Generalizaciones y Herencia

Programación Orientada a Objetos

Definición

+ Las clases con atributos y operaciones comunes se pueden organizar de forma jerárquica, mediante la herencia. La herencia es una abstracción importante para compartir similitudes entre clases, donde todos los atributos y operaciones comunes a varias clases se pueden compartir por medio de la superclase, una clase más general. Las clases más refinadas se conocen como las subclases.

Ejemplo

+ Las Impresoras Láser, de Burbuja, y de Matriz, son todas subclases de la superclase Impresora. Los atributos generales de una Impresora son el Modelo, Velocidad, y Resolución, mientras que sus operaciones son Imprimir y Alimentar.
+ Herencia es una relación "es-una" entre las clases las más refinadas y más generales.
+ Ejemplo: Impresora Láser es una Impresora.

Concepto de superclase

+ La superclase generaliza a sus subclases, y las subclases especializan a la superclase. El proceso de especialización es el inverso de generalización. Una instancia de una subclase, o sea un objeto, es también una instancia de su superclase.

Ejemplo

+ Cuando se crea un objeto de tipo Impresora Láser, este objeto incluye toda la información descrita en la subclase Impresora Láser, al igual que en la superclase Impresora; por lo tanto se considera que el objeto es una instancia de ambas.

Jerarquía

+ La herencia es transitiva a través de un número arbitrario de niveles. Los ancestros de una clase son las superclases de una clase en cualquier nivel superior de la jerarquía, y los descendientes de una clase son las subclases de una clase en cualquier nivel inferior de la jerarquía.

Ejemplo

+ Si además de Impresora de Burbuja, se define una clase más especializada como Impresora de Burbuja Portátil, entonces Impresora e Impresora de Burbuja son ancestros de la clase Impresora de Burbuja Portátil, mientras que Impresora de Burbuja e Impresora de Burbuja Portátil son descendientes de Impresora.

Características de jerarquía de herencia

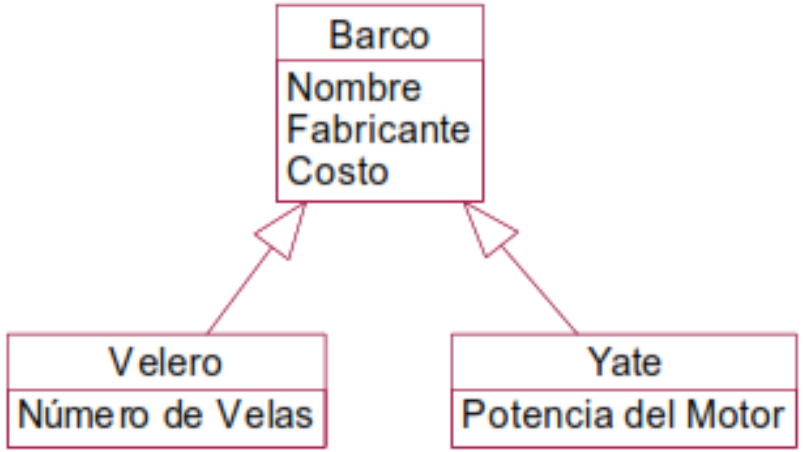
- Las siguientes características se aplican a clases en una jerarquía de herencia:
  - Los valores de una instancia incluyen valores para cada atributo de cada clase ancestra.
  - Cualquier operación de cualquier clase ancestra, se puede aplicar a una instancia.
  - Cada subclase no solo hereda todas las características de sus ancestros sino también añade sus propios atributos y operaciones.

Generalización de herencia

- Los nombres de atributos y operaciones deben ser únicos en la jerarquía de herencia.
  - Ejemplo: Una Impresora de Burbuja Portátil incorpora todas las características, primero de una Impresora, y luego de una Impresora de Burbuja, conteniendo valores para todos los atributos ancestrales y pudiéndose aplicar todas las operaciones ancestrales.
- La generalización se puede extender a múltiples niveles de jerarquías, donde una clase hereda de su superclase, que a su vez hereda de otra superclase, hacia arriba en la jerarquía. En otras palabras, las relaciones entre subclases y superclases son relativas. La herencia define una jerarquía de clases donde existen ancestros y descendientes, que pueden ser directos o no.

Ejemplo

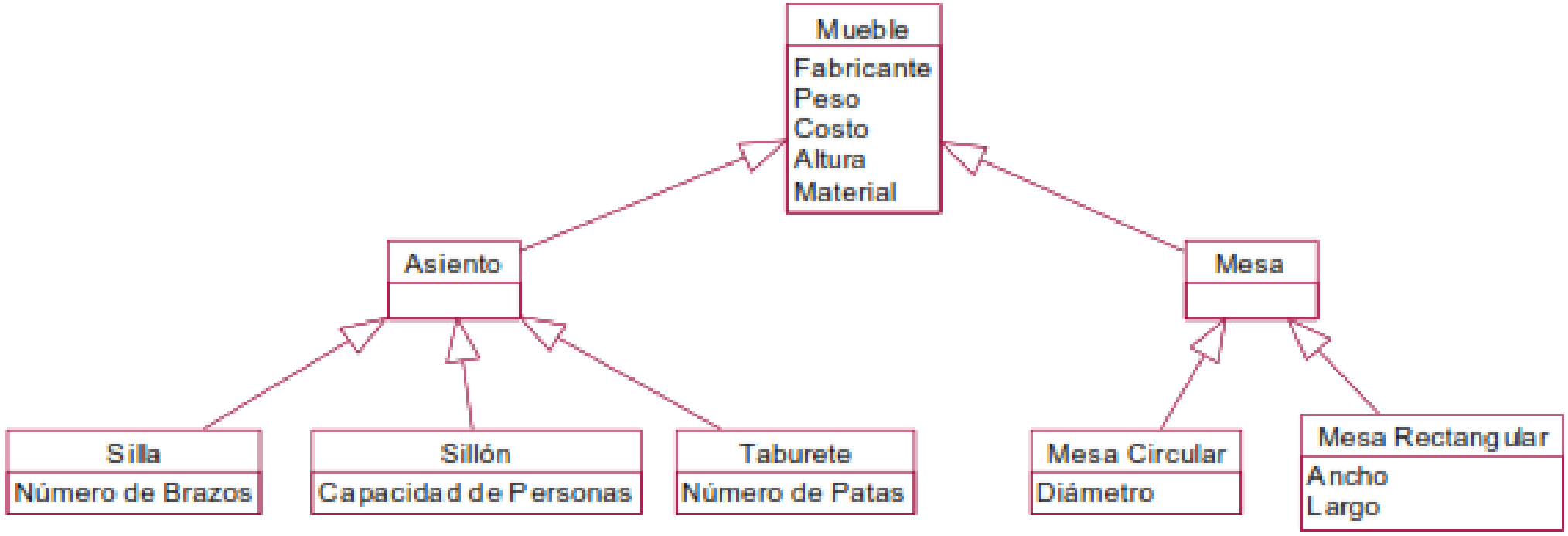
- Un Barco tiene un Nombre, Fabricante, y Costo. Tipos especiales de Barco, como Velero, tienen además de estas características básicas, un Número de Velas, mientras que otro tipo especial de Barco, como Barco a Motor, tiene un Motor.



Ejercicio 1

+ Una jerarquía conteniendo una superclase Mueble, y varias subclases Mesa y Asiento, puede ser extendida con nuevas subclases, como Mesa Circular, Mesa Rectangular, mientras que un Asiento puede extenderse con las subclases Silla, Sillón, y Taburete. Cada clase tiene sus propios atributos los cuales se van especializando a medida que las clases son cada vez más especializadas.

Solución


Mueble
Fabricante
Peso
Costo
Altura
Material
Asiento
Mesa
Silla
Número de Brazos
Sillón
Capacidad de Personas
Taburete
Número de Patas
Mesa Circular
Diámetro
Mesa Rectangular
Ancho
Largo

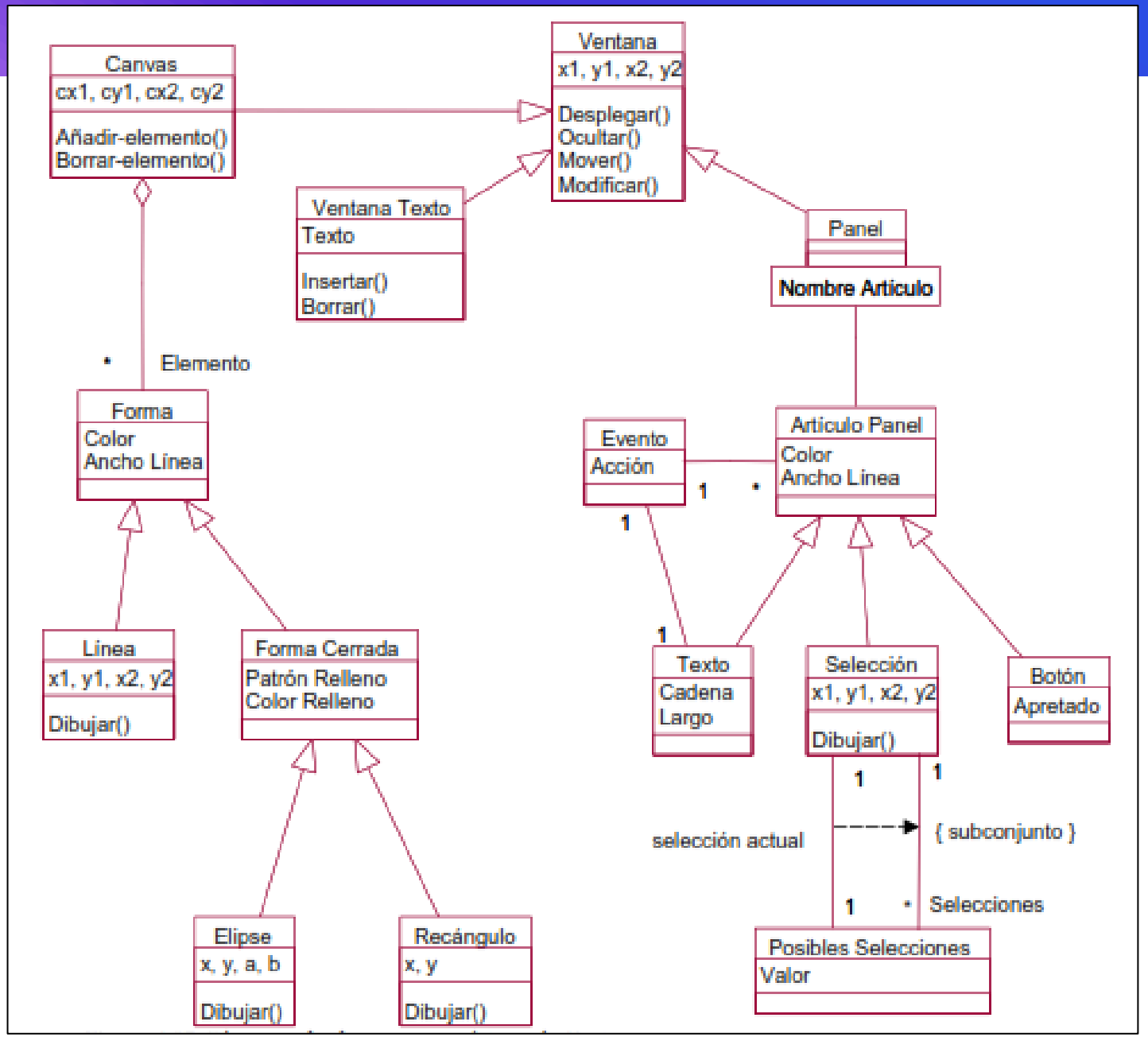
Ejercicio 2

Una clase Ventana tiene atributos para los vértices de la ventana y operaciones para Desplegar, Ocultar, Mover y Modificar la ventana. Canvas, Panel, y Ventana de Texto son tipos diferentes de Ventanas.

Un Canvas se utiliza para diferentes despliegues gráficos, incluyendo atributos como el tamaño del elemento gráfico y operaciones para Añadir y Borrar tales elementos. El Canvas se relaciona con varios Elementos (Formas) que son Líneas o Formas Cerradas, como Elipses o Polígonos. Un Polígono consiste de una lista ordenada de Puntos.

Un Panel contiene diferentes Artículos de Panel, los cuales pueden ser de tipo Botón, Selección, o Texto. Todos los Artículos de Panel están relacionados con Eventos del ratón, y el artículo de tipo Texto se relaciona además con un Evento del teclado. Cuando un Artículo de Panel se escoge, un Evento se genera.

Solución

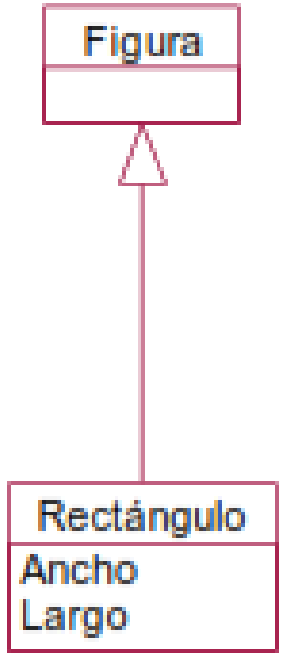

Canvas
cx1, cy1, cx2, cy2
Añadir-elemento()
Borrar-elemento()
Ventana
x1, y1, x2, y2
Desplegar()
Ocultar()
Mover()
Modificar()
Ventana Texto
Texto
Insertar()
Borrar()
Panel
Nombre Articulo
*
Elemento
Forma
Color
Ancho Línea
Evento
Acción
1
*
1
Articulo Panel
Color
Ancho Línea
Línea
x1, y1, x2, y2
Dibujar()
Forma Cerrada
Patrón Relleno
Color Relleno
1
Texto
Cadena
Largo
Selección
x1, y1, x2, y2
Dibujar()
Botón
Apretado
1
1
selección actual
{ subconjunto }
1
*
Selecciones
Posibles Selecciones
Valor
Elipse
x, y, a, b
Dibujar()
Recángulo
x, y
Dibujar()

Razones de la generalización

- La generalización puede hacerse por diferentes razones:
- **Extensión**: clases definidas por operaciones con estructuras de información.
- **Restricción**: clases como implementaciones de tipos, especializaciones, subconjunto de todas las instancias de un ancestro.

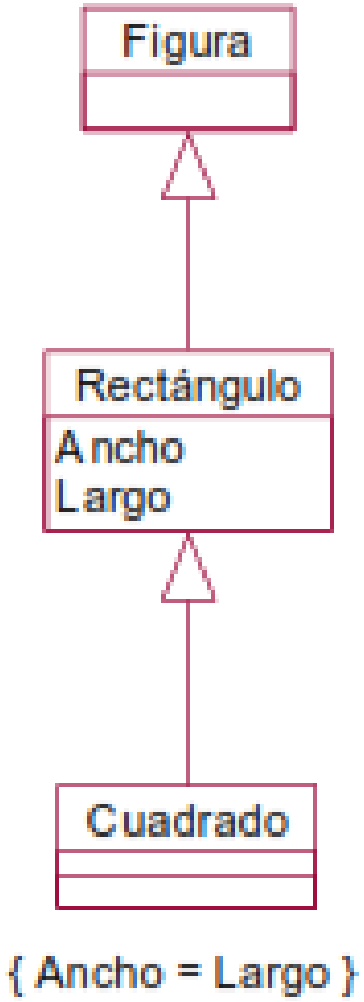
Extensión de Clase

- La extensión de clase expresa que una subclase puede añadir nuevos atributos u operaciones a la superclase.
- Ejemplo: La clase Rectángulo añade los nuevos atributos, Ancho y Largo.



Restricción de Clase

- La restricción de clase indica como una subclase puede restringir los atributos heredados de una superclase. Una clase descendiente no puede suprimir los atributos u operaciones de sus ancestros.
- Ejemplo: La clase Cuadrado restringe los atributos de Rectángulo, ya que Ancho debe ser igual a Largo.



Sobrescritura de Operaciones

- La sobrescritura de operaciones, también conocida como "override" en el contexto de la programación orientada a objetos (POO), se refiere a la capacidad de una clase hija o subclase de proporcionar una implementación diferente de un método o función que ya está definido en su clase padre o superclase. En otras palabras, la sobrescritura permite a una clase hija proporcionar su propia implementación de un método que tiene el mismo nombre y firma en su clase padre. Sin embargo, la sobrescritura es parte fundamental de la orientación a objetos, en particular del polimorfismo.

Reglas de la sobre escritura

+ La sobrescritura de operaciones es un concepto importante en la POO y permite la especialización y personalización del comportamiento de un método en una clase hija, adaptándolo a las necesidades específicas de esa clase en particular. Para sobrescribir un método en una clase hija, se debe seguir ciertas reglas:

1. El método en la clase hija debe tener el mismo nombre que el método en la clase padre que se desea sobrescribir.
2. El método en la clase hija debe tener la misma firma que el método en la clase padre, lo que significa que debe tener el mismo nombre, el mismo tipo de retorno y la misma lista de parámetros.
3. La visibilidad del método en la clase hija no puede ser más restrictiva que la visibilidad del método en la clase padre. Por ejemplo, si el método en la clase padre es público, entonces el método en la clase hija también debe ser público o protegido, pero no privado.
4. La anotación @Override se puede usar opcionalmente en el método de la clase hija para indicar que se está sobrescribiendo un método de la clase padre. Esta anotación es útil para evitar errores de sintaxis y ayuda a mantener un código más limpio y legible.

Ejemplo practico

- En este ejemplo, se define una clase padre Figura con un método area() que es sobrescrito en las clases hijas Cuadrado y Circulo con implementaciones específicas para calcular el área de cada figura geométrica. Luego, se crean instancias de las clases hijas Cuadrado y Circulo y se llama al método area() en cada una de ellas para calcular y mostrar el área correspondiente de la figura.

```java
class Figura {
    public double area() {
        return 0;
    }
}
class Cuadrado extends Figura {
    private double lado;

    public Cuadrado(double lado) {
        this.lado = lado;
    }
    @Override
    public double area() {
        return lado * lado;
    }
}
class Circulo extends Figura {
    private double radio;

    public Circulo(double radio) {
        this.radio = radio;
    }

    @Override
    public double area() {
        return 3.14159 * radio * radio;
    }
}
public class Principal {
    public static void main(String[] args) {
        Cuadrado cuadrado = new Cuadrado(5);
        Circulo circulo = new Circulo(3);

        // Calcular y mostrar el área de cada figura
        System.out.println("Área del cuadrado: " + cuadrado.area()); // Salida: Área del cuadrado: 25.0
        System.out.println("Área del círculo: " + circulo.area()); // Salida: Área del círculo: 28.27431
    }
}
```