Ayudantía 5

Herencia

Benjamín Aceituno

Programación Avanzada S09

Herencia

Con herencia nosotros podemos crear otras clases que derivan de otra clase.

Al crear una nueva clase nosotros podemos indicar que esta clase heredará otras subclases, en resumen, tendremos una clase base que tendrá subclases con los mismos atributos y métodos de la clase base

Ejemplos...

Animales

- Gato
- Perro
- Caballo
- Hámster

Instrumentos

- Guitarra
- Bajo
- Ukulele
- Teclado

Persona

- Estudiante
- Profesor
- Ayudante

Vehículo

- Auto
- Moto
- Camión
- Bicicleta

Dispositivo

- Celular
- Computador
- Tablet
- Notebook

Producto

- Ropa
- Alimento
- Electrodomestico

Características

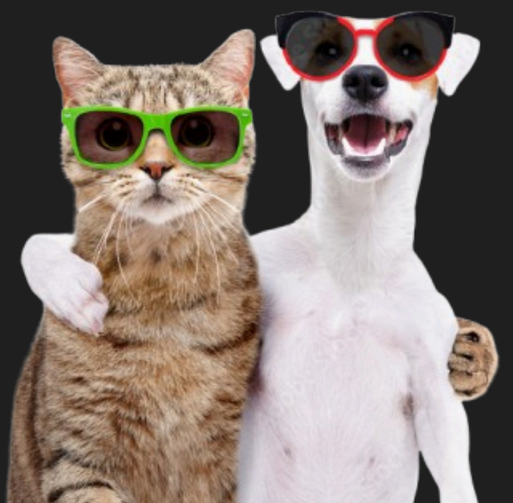
La clase derivada hereda todas las características de la clase base, es decir, hereda sus métodos y atributos.

La clase derivada puede implementar nuevos métodos y atributos, ejemplo, tenemos la clase animal que deriva gato y perro, gato podrá maullar y perro podrá ladrar.

```cpp
#include <iostream>
using namespace std;

class ClaseBase {
protected: // Ahora usaremos protected para la clase base, para queda pueda ser heredable
    // atributos
public:

    // constructor

    // métodos

};

class ClaseDerivada : public ClaseBase { // La clase derivada es heredada de clase base
public: // subClase(atributos) : claseBase(atributos)
    // ClaseDerivada(atributos con sus tipos) : ClaseBase(atributos sin tipos) {}

    // métodos propios de clase derivada
};
```

Ejercicio

Realice un programa en C++ usando herencia que tenga la clase base Animal y tenga dos clases heredadas gato y perro.
Estas dos clases tienen caracteristicas propias como poder ladrar y maullar. El animal tiene nombre y edad, además de un método para mostrar la información del animal en pantalla.



```cpp
class Animal {
protected: // Ahora usaremos protected para la clase base, para queda pueda ser heredable
    string nombre;
    int edad;
public:
    Animal(string nombre, int edad) {
        this->nombre = nombre;
        this->edad = edad;
    }

    void mostrarInfo() { // Metodo que puede usar gato y perro
        cout << "Nombre: " << nombre << ", Edad: " << edad << " años" << endl;
    }
};

class Perro : public Animal { // La clase perro es heredada de la clase animal
public: // subClase(atributos) : claseBase(atributos)
    Perro(string nombre, int edad) : Animal(nombre, edad) {}

    void ladrar() { // Metodo de la clase Perro
        cout << nombre << " está ladrando: ¡Guau, guau!" << endl;
    }
};
```

Aquí tenemos la clase base, usamos protected en vez de private, esto nos permite que otras clases puedan acceder al contenido de la clase.

Este método al estar dentro de la clase base puede ser usado por todas sus subclases

Sintaxis:

class (Subclase) : public (Clase base)

public:

Subclase(atributos con sus tipos) : claseBase(atributos)

Cada objeto va a llamar al método correspondiente, si nos fijamos acá gato1 no puede ladrar porque este es un método de perro, pero si puede mostrarInfo ya que esto es parte de la clase base.

```cpp
int main() {
    Perro *perro1 = new Perro("Vicente", 3);
    Gato *gato1 = new Gato("Pepe", 2);

    cout << "Información del Perro:" << endl;
    perro1->mostrarInfo();
    perro1->ladrar();
    cout << "Información del Gato:" << endl;
    gato1->mostrarInfo();
    gato1->maullar();

    gato1->ladrar();

    return 0;
}
```

Ejercicio

La prestigiosa tienda de música YellowHouse vende Instrumentos de la clase guitarra, bajo y batería.
Estos instrumentos pueden ser comprados por distintas personas que tienen una ocupación en una banda, donde estos pueden ser guitarristas, bajistas y bateristas. La tienda debe mostrar su inventario de instrumentos, donde cada instrumentos tiene marca y precio. Las personas deben ser capaces de presentarse, y como el guitarrista es muy humilde, puede tocar un solo de guitarra, donde se mostrará por pantalla su épico solo de guitarra.

```cpp
#include <iostream>
#include <string>
using namespace std;

class Instrumento {
protected:
    string tipo;
    string marca;
    float precio;
public:
    Instrumento(string tipo, string marca, float precio) {
        this->tipo = tipo;
        this->marca = marca;
        this->precio = precio;
    }

    void mostrarInfo() {
        cout << "Instrumento: " << tipo << " - Marca: " << marca << ", Precio: $" << precio << endl;
    }
};

class Guitarra : public Instrumento {
public:
    Guitarra(string marca, float precio) : Instrumento("Guitarra", marca, precio) {}

    void tocarUnSolo(){
        cout << "Estoy tocando un solo de guitarra!!! wauuu!!!" << endl;
    }
};

class Bajo : public Instrumento {
public:
    Bajo(string marca, float precio) : Instrumento("Bajo", marca, precio) {}
};

class Bateria : public Instrumento {
public:
    Bateria(string marca, float precio) : Instrumento("Batería", marca, precio) {}
};
```

```cpp
class Persona {
protected:
    string nombre;
    int edad;
    string ocupacion;
public:
    Persona(string nombre, int edad, string ocupacion) {
        this->nombre = nombre;
        this->edad = edad;
        this->ocupacion = ocupacion;
    }

    void presentarse() {
        cout << "Soy " << ocupacion << ". Nombre: " << nombre << ", Edad: " << edad << " años" << endl;
    }
};

class Guitarrista : public Persona {
public:
    Guitarrista(string nombre, int edad) : Persona(nombre, edad, "guitarrista") {}
};

class Bajista : public Persona {
public:
    Bajista(string nombre, int edad) : Persona(nombre, edad, "bajista") {}
};

class Baterista : public Persona {
public:
    Baterista(string nombre, int edad) : Persona(nombre, edad, "baterista") {}
};
```

```cpp
int main() {

    Guitarra *guitarra1 = new Guitarra("Fender", 1200.0);
    Bajo *bajo1 = new Bajo("Squier", 900.0);
    Bateria *bateria1 = new Bateria("Alesis", 1500.0);

    cout << "Inventario de la tienda de instrumentos:" << endl;
    guitarra1->mostrarInfo();
    bajo1->mostrarInfo();
    bateria1->mostrarInfo();

    Guitarrista *persona1 = new Guitarrista("Diego", 25);
    Bajista *persona2 = new Bajista("Vicente", 30);
    Baterista *persona3 = new Baterista("Claudio", 27);

    cout << "Clientes en la tienda:" << endl;
    persona1->presentarse();
    persona2->presentarse();
    persona3->presentarse();

    cout << "El guitarrista toca un solo..." << endl;
    guitarra1->tocarUnSolo();


    return 0;
}
```

Contacto

+569 86031881

benjamin.aceituno@mail.udp.cl

benja.mp4

https://github.com/benjamp4/prog.av-2024-2