HERENCIA

Alumno:

Gómez Faustino Paola | 01296824

Profesor:

M.C. Claudia Tona

Grupo:

544

Materia:

Programación orientada a objetos

Definición de herencia.

a. La herencia es uno de los 4 pilares de la programación orientada a objetos, es el mecanismo por el cual una clase permite heredar las características (Clases y atributos) de otra clase. La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando una jerarquía de clases una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados.

b. Cómo se utiliza la herencia en Java.

En java, la herencia se logra mediante la palabra “extends”, una clase hija o subclase puede heredar los atributos y métodos de una clase padre o superclase. La herencia permite establecer una relación "Es-A" entre dos clases, donde la subclase hereda las características de la superclase.

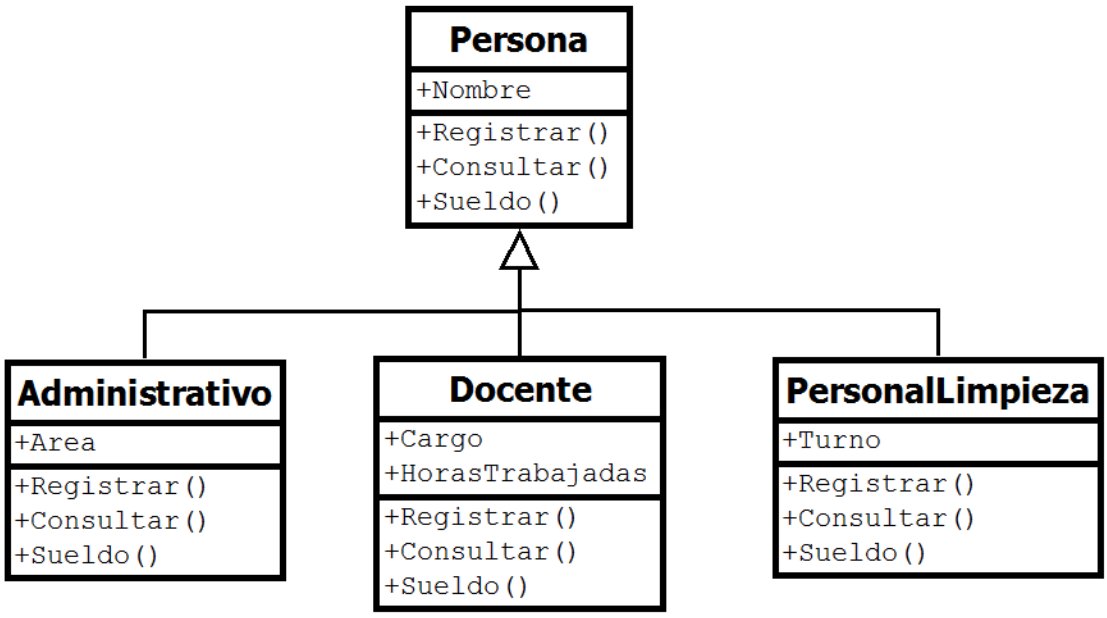
c. Para qué se utiliza y cuáles son sus ventajas.

La herencia se usa principalmente para aprovechar las características comunes entre clases relacionadas, evitando la duplicación de código. Permite definir comportamientos generales en una superclase que pueden ser compartidos y específicos para varias subclases, mejorando la organización y la claridad del código. También facilita la extensión de funcionalidades sin modificar la superclase directamente, lo que contribuye a un diseño más flexible y mantenible.

Ventajas de la herencia

1. Reutilización de código: Al heredar de una superclase, las subclases pueden reutilizar métodos y atributos, lo que reduce la redundancia.
2. Facilidad de mantenimiento: Los cambios realizados en la superclase afectan automáticamente a las subclases, centralizando las actualizaciones de código.
3. Organización y estructura: Permite crear una jerarquía lógica de clases, haciendo el código más intuitivo y fácil de entender.
4. Extensibilidad: Se pueden crear nuevas subclases que hereden características de una superclase, permitiendo agregar funcionalidades específicas sin modificar la clase base.
5. Polimorfismo: Facilita el uso del polimorfismo, lo cual permite tratar objetos de distintas subclases como si fueran de la superclase, incrementando la flexibilidad en el diseño de aplicaciones.

d. Cómo se representa la herencia utilizando diagramas de clases UML.



e. Elaboraré tres ejemplos de herencia, definiendo las clases, sus atributos, y métodos. Los ejemplos deben estar acompañados de su respectivo diagrama UML que muestre las relaciones entre las clases.

1. Superclase: Vehiculo

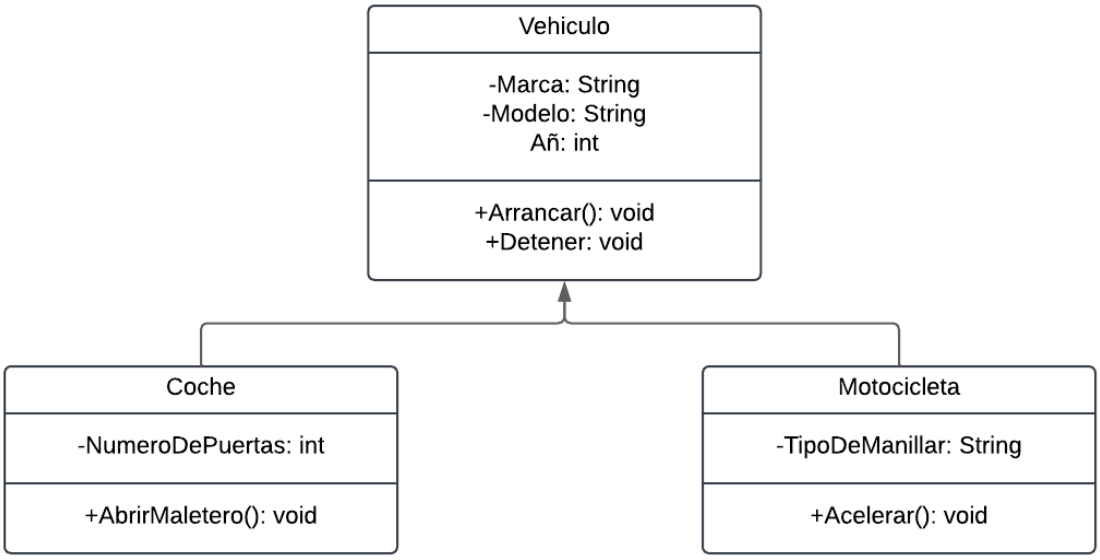
- **Atributos:** Marca (Cadena), modelo(Cadena), año(int)
- **Métodos:** Arrancar() ,detener()

2. Subclase: Coche

- **Atributos:** NumeroDePuertas (int)
- **Métodos:** AbrirMaletero()

3. Subclase: Motocicleta

- **Atributos:** TipoDeManillar (Cadena)
- **Métodos:** HacerCaballito()



Ejemplo 2:

1. Superclase:DispositivoElectronico

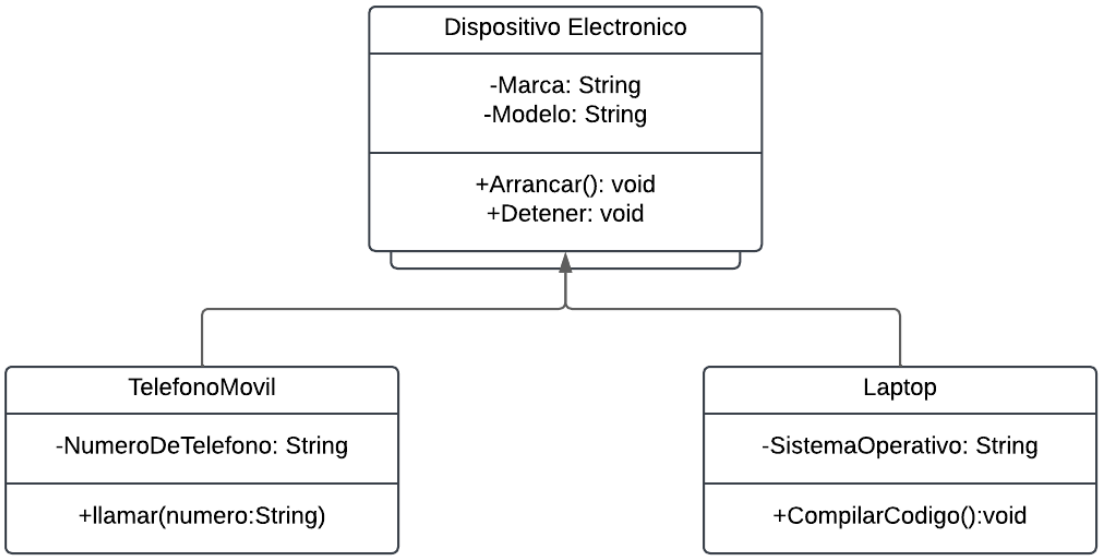
- **Atributos:** marca (Cadena), modelo(Cadena)
- **Métodos:** encender() ,apagar()

2. Subclase:TelefonoMovil

- **Atributos:** numeroDeTelefono (Cadena)
- **Métodos:** llamar(numero: String)

3. Subclase:Laptop

- **Atributos:** sistemaOperativo (Cadena)
- **Métodos:** compilarCodigo()



Ejemplo: Animales y Tipos Específicos

1. **Superclase:** Animal
    - **Atributos:** nombre (Cadena), edad(int)
    - **Métodos:** comer() ,dormir()
2. **Subclase:** Perro
    - **Atributos:** raza (Cadena)
    - **Métodos:** ladrar()
3. **Subclase:** Gato
    - **Atributos:** color (Cadena)
    - **Métodos:** maullar()