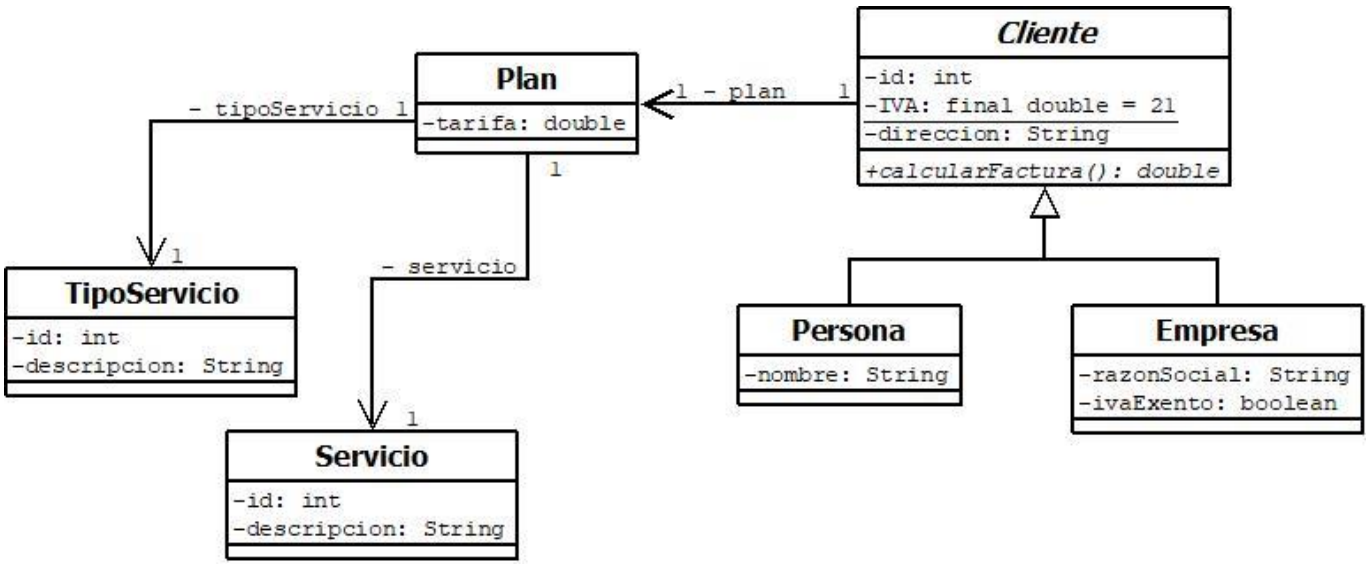
1° PARCIAL DE ALGORITMICA Y PROGRAMACION II - 2024

1) Un proveedor de Internet necesita un sistema para gestionar el cobro de sus servicios. La empresa tiene dos tipos de servicios: Básico y Premium y ofrece servicios de conexión por fibra óptica a distintas velocidades. Cada plan tiene una tarifa distinta para cada tipo de servicio y servicio ofrecido (velocidad de conexión). Los clientes pueden ser personas o empresas. El cálculo de la factura para las personas corresponde al valor de la tarifa del plan que tiene contratado más el 21% de iva. Para las empresas el cálculo de la factura también corresponde al valor de la tarifa del plan que tiene contratado si está exento al iva y se suma el 21% si no lo está.

Diagrama de Clases



Crear todas las clases con sus atributos y constructores, gets y sets, equals y toString. Implementar los métodos que correspondan.

2) Realizar un programa donde pruebe:

a) Cargar los siguientes planes:

- Básico, 25 mbits fibra óptica, $ 18.500
- Básico, 50 mbits fibra óptica, $ 21.700
- Premium, 50 mbits fibra óptica, $ 37.100
- Premium, 100 mbits fibra óptica, $ 54.500

b) Cargar un arreglo con personas y empresas (exentas y no exentas de iva) con diferentes planes contratados.

c) Recorrer el arreglo y mostrar el Id del cliente y el importe de la factura a cobrar para cada cliente y el importe total a cobrar a todos los clientes.

d) Recorrer el arreglo y mostrar el Id del cliente y el importe de la factura a cobrar para cada cliente y el importe total a cobrar a todos los clientes para el servicio de 50 mbits fibra óptica.

3) Dada una lista simplemente enlazada, agregar el siguiente método:

```
/**
 * Retorna la lista con todos los elementos de la lista pasada combinados después del
 * primer elemento de la lista.
```

```
 *
 * Por ejemplo:
 *
 * {A, B, C, D} {W, X, Y, Z} => {A, W, B, X, C, Y, D, Z}
 *
 * {A, B, C, D} {W, X} => {A, W, B, X, C, D}
 *
 * {A, B} {W, X, Y, Z} => {A, W, B, X, Y, Z}
 *
 * {A, B, C, D} {} => {A, B, C, D}
 *
 * {} {W, X, Y, Z} => {W, X, Y, Z}
 *
 * {} {} => {}
 *
 * @param SinglyLinkedList<E> l : lista con los elementos a combinar
 *
 * @return lista original con todos los elementos de la lista l combinados
 *         después del primer elemento de la lista original
 *
 */

     public void addCombineAfter(SinglyLinkedList<E> l)
```

4) Dada una lista simplemente enlazada, realice una nueva versión del método implementado en 3).

```
/**
 * Retorna la lista con todos los elementos de la lista pasada combinados antes del
 * primer elemento de la lista.
 *
 * Por ejemplo:
 *
 * {A, B, C, D} {W, X, Y, Z} => {W, A, X, B, Y, C, Z, D}
 *
 * {A, B, C, D} {W, X} => {W, A, X, B, C, D}
 *
 * {A, B} {W, X, Y, Z} => {W, A, X, B, Y, Z}
 *
 * {A, B, C, D} {} => {A, B, C, D}
 *
 * {} {W, X, Y, Z} => {W, X, Y, Z}
 *
 * {} {} => {}
 *
 * @param SinglyLinkedList<E> l : lista con los elementos a combinar
 *
 * @return lista original con todos los elementos de la lista l combinados
 *         antes del primer elemento de la lista original
 *
 */
     public void addCombineBefore(SinglyLinkedList<E> l)
```

IMPORTANTE:

Subir al Campus solamente los archivos con extensión .java.