

CLASES Y OBJETOS

CONTENIDO

- Programación Orientada a Objetos.

1. CONCEPTOS.
2. VENTAJAS Y DESVENTAJAS.
3. PRINCIPIOS FUNDAMENTALES.
4. CLASES Y OBJETOS.
5. ATRIBUTOS, MÉTODOS MENSAJES.



Gonzalo Quedena


PROGRAMACIÓN ORIENTADA A OBJETOS

¿QUÉ ES?

- ES UN PARADIGMA DE PROGRAMACIÓN EN LA QUE LOS OBJETOS SE UTILIZAN COMO METÁFORA PARA PODER EMULAR LAS ENTIDADES REALES DEL NEGOCIO A MODELAR.
- EN EL PARADIGMA POO SE TRABAJA CON **"OBJETOS"** QUE INTERCAMBIAN INFORMACIÓN ENTRE SÍ, PERO CADA UNO CONSERVANDO UN ESTADO Y UNOS DATOS QUE LES SON PROPIOS Y QUE NO SON VISIBLES PARA OTROS OBJETOS.
- ES UN PARADIGMA DE PROGRAMACIÓN, ES DECIR, UN ESTILO Y UNA FORMA DE PENSAR PARA PODER DAR LA SOLUCIÓN DE DIFERENTES PROBLEMAS, ESTE PARADIGMA MUESTRA APLICACIONES BASADAS EN **OBJETOS** EN LUGAR DE UNA SERIE DE COMANDOS Y DATOS EN LUGAR DE LA LÓGICA.

CARACTERÍSTICAS:

- CADA OBJETO TIENE SU PROPIA NATURALEZA. LA PROPIEDAD O CARACTERÍSTICA DE UN OBJETO DISTINGUE DE OTRO.
- CADA OBJETO EXISTE PARA UN PROPÓSITO. ESTE PROPÓSITO DEFINE LAS ACCIONES, LOS PROCEDIMIENTOS, SERVICIOS O RESPONSABILIDADES QUE UN OBJETO PUEDE PROPORCIONAR.
- LOS SERVICIOS SOLICITADOS A LOS OBJETOS DEPENDE DE LA NATURALEZA DE LOS MISMO, POR EJEMPLO:
 1. NO SE PUEDE CONDUCIR UN FOCO.
 2. NO SE PUEDE HACER VOLAR UN INTERRUPTOR.



Estas acciones o servicios mostrados son inapropiados porque **NO** forman parte del comportamiento natural de los objetos.

POO: VENTAJAS / DESVENTAJAS

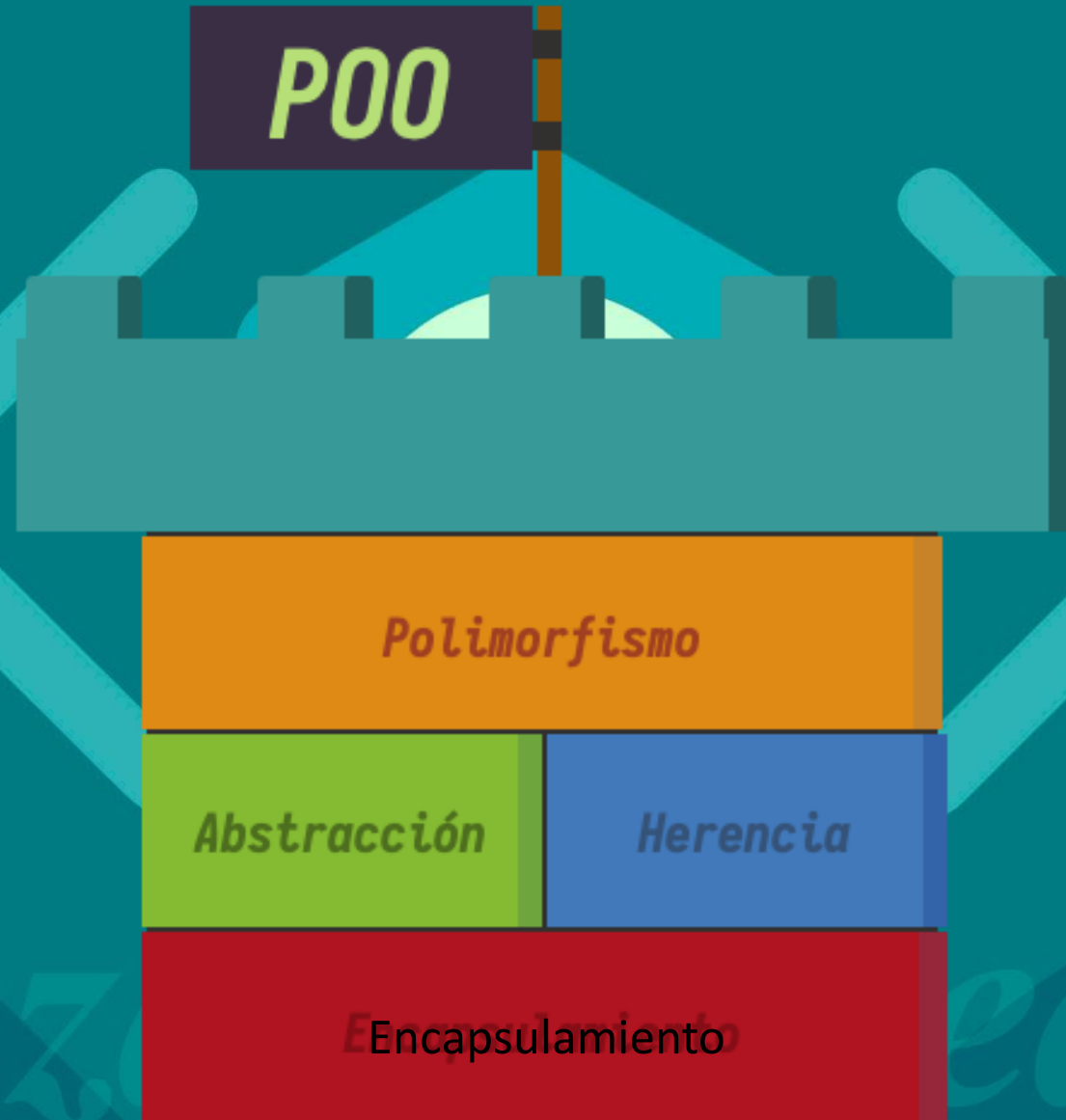
• **VENTAJAS:**

1. REUSABILIDAD.
2. EXTENSIBILIDAD.
3. PORTABILIDAD.
4. FACILIDAD DE MANTENIMIENTO.
5. RAPIDEZ DE DESARROLLO. MÁS FÁCILES DE ENTENDER PORQUE SE USAN ABSTRACCIONES MÁS CERCANAS A LA REALIDAD.

• **DESVENTAJAS:**

1. DIFICULTAD EN LA ABSTRACCIÓN.
2. CURVAS DE APRENDIZAJE LARGAS.

PRINCIPIOS FUNDAMENTALES DE LA POO



PRINCIPIOS FUNDAMENTALES DE LA POO

LA ABSTRACCIÓN CONSISTE EN SELECCIONAR DATOS DE UN CONJUNTO MÁS GRANDE PARA MOSTRAR SOLO LOS DETALLES RELEVANTES DEL OBJETO.

CONSISTE EN REPRESENTAR LAS CARACTERÍSTICAS ESENCIALES DE UN OBJETO, DEJANDO DE LADO LAS NO ESENCIALES.



Abstracción

UNA CLASE ES UNA ABSTRACCIÓN DE UN GRUPO DE OBJETOS, QUE TIENEN LA POSIBILIDAD DE REALIZAR UNA SERIE DE OPERACIONES

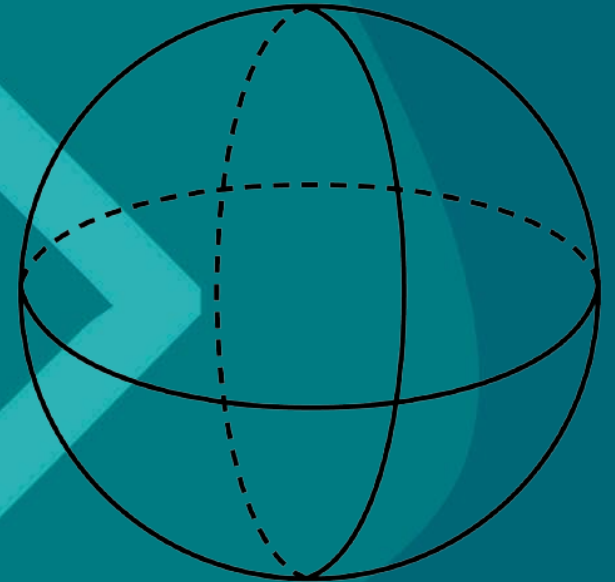
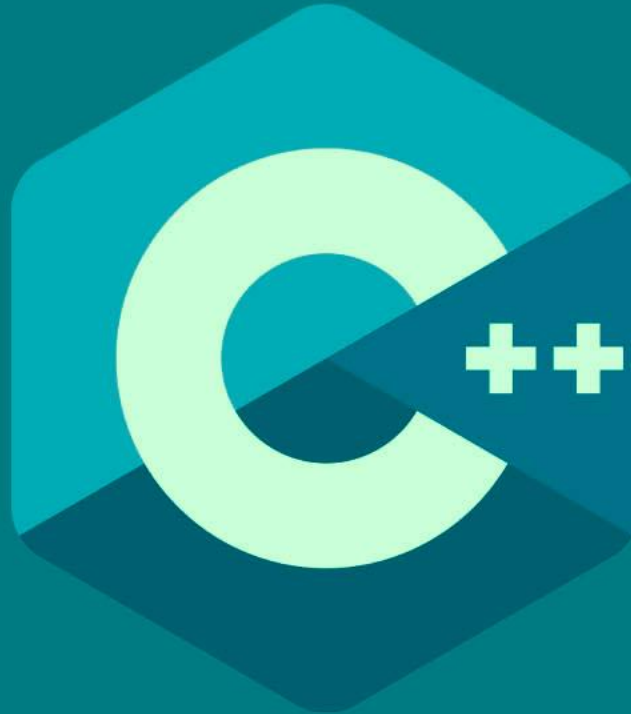
SE ENCARGA DE IDENTIFICAR LAS CARACTERÍSTICAS ESENCIALES DE UN OBJETO PARA CAPTURAR SU COMPORTAMIENTO.

APLICACIÓN: ABSTRACCIÓN

CARACTERIZANDO EL OBJETO

ESFERA:

- ¿UNA ESFERA ES UN OBJETO?
- ¿QUÉ CARACTERIZA A UNA ESFERA?
- ¿QUÉ OPERACIONES O CÁLCULOS SE PUEDE HALLAR CON UNA ESFERA?



Gonzalo Quedena

APLICACIÓN: ABSTRACCIÓN

CARACTERIZANDO EL OBJETO ESFERA:

- ¿QUÉ CONOZCO DE LA ESFERA? ¿ QUÉ CARACTERIZA A UNA ESFERA?

☐ RADIO.

- ¿QUÉ CÁLCULOS SE PUEDE HACER CON LA ESFERA?

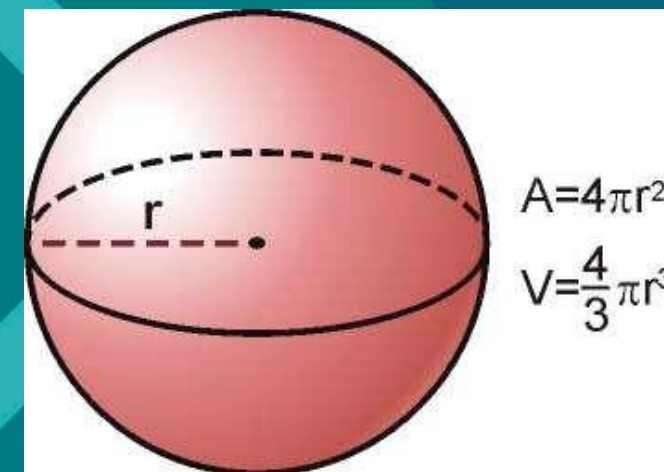
☐ CALCULAR EL ÁREA.

☐ CALCULAR EL VOLUMEN.

- ¿QUÉ OTROS CÁLCULOS U OPERACIONES SE PUEDEN HACER?

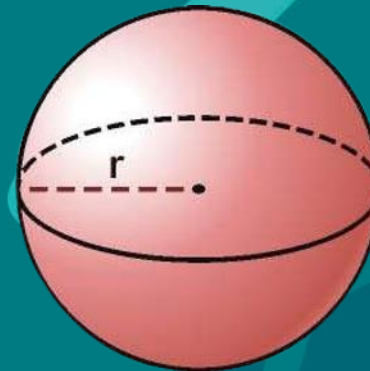
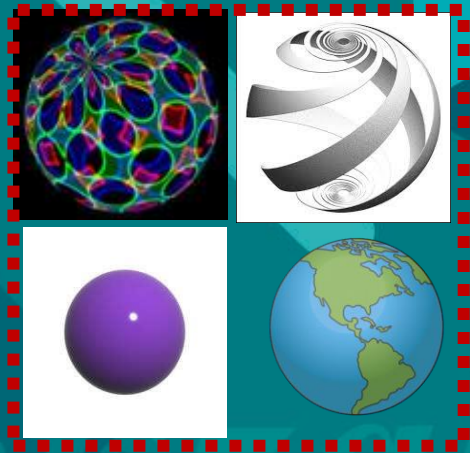
☐ RECONOCER EL VALOR DEL RADIO.

☐ MODIFICAR EL VALOR DEL RADIO.



CLASE & OBJETOS

- ❑ UNA **CLASE** ES LA DESCRIPCIÓN DE UN CONJUNTO DE OBJETOS SIMILARES, CONSTA DE MÉTODOS Y DE DATOS QUE RESUMEN LAS CARACTERÍSTICAS COMUNES DE DICHO CONJUNTO.
- ❑ UNA **CLASE** DENOTA A UNA COLECCIÓN DE OBJETOS DE UN MISMO TIPO.



CLASE & OBJETOS

- ❑ UNA CLASE POSEE ATRIBUTOS Y OPERACIONES. LOS ATRIBUTOS SON VARIABLES QUE TIENE UN TIPO DE DATO ASOCIADO.
- ❑ UNA CLASE SE FORMALIZA/REPRESENTA GRÁFICAMENTE MEDIANTE UN DIAGRAMA QUE REPRESENTA A UNA CLASE.

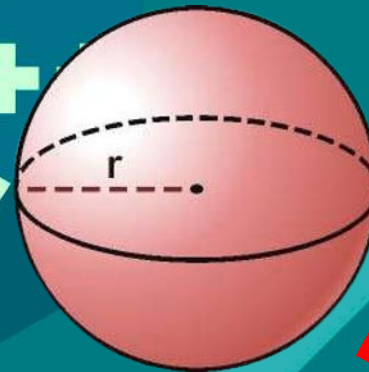
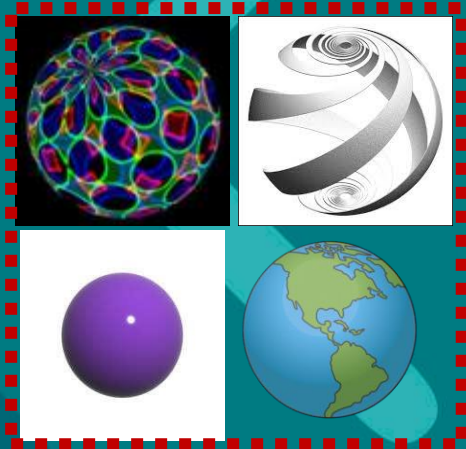


DIAGRAMA DE
LA CLASE ESFERA

Esfera

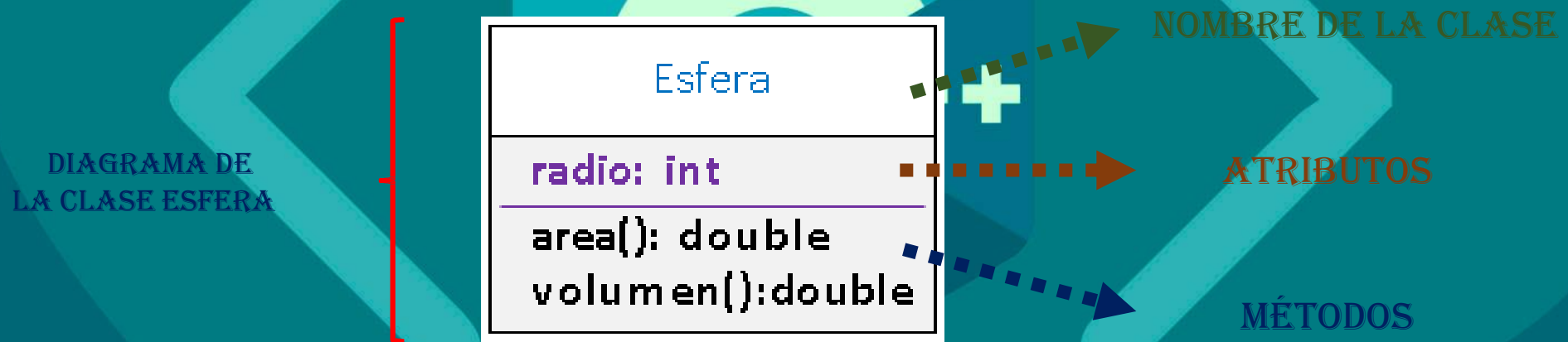
radio: int

area(): double

volumen():double

CLASE & OBJETOS

- ❑ EL DIAGRAMA DE UNA CLASE TIENE TRES BLOQUES QUE DENOTAN A LA CLASE, LAS PROPIEDADES Y MÉTODOS.

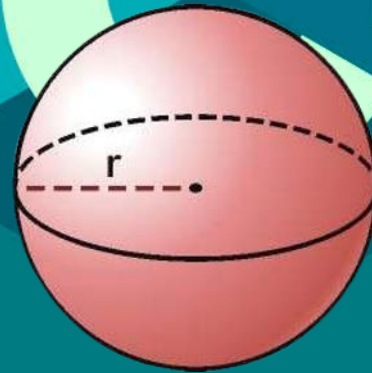
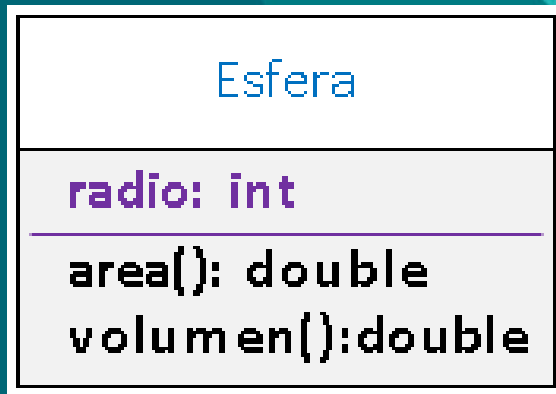


Gonzalo Quedena

CLASE & OBJETOS

- ❑ UNA CLASE UNA VEZ ESTABLECIDA, PUEDE EMPEZAR A CODIFICARSE.
- ❑ EXISTE UNA RELACIÓN DIRECTA ENTRE EL DIAGRAMA DE CLASE Y EL CÓDIGO DE IMPLEMENTACIÓN DE LA CLASE.

DIAGRAMA DE
LA CLASE ESFERA



++

```
class CEsfera {  
  
    //Atributos.  
    int radio;  
    //Métodos.  
    float Area(){  
        /*---*/  
    }  
    float Volumen() {  
        /*---*/  
    }  
};
```

CLASE & OBJETOS

MÁS CARACTERÍSTICAS

- ❑ **TIEMPO DE VIDA:** ESTA DADA POR LA DURACIÓN DE UN OBJETO EN UN PROGRAMA. LOS OBJETOS SON CREADOS MEDIANTE UN MECANISMO DENOMINADO INSTANCIACIÓN, Y CUANDO DEJAN DE EXISTIR SE DICE QUE SON DESTRUIDOS.
- ❑ **ESTADO:** DEFINIDO POR SUS ATRIBUTOS (EL VALOR DE CADA UNO DE SUS ATRIBUTOS).
- ❑ **COMPORTAMIENTO:** LOS MÉTODOS DE LOS OBJETOS PRESENTAN UNA INTERFAZ EN NUESTRO PROGRAMA, ES LA FORMA EN CÓMO INTERACTÚA EL OBJETO

ATRIBUTOS Y ESTADO

- ❑ LOS **ATRIBUTOS** SON LAS CARACTERÍSTICAS QUE EXPONEN TODOS LOS OBJETOS DE UNA CLASE. SU USO ES PARA DESCRIBIR E IDENTIFICAR EL ESTADO.
- ❑ EL **ESTADO** DE UN OBJETO O CLASE VIENE DADO POR LOS VALORES DE SUS ATRIBUTOS EN EL INSTANTE EN EL QUE LO ESTAMOS CONSULTANDO.

| Esfera |
|-------------------------|
| radio: int |
| area(): double |
| volumen():double |

} ATRIBUTOS

MÉTODOS

- ❑ SON FUNCIONES.
- ❑ SON LAS **ACCIONES** QUE DEBEN SER **REALIZADAS POR UN OBJETO** DE UNA CLASE.
- ❑ SON LAS **RESPONSABILIDADES** DEL OBJETO QUE INCLUYEN TANTO EL COMPORTAMIENTO COMO EL ACCESO A LOS **ATRIBUTOS** DE LOS OBJETOS Y CLASES.
- ❑ SON CONTENEDORES DE CÓDIGO EN DONDE HAY ALGORITMOS QUE PROCESAN LOS DATOS O ATRIBUTOS DE UN OBJETO.

| |
|-------------------|
| Esfera |
| radio: int |
| <hr/> |
| area(): double |
| volumen():double |

} MÉTODOS

MÉTODO: CONSTRUCTOR/DESTRUCTOR

❑ CONSTRUCTOR:

*ES UN MÉTODO ESPECIAL DE UNA CLASE, EN DONDE SE APLICA AUTOMÁTICAMENTE A LOS OBJETOS EN EL MOMENTO DE SU CREACIÓN.

PROPÓSITO Y CARACTERÍSTICAS:

*SE UTILIZA PARA INICIALIZAR LOS ATRIBUTOS DE LA CLASE.

*PUEDEN RECIBIR PARÁMETROS, PERO NO PUEDEN RETORNAR NINGÚN VALOR.

*SE PUEDEN SOBRECARGAR, ES DECIR, PODEMOS DEFINIR VARIOS CONSTRUCTORES LOS CUALES SE DIFERENCIAN EN LA CANTIDAD, TIPO Y ORDEN DE LOS PARÁMETROS.

*SE NOMBRAN AL IGUAL QUE LA CLASE.

```
class CEsfera {  
    //Atributos.  
    int radio;  
    //Constructor.  
    CEsfera() {  
        radio = 50;  
    }  
    //Sobrecarga de constructores.  
    CEsfera(int radio) {  
        this->radio = radio;  
    }  
    //Métodos.  
    float Area(){  
        /*---*/  
    }  
    float Volumen() {  
        /*---*/  
    }  
};
```

MÉTODO: CONSTRUCTOR/DESTRUCTOR

❑ **DESTRUCTOR:**

*ES UN MÉTODO ESPECIAL QUE SE INVÓCA CUANDO LA VIDA DE UN OBJETO TERMINA.

*ESTO SE EJECUTA AL FINALIZAR EL PROGRAMA.

++

❑ **PROPÓSITO:**

* LIBERAR LOS RECURSOS QUE EL OBJETO PUDIERA HABER ADQUIRIDO DURANTE SU VIDA.

```
class CEsfera {  
  
    //Atributos.  
    int radio;  
    //Constructor.  
    CEsfera() {  
        radio = 50;  
    }  
    //Sobrecarga de constructores.  
    CEsfera(int radio) {  
        this->radio = radio;  
    }  
    //Destructor.  
    ~CEsfera() {  
    }  
    //Metodos.  
    float Area(){  
        /*---*/  
    }  
    float Volumen() {  
        /*---*/  
    }  
};
```

MÉTODO: SET/GET

❑ SE UTILIZAN PARA:

***GET:** OBTENER EL ESTADO DE UN ATRIBUTO: RECUPERAR EL VALOR DE UN ATRIBUTO.

***SET:** ASIGNAR EL ESTADO A UN OBJETO: MODIFICAR EL VALOR DE ALGÚN ATRIBUTO.

```
class CEsfera {  
  
    //Atributos.  
    int radio;  
    //Constructor.  
    CEsfera() {  
        radio = 50;  
    }  
  
    //Sobrecarga de constructores.  
    CEsfera(int radio) {  
        this->radio = radio;  
    }  
  
    //Destructor.  
    ~CEsfera() {  
    }  
  
    //Métodos.  
    float Area(){  
        /*---*/  
    }  
  
    float Volumen() {  
        /*---*/  
    }  
  
    //Método Set.  
    void setRadio(int radio) {  
        this->radio = radio;  
    }  
  
    //Método Get.  
    int getRadio() {  
        return radio;  
    }  
};
```

CICLO DE VIDA DE LOS OBJETOS

❑ EL PROCESO PARA CREAR OBJETO TIENE DOS ETAPAS:

- * DECLARACIÓN DE LA VARIABLE.
- * CREACIÓN DEL OBJETO FÍSICO.

```
void main(){  
  
    //Declaro mi objeto esfera.  
    CEsfera* objEsfera;  
    //Instancio mi objeto esfera con su constructor.  
    objEsfera = new CEsfera();  
  
    std::cout << "El area de mi esfera vale: " << objEsfera->getRadio() << std::endl;  
  
    getch();  
}
```

```
class CEsfera {  
  
    //Atributos.  
    int radio;  
public:  
    //Constructor.  
    CEsfera() {  
        radio = 50;  
    }  
    //Sobrecarga de constructores.  
    CEsfera(int radio) {  
        this->radio = radio;  
    }  
    //Destructor.  
    ~CEsfera() {  
    }  
    //Métodos.  
    float Area(){  
        /*---*/  
    }  
    float Volumen() {  
        /*---*/  
    }  
    //Método Set.  
    void setRadio(int radio) {  
        this->radio = radio;  
    }  
    //Método Get.  
    int getRadio() {  
        return radio;  
    }  
};
```


MENSAJES

- * UN MENSAJE ES UNA PETICIÓN QUE HACE UN OBJETO PARA SOLICITAR UNA LLAMADA A UNA FUNCIÓN.
- * TODOS LOS OBJETOS DE UNA DETERMINADA CLASE PUEDEN RECIBIR LOS MISMOS MENSAJES.
- * PARA ENVIAR UN MENSAJE AL OBJETO, SE HACE REFERENCIA EL NOMBRE DEL OBJETO Y DEPENDIENDO AL TIPO DE INSTANCIA SE AGREGA EL NOMBRE DEL MÉTODO A EJECUTARSE MEDIANTE UN PUNTO O FLECHITA-



```
void main(){

    /*Forma - 1.*/
    //Declaro mi objeto esfera.
    CEsfera* objEsfera;
    //Instancio mi objeto esfera con su constructor.
    objEsfera = new CEsfera();

    std::cout << "El area de mi esfera vale: " << objEsfera->getRadio() << std::endl;

    /*Forma - 2.*/
    //Declaro mi obeto esfera.
    CEsfera objEsfera2;
    std::cout << "El area de mi esfera vale: " <<objEsfera2.getRadio() << std::endl;

    _getch();
}
```

PRINCIPIOS FUNDAMENTALES DE LA POO

ENCAPSULAMIENTO

REÚNE LOS ELEMENTOS DE UN OBJETO QUE SE CONSIDEREN DE UNA MISMA ENTIDAD, ESTO PERMITE AUMENTAR LA COHESIÓN DE LOS COMPONENTES DEL SISTEMA

CONSISTE EN LA COMBINACIÓN DE DATOS E INSTRUCCIONES EN UN NUEVO TIPO DE DATO, DENOMINADO CLASE.

Encapsulamiento

EL ACCESO A LOS ATRIBUTOS DE LA CLASE SE REALIZA A TRAVÉS DE LOS MÉTODOS DE LA INTERFAZ. (SET/GET)

UNA VEZ CREADA LA CLASE, LAS FUNCIONES USUARIAS NO REQUIEREN CONOCER LOS DETALLES DE SU IMPLEMENTACIÓN.

ENCAPSULAMIENTO

TODOS ELLOS ESTÁN **ENCAPSULADOS** DENTRO DE LA MISMA CLASE, DE MODO QUE SON MIEMBROS DE DICHA CLASE.

ESOS **MÉTODOS Y ATRIBUTOS** PUEDEN SER UTILIZADOS POR OTRAS CLASES SI LA CLASE QUE LOS ENCAPSULA LES BRINDA LOS **PERMISOS** NECESARIOS PARA ACCEDER A ELLA.

FORMAS DE ENCAPSULAR:

ABIERTO (PUBLIC): LOS DATOS PUEDEN SER ACCEDIDOS SIN NINGUNA RESTRICCIÓN.

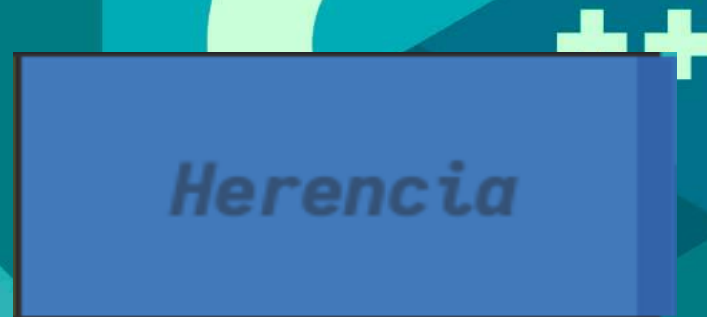
PROTEGIDO (PROTECTED): LOS DATOS SON ACCEDIDOS BAJO CIERTAS RESTRICCIONES.

CERRADO (PRIVATE): NO ES POSIBLE ACCEDER A SUS DATOS.

PRINCIPIOS FUNDAMENTALES DE LA POO

HERENCIA

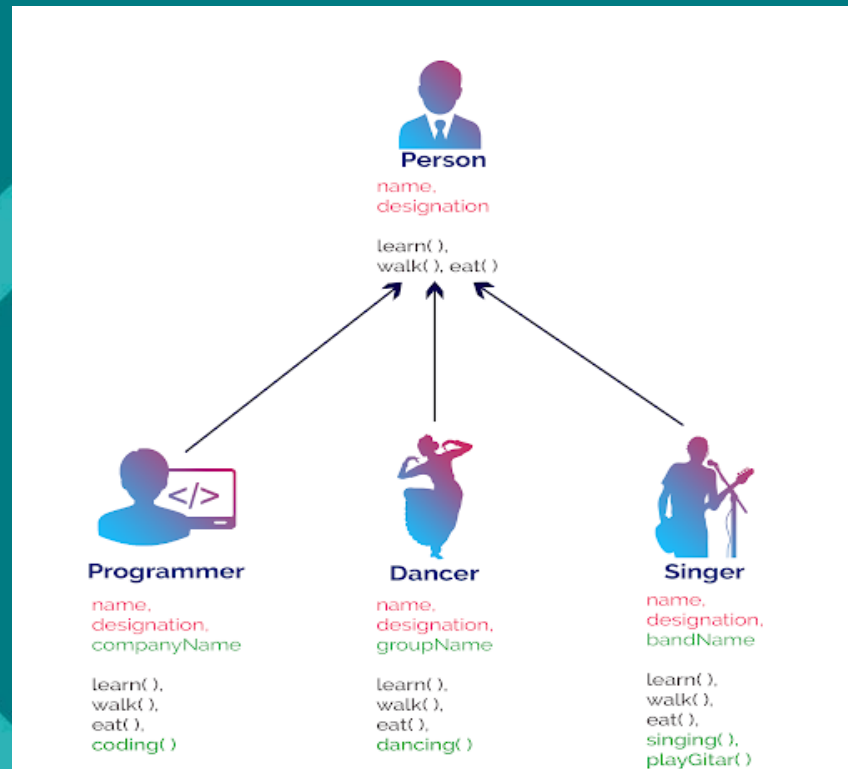
SE ENCARGA DE ORGANIZAR, FACILITAR EL POLIMORFISMO Y ENCAPSULAMIENTO, DE ESTE MODO PERMITE CREAR OBJETOS COMO TIPOS ESPECIALES DE OBJETOS PREDEFINIDOS. ESTOS HEREDAN LOS ATRIBUTOS Y MÉTODOS DE SU CLASE PADRE SIN NECESIDAD DE VOLVER A IMPLEMENTARLOS.



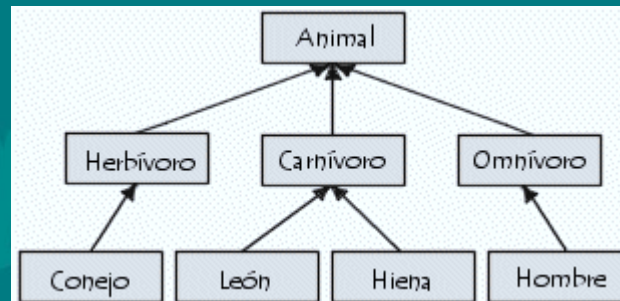
LA HERENCIA DEFINE UNA RELACIÓN ENTRE CLASES DONDE UNA CLASE COMPARTE LA ESTRUCTURA (ATRIBUTOS Y MÉTODOS) EN UNA O MÁS CLASES.

HERENCIA

EJEMPLO - 1:



EJEMPLO - 2:

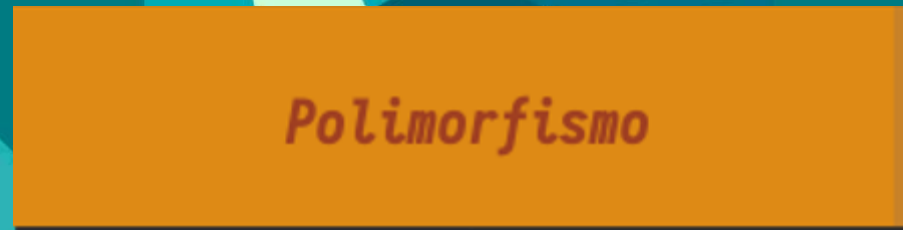


PRINCIPIOS FUNDAMENTALES DE LA POO

POLIMORFISMO

EL **POLIMORFISMO** PERMITE QUE EL MISMO MÉTODO EJECUTE DIFERENTES COMPORTAMIENTOS DE DOS FORMAS:

- ☐ ANULACIÓN DE MÉTODO.
- ☐ SOBRECARGA DE MÉTODO.



TIENEN LOS MISMOS MENSAJES, PERO SU EJECUCIÓN ES DISTINTA.

Gonzalo Quedena

POLIMORFISMO

EJEMPLO:

