Temas Puntuales:

Herencia, Tipos , Alcances.

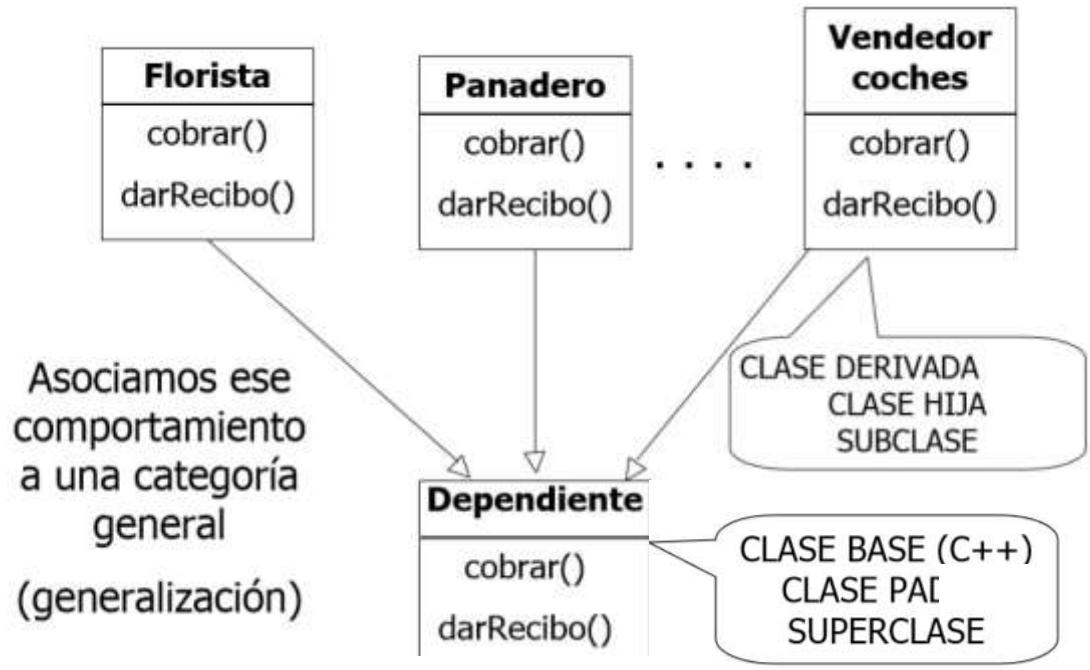
La Herencia

Es uno de los 4 pilares de la programación orientada a objetos (POO) junto con la Abstracción , Encapsulación y Polimorfismo.

Respecto a la herencia se han dado muchas definiciones como por ejemplo la siguiente: "La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos. " o “La Herencia es el mecanismo de implementación mediante el cual elementos más específicos incorporan la estructura y comportamiento de elementos más generales”…….

Entender el mecanismo de abstracción de la herencia.

- Distinguir entre los diferentes tipos de herencia
- Saber discernir entre jerarquías de herencia seguras (bien definidas) e inseguras.
- Reutilización de código: Ser capaz de decidir cuándo usar herencia y cuándo optar por composición.



La herencia consigue clasificar los tipos de datos (abstracciones) por variedad, acercando un poco más el mundo de la programación al modo de razonar humano.

Gracias a la herencia es posible especializar o extender la funcionalidad de una clase, derivando de ella nuevas clases.

La herencia es siempre transitiva: una clase puede heredar características de superclases que se encuentran muchos niveles más arriba en la jerarquía de herencia.

- **Ejemplo:** si la clase Perro es una subclase de la clase Mamífero, y la clase Mamífero es una subclase de la clase Animal, entonces el Perro heredará atributos tanto de Mamífero como de Animal.

HERENCIA "ES-UN"

- La clase A se debe relacionar mediante herencia con la clase B si "A ES-UN B". Si la frase suena bien, entonces la situación de herencia es la más probable para ese caso
- Un pájaro es un animal
- Un gato es un mamífero
- Un pastel de manzana es un pastel
- Una matriz de enteros es un matriz
- Un coche es un vehículo

"ES-UN"

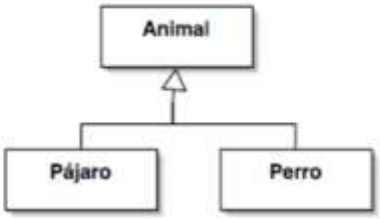
- **Sin embargo, si la frase suena rara por una razón u otra, es muy probable que la relación de herencia no sea lo más adecuado. Veamos unos ejemplos:**
- Un pájaro es un mamífero
- Un pastel de manzana es una manzana
- Una matriz de enteros es un entero

- Un motor es un vehículo
-

Los tipos de Herencia pueden ser

- Simple/Múltiple
- De implementación/de interfaz

- Simple/Múltiple
  - **Simple**: única clase base



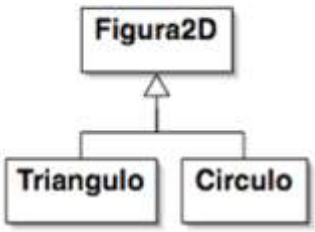
  - **Múltiple**: Más de una clase base

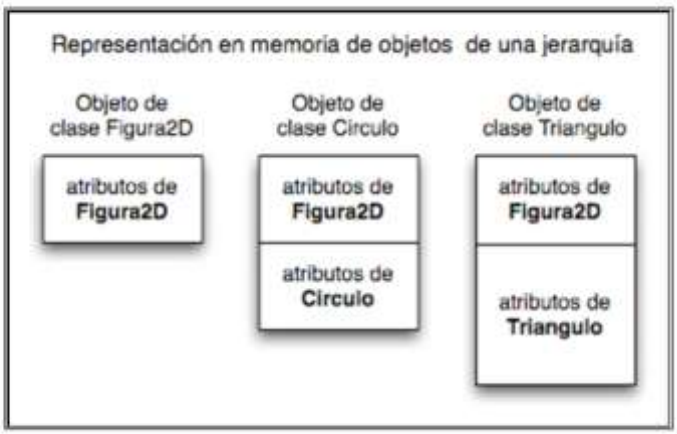
De implementación/de interfaz

De implementación: La implementación de los métodos es heredada. Puede sobreescribirse en las clases derivadas.

De interfaz: Sólo se hereda la interfaz, no hay implementación a nivel de clase base (interfaces en Java, clases abstractas)

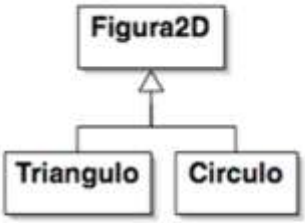
En la herencia las propiedades definidas en una clase base son heredadas por la clase derivada y la clase derivada puede añadir propiedades específicas (atributos, métodos) que la superclase no tiene.






```
class Figura2D {
  public:
    ...
    void setColor(Color c);
    Color getColor() const;
  private:
    Color colorRelleno;
... };

class Circulo : Figura2D {
...
  public:
    void vaciarCirculo() {
      colorRelleno=NINGUNO;
      // ¡ERROR! colorRelleno es privado
      setColor(NINGUNO); // OK
    }
};
```




```
int main() {
  Circulo c;
  c.setColor(AZUL);
  c.getColor();
  c.vaciarCirculo();
  ...
}
```

La parte privada de una clase base no es directamente accesible desde la clase derivada.

Visibilidad de los Métodos y Atributos.

Ámbito de visibilidad **protected**

Los datos/funciones miembros **protected** son directamente accesibles desde la propia clase y sus clases derivadas. Tienen visibilidad privada para el resto de ámbitos.

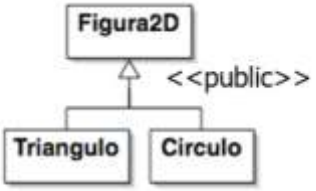
```
class Figura2D {
    protected:
      Color colorRelleno;
...
};

class Circulo : Figura2D {
  public:
    void vaciarCirculo() {
      colorRelleno=NINGUNO; //OK, protected
    }
...
};
```

```
int main () {
  Circulo c;
  c.colorRelleno=NINGUNO;
 // ¡ERROR! colorRelleno
 // es privado aqui
}
```

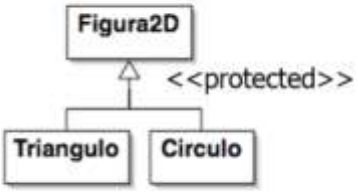
- Herencia Pública (por defecto)

```
class Circulo : public Figura2D {
...
};
```



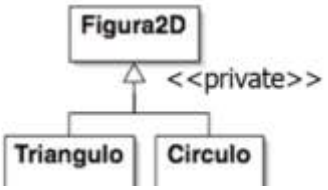
- Herencia Protegida

```
class Circulo : protected Figura2D {
...
};
```

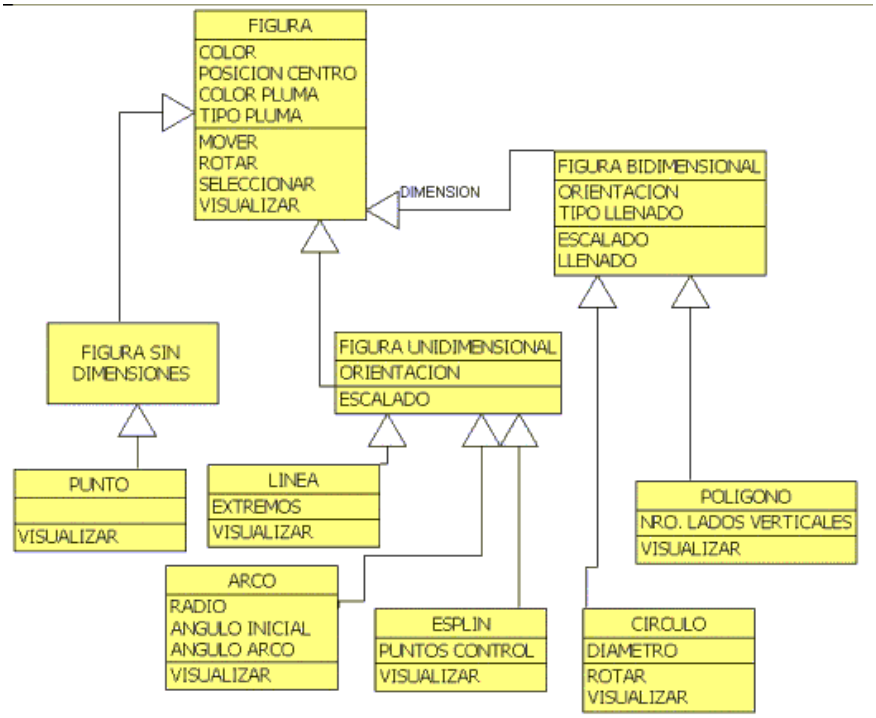


- Herencia Privada

```
class Circulo : private Figura2D {
...
};
```



Otro ejemplo de Herencia



Algo más sobre Herencia ……

Los constructores (que se usan para crear clases) no son heredados por las subclases.

Para crear una subclase, se incluye la palabra clave **extends** en la declaración de la clase.

class nombreSubclase **extends** nombreSuperclase{

}

En Java, la clase padre de todas las clases es la clase **Object** y cuando una clase **no tiene** una superclase explícita, su superclase es Object también.