
protected: ...
public:...
void sonido()
{
cout<<"bla bla"<<endl;
}
};
```



```
Class Perro: public Animal
{
private: ...
public:...
Virtual void sonido()
{
cout<<"Guau"<<endl;
}
};
```

```
Class Gato: public Animal
{
private: ...
public:...
Virtual void sonido()
{
cout<<"miau"<<endl;
}
};
```

```
int Main()
{
Perro* Obj= new Perro();
Gato* Obj1= new Gato();

Obj->Sonido(); //Guau
Obj1->Sonido(); //Miau
}
```

Ejercicio

Se requiere desarrollar un sistema de seguros que permita almacenar y mostrar información de diferentes tipos de seguros, incluyendo Seguro de Hogar, Seguro de Parcela, Seguro Automotriz y Seguro de Vida.

- Seguro: Es la clase base para todas las demás (excepto la clase Parcela). Sus parámetros son: int numpol, string rut, string nombre, int monto.

- Hogar: Tiene los parámetros: string incendio, string terremoto, string inundación.

- Parcela: Es la única que toma como clase padre a la clase Hogar. Sus atributos son: int m2, int porc.

- Automotriz: Tiene los atributos: string patente, int anho.

- Vida: Tiene el atributo: string beneficiario.

Todas las clases tienen un método imprimir correspondiente, y las clases hijas deben usar el imprimir de su padre dentro de su propio método imprimir.
Usar virtual void para los métodos imprimir.

Contacto {

+56 9 6070 8865

diego.mena2@mail.udp.cl

https://github.com/Dmena1/Ayudantia-prograAvanzada

}