Interacción entre Objetos - Ejemplo

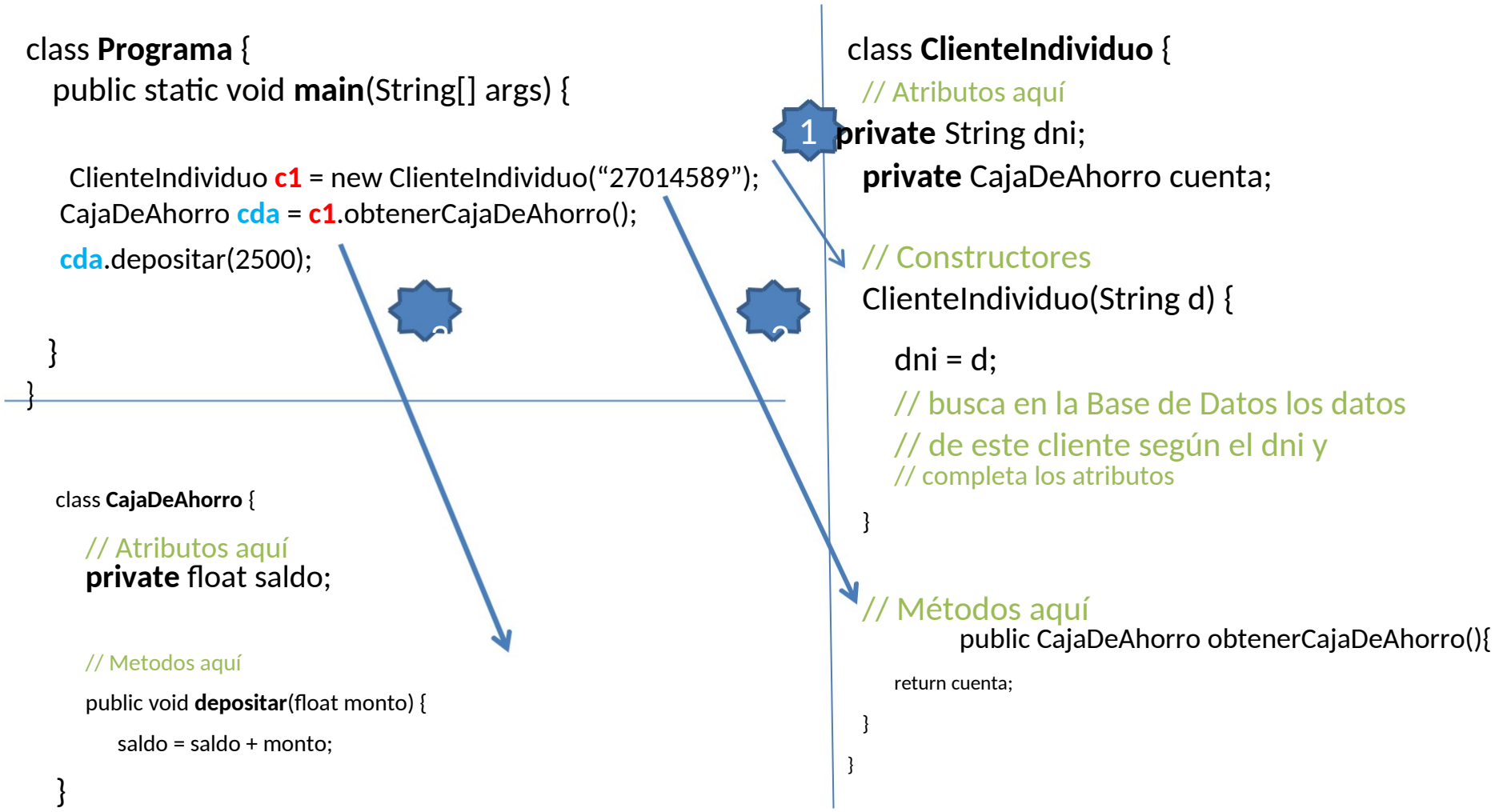
Asumiendo que un cliente individual utiliza el cajero automático para realizar un deposito de $2500, ingresa su dni numero 27.014.589 como pin y realiza el deposito.

La interacción básica sería:

```
// Obtiene el cliente
ClienteIndividuo c = ClienteIndividuo(“27014589”);

/   Obtiene la caja de ahorro del cliente
CajaDeAhorro cda = c.obtenerCajaDeAhorro();

/   Realiza el deposito
cda.depositar(2500);
```

El código completo correspondiente se presenta a continuación:

Interacción entre Objetos - Ejemplo



Ejercicio #7 – Uso de Objetos

Suponiendo que un cliente corporativo desea realizar una extracción de una cuenta corriente, teniendo en cuenta los siguientes datos:

- ✓ el cuit de la empresa es 30-12345678-1
- ✓ desea extraer $20.000
- ✓ en la cuenta hay disponible $18.000
- ✓ el giro en descubierto es de $5.000

Realizar el código en forma genérica utilizando las clases Programa, ClienteCorporativo y CuentaCorriente

TIP: el giro en descubierto es el monto máximo que el banco le presta al cliente en caso de que el saldo en su cuenta sea cero. Es decir que si el cliente no tiene fondos en la cuenta, y desea realizar una extracción o un pago, puede utilizar automáticamente hasta $5.000

Ejercicio #7 – Codificación

Código de la clase Programa:

```
class Programa {

    public static void main(String[] args) {

            // Obtiene el cliente de acuerdo a su cuit
            ClienteCorporativo cliente = new ClienteCorporativo("30-12345678-1");

            /   Obtiene la cuenta corriente del cliente
            CuentaCorriente cc = cliente.obtenerCuentaCorriente();

            /   Realiza la extraccion
            cc.extraer(20000);

    }

}
```

Ejercicio #7 – Codificación

Código de la clase ClienteCorporativo:

```
class ClienteCorporativo {

        // Atributos aquí
        private String cuit;
        private CuentaCorriente cuenta;

         /   Constructores
        ClienteCorporativo(String c) {
            cuit = c;
           /   busca en la Base de Datos los datos
           /   de este cliente según el cuit y
           /   completa los atributos
         }

         // Métodos aquí
         public CuentaCorriente obtenerCuentaCorriente(){
            return cuenta;
         }
}
```

Ejercicio #7 – Codificación

```
class CuentaCorriente {

        private float saldo;
        private float saldoDescubierto;

        public void extraer(float monto) {
            if ( saldo >= monto ) {
                saldo = saldo - monto;
                print("Extraccion ok");
            }
            else {
                montoExcedente = monto - saldo;
                if ( montoExcedente <= saldoDescubierto) {
                        saldo = 0;
                        saldoDescubierto = saldoDescubierto - montoExcedente;
                        print("Extraccion ok");
                }
                else {
                        print("No hay fondos suficientes");
                }
             }
        }
}
```

Mecanismo de Herencia

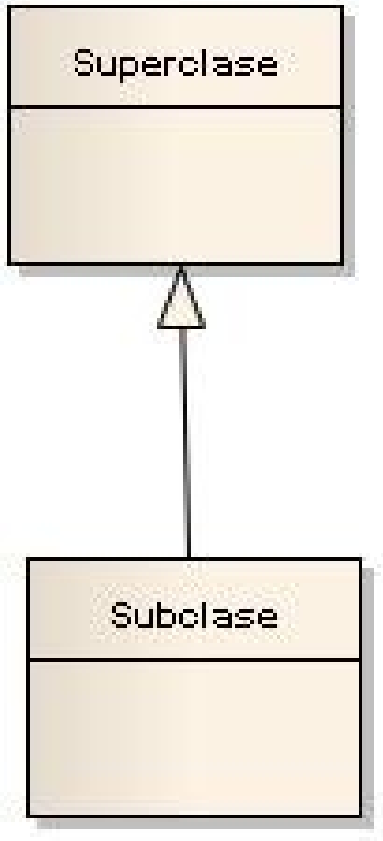
Mecanismo que permite que una clase "herede de otra clase" o "extienda otra clase“, ***recibiendo o heredando atributos y operaciones*** de su clase "padre".

La **clase principal** se denomina: superclase o clase padre

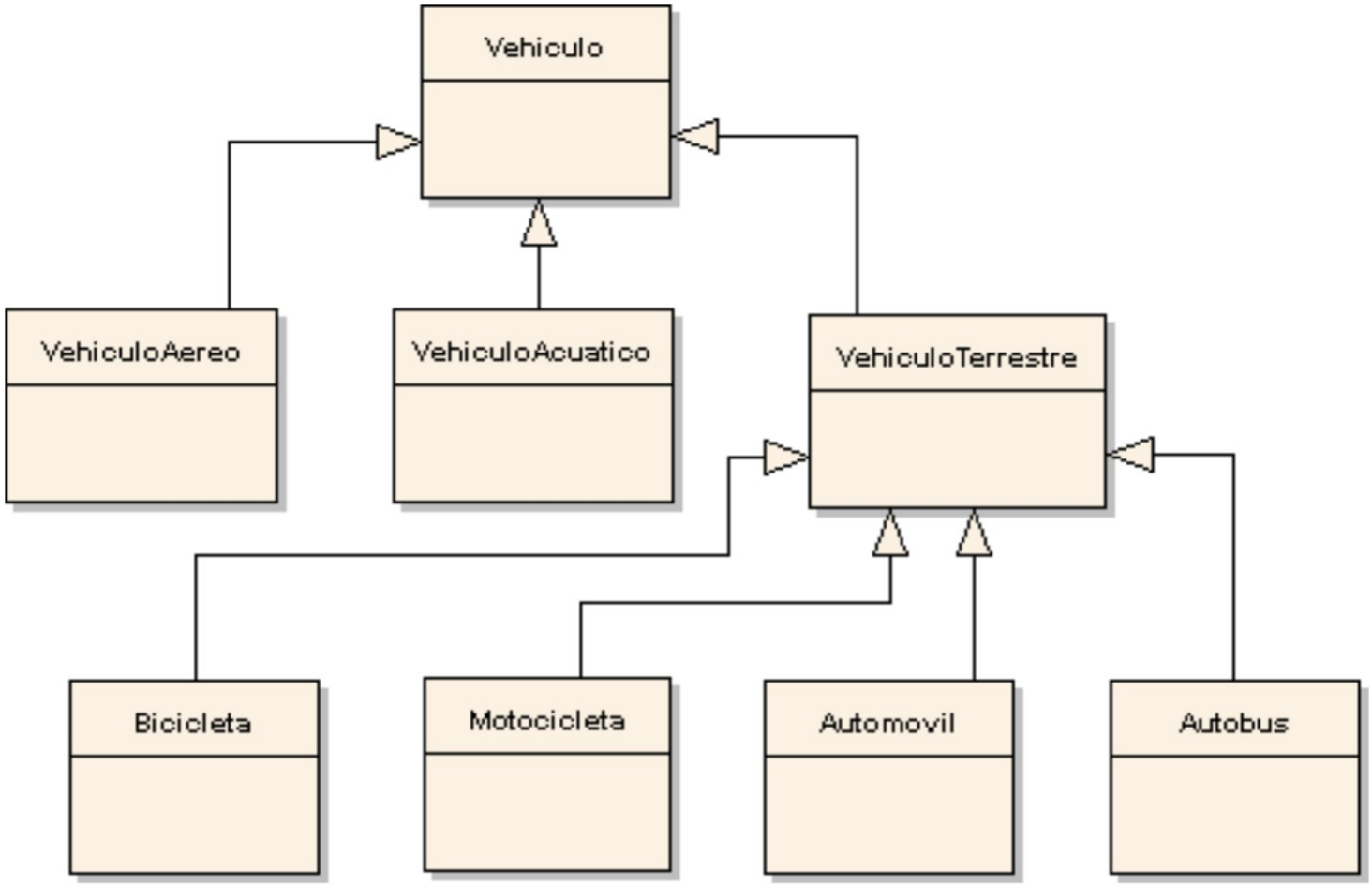
La **clase que hereda** se denomina: subclase o clase hija o clase derivada

La relación se interpreta como “ES UN”

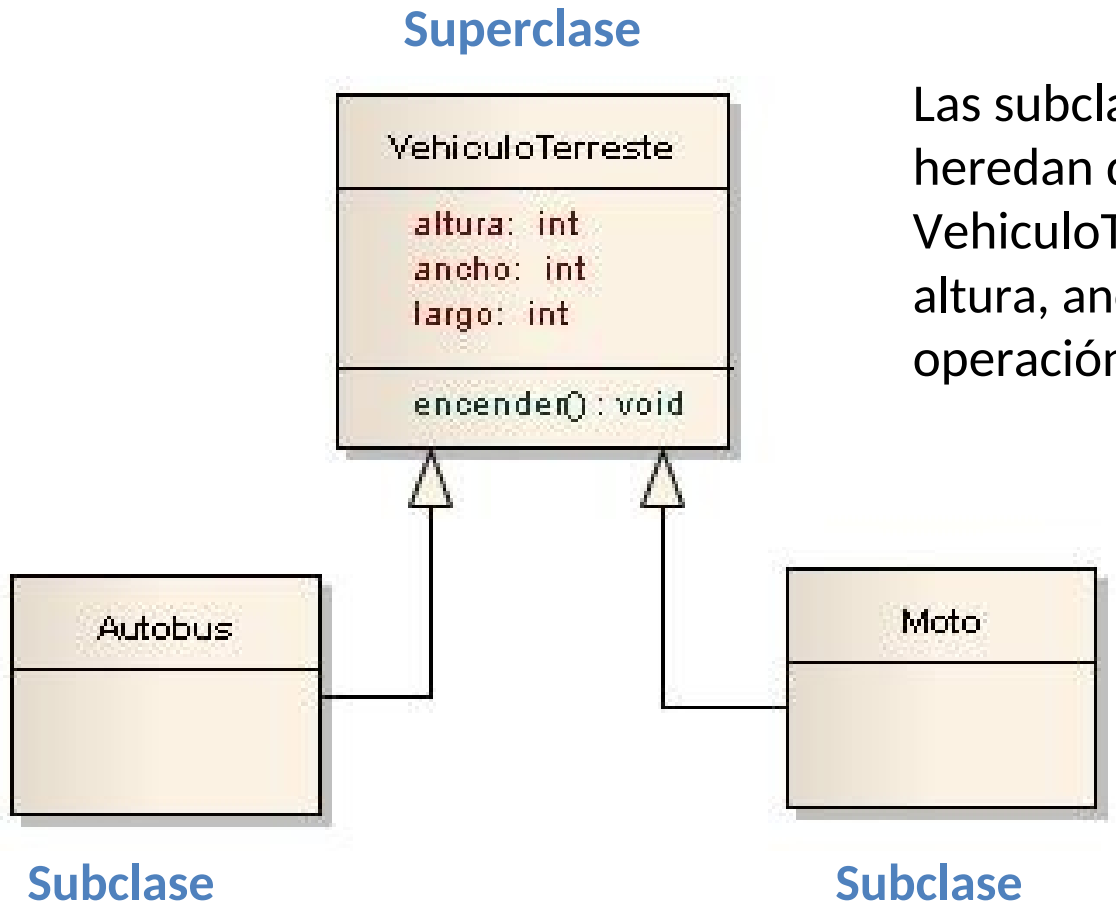
Ejemplo: un Auto **ES UN** VehiculoTerrestre



Mecanismo de Herencia - Ejemplo



Mecanismo de Herencia - Ejemplo



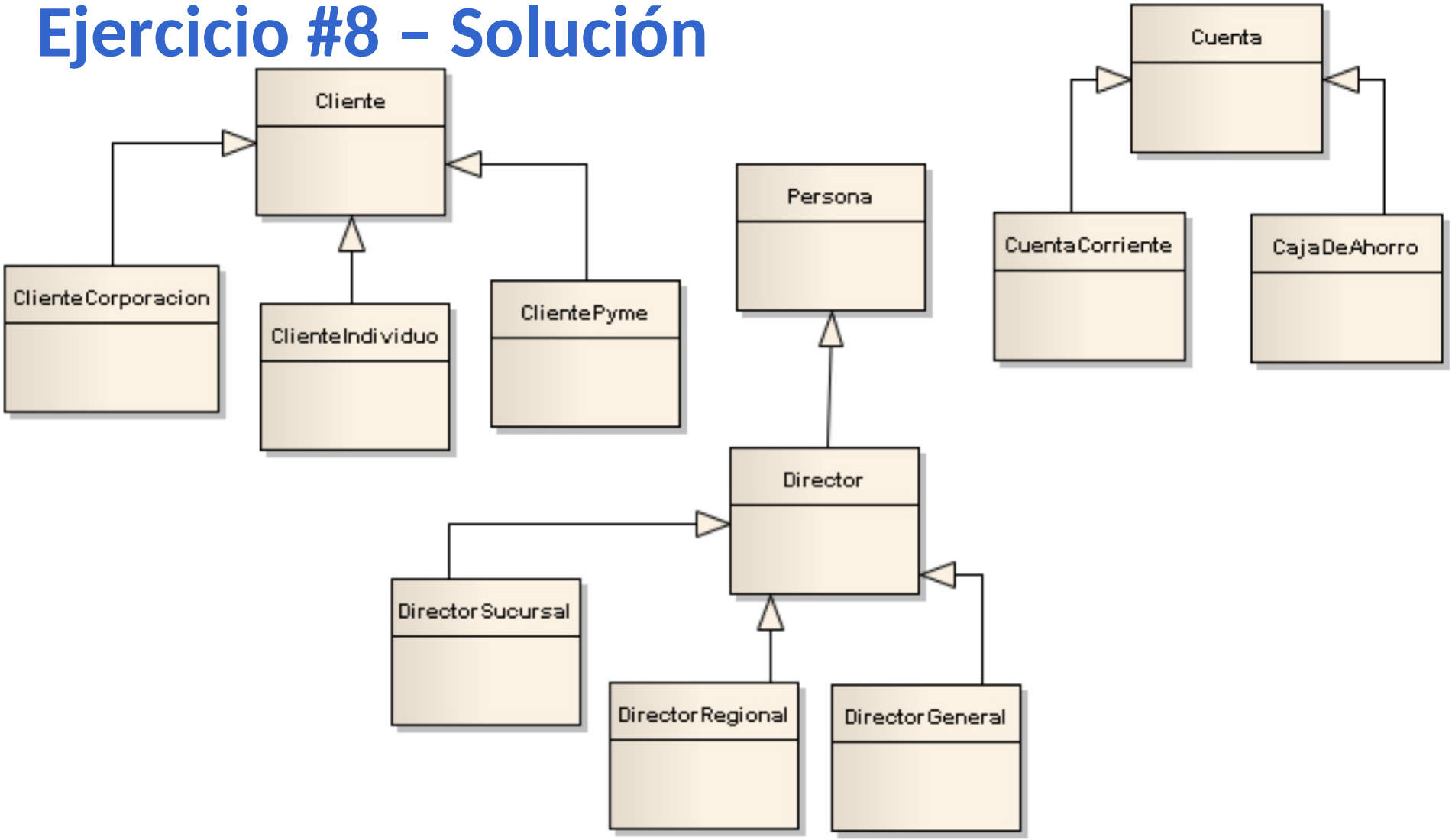
Las subclases Autobus y Moto heredan de la superclase VehiculoTerreste los atributos altura, anchura, largo, y también la operación encender()

Ejercicio #8 – Herencia

A partir de las clases detectadas previamente, identificar las relaciones de herencia que existen

TIP: Si consideras útil y/o necesario, puedes proponer nuevas clases.

Ejercicio #8 – Solución



Ejercicio #8 – Codificación

```
class Cliente {

        // Atributos
        String cuit;
        String direccion;

        /   Constructores
        /   Métodos
}
```

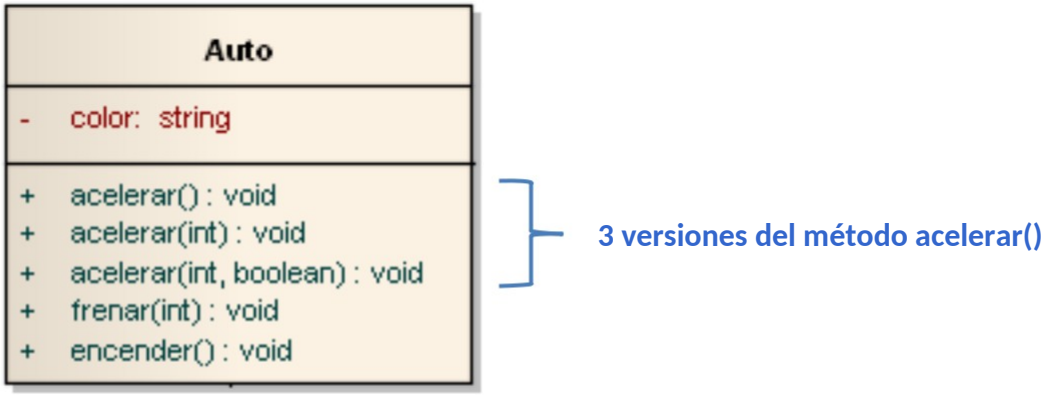
```
class ClientePyme extends Cliente {

        // Atributos - Los atributos heredados no se vuelven a codificar!
        String razonSocial;

        /   Constructores
        /   Métodos
}
```

Que es el Polimorfismo

Es la posibilidad de que una clase presente un comportamiento distinto de acuerdo a una situación

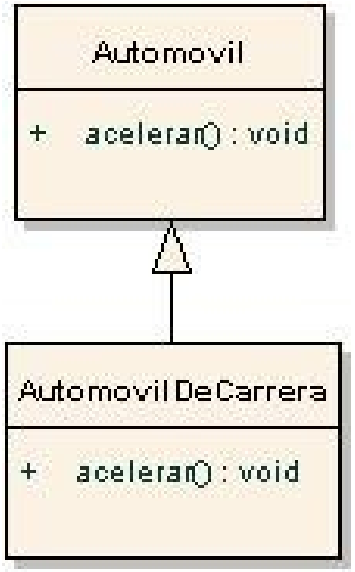
2 Tipos de Polimorfismo: sin redefinición y con redefinición

Polimorfismo sin redefinición: Una clase que posee varios métodos llamados iguales pero con diferentes firmas. También llamado *Sobrecarga de Operaciones*



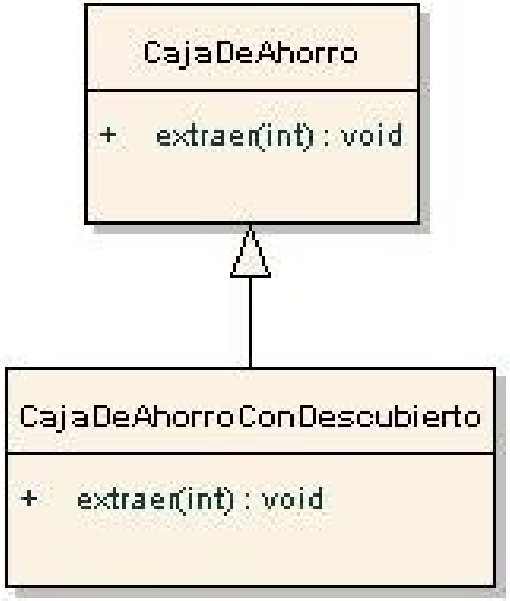
Que es el Polimorfismo

Polimorfismo con redefinición: Una subclase hereda métodos de su superclase pero decide modificarlos por que debería comportarse de forma diferente. También llamado *Redefinición de Métodos o Method Override*



Cualquier tipo de Automovil acelera igual que un AutomovilDeCarrera ?

Extraer dinero de una caja de ahorro sin descubierto se realiza de la misma forma que desde una caja con descubierto ?



Que es el Polimorfismo - Codificación

```
class CajaDeAhorro {

        / Atributos
        float saldo;

        / Métodos
        public void extraer(int monto) {
                saldo = saldo – monto;
        }
}
```

```
class CajaDeAhorroDescubierto extends CajaDeAhorro {

        // Atributos
        float saldoDescubierto;

        // Métodos
        public void extraer(int monto) {
                / este método se vuelve a escribir, misma firma pero diferente cuerpo
                / aquí el código que contemple la extracción con saldo descubierto
        }

}
```

Atributos de Instancia y de Clase

Atributos de Instancia: son atributos que pertenecen a un objeto en particular

Atributos de Clase: son atributos que pertenecen a la clase y no a un objeto o instancia de clase. Esto significa que son atributos compartidos por todos los objetos. También llamados *atributos estáticos*

Para definir un atributo estático se utiliza la palabra clave **static**

```
class CajaDeAhorroConDescubierto {

   / Atributos aquí
   public float saldo;
   public static int descubierto = 5000;

}
```

Asumimos en este caso que todas las cajas de ahorro cuentan con el mismo descubierto

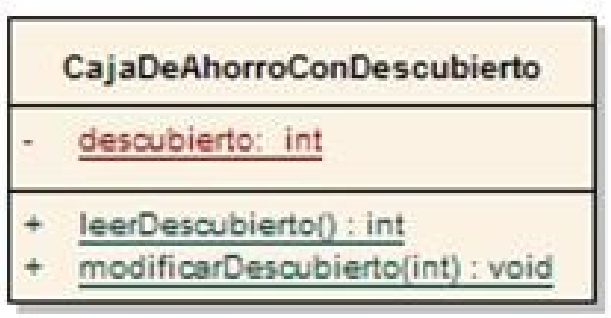
Métodos de Instancia y de Clase

Métodos de Instancia: son métodos que pertenecen a un objeto en particular e impactan en el comportamiento de ese objeto únicamente

Métodos de Clase: son métodos que pertenecen a la clase y no a un objeto o instancia de clase. Esto significa que son métodos compartidos por todos los objetos. También llamados *métodos estáticos*

Para definir un método estático se utiliza la palabra clave **static**

```
class CajaDeAhorroConDescubierto {
    / Atributos aquí
    public float saldo;
    private static int descubierto = 5000;
    / Metodos aquí
    public static int leerDescubierto()
        { return descubierto;
    }
}
```



Clases Abstractas y Clases Concretas

Las Clases Concretas

- son clases que se pueden instanciar
- por ejemplo la clase Alumno existe para generar diversos objetos del tipo Alumno

Las Clases Abstractas

- son clases que no se pueden instanciar

-representan conceptos muy genéricos de la realidad

- por ejemplo la clase Vehiculo o la clase Persona son conceptos muy abstractos, seria difícil pensar en armar un objeto a partir de estas clases

Codificación:

```
abstract class Persona {

    / Atributos aquí

    / Métodos aquí

}
```

Generación Automática de Código

El mecanismo de generación automática de código se denomina ***Ingeniería Directa***

Existen herramientas que permiten generar código fuente a partir de diagramas, por ejemplo el Enterprise Architect

Estas aplicaciones permiten generar código fuente en diversos lenguajes, como ser: C#, VB.NET, PHP, Java, Actionscript, y mas

FIN

Muchas Gracias!