

Proyecto Bimestral

Creación de código java para servicio de telefónica

INTEGRANTES: -Jean Iñiguez -Carlos Piedra

1. **Análisis del problema**

El problema plantea el desarrollo de un "Sistema de Gestión de clientes y facturación" para una telefónica que ofrece servicios de telefonía móvil a la comunidad universitaria. El sistema debe permitir el registro, actualización, consulta y eliminación de clientes, así como la facturación de los planes celulares que se ofrecen. Para abordar este problema, se propone el uso de la Programación Orientada a Objetos (POO) con los conceptos de herencia y polimorfismo.

El objetivo del sistema es proporcionar una herramienta para gestionar la información de los clientes de la telefónica, permitiendo el registro de sus datos básicos y la asignación de diferentes planes móviles según las necesidades y preferencias de cada cliente.

La telefónica cuenta con la infraestructura tecnológica para ofrecer servicios de comunicación móvil, pero la falta de un sistema de gestión de clientes y facturación impide su funcionamiento. Por tanto, se necesita una aplicación que permita registrar a los clientes y gestionar los planes móviles que ofrecen, incluyendo información específica para cada tipo de plan.

El cliente está compuesto por sus nombres, pasaporte/cédula, ciudad, marca, modelo y número de celular, además de dos atributos adicionales. Estos datos se deben almacenar y gestionar en el sistema.

Se deben ofrecer diferentes tipos de planes móviles que se caracterizan según su clasificación:

a. PlanPostPagoMinutosMegasEconomico:

Este plan tiene información sobre minutos, costo por minuto, megas en gigas, costo por cada giga y un porcentaje de descuento aplicado.

b. PlanPostPagoMinutos:

Este plan se caracteriza por tener información sobre minutos nacionales, costo por minuto nacional, minutos internacionales y costo por minuto internacional.

c. PlanPostPagoMegas:

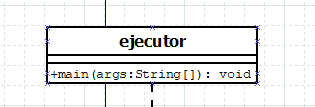
Este plan requiere información sobre megas expresados en gigas, costo por cada giga y una tarifa base.

d. PlanPostPagoMinutosMegas:

Este plan combina información sobre minutos, costo por minuto, megas expresados en gigas y costo por cada giga.

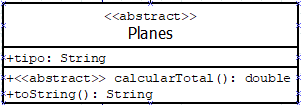
En resumen, el análisis del problema "Gestión de telefonía móvil estudiantil: Mov-UTPL" implica el desarrollo de un sistema de gestión de clientes y facturación que permita administrar los datos de los clientes y los planes móviles ofrecidos por la telefónica, utilizando los conceptos de POO como herencia y polimorfismo para gestionar los diferentes tipos de planes de manera eficiente.

1. **Explicación del UML**

****

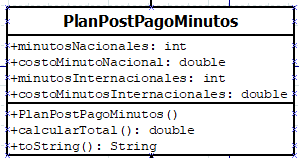
1. En este caso implementamos la clase ejecutor que es la encargada de inicializar el programa y el encargado de como lo va a visualizar el usuario.

https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo\_1b-7/blob/main/2\_SOLUCION/ProyectoPOO/src/VIEW/ejecutor.java

****

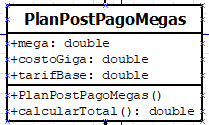
1. A continuación implementamos la clase Planes que en si es la clase padre ya que esta va a tener un método abstract que va a ser el de calcular el total y el que va a heredar un método de tipo String que va a ser para identificar cada plan.

https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo\_1b-7/blob/main/2\_SOLUCION/ProyectoPOO/src/MODEL/Planes.java



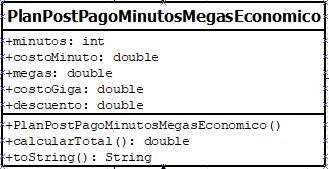
1. En este apartado creamos las clases para cada uno de los planes móviles con sus atributos y sus respectivos métodos y heredando lo de la clase Planes y calculando el total y aplicando cada uno su manera de calcularlo.

<https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo_1b-7/blob/main/2_SOLUCION/ProyectoPOO/src/MODEL/PlanPostPagoMinutos.java>



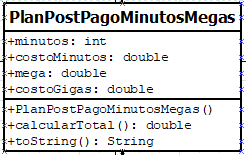
1. Aquí aplicamos la otra clase que igualmente como el apartado anterior hereda de la clase Planes y igualmente con su respectiva forma de calcular el total.

https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo\_1b-7/blob/main/2\_SOLUCION/ProyectoPOO/src/MODEL/PlanPostPagoMegas.java



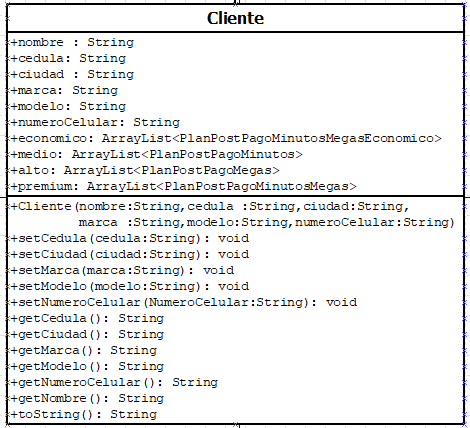
1. Esta es otra clase de los planes móviles q hace lo mismo que las anteriores.

<https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo_1b-7/blob/main/2_SOLUCION/ProyectoPOO/src/MODEL/PlanPostPagoMinutosMegasEconomico.java>



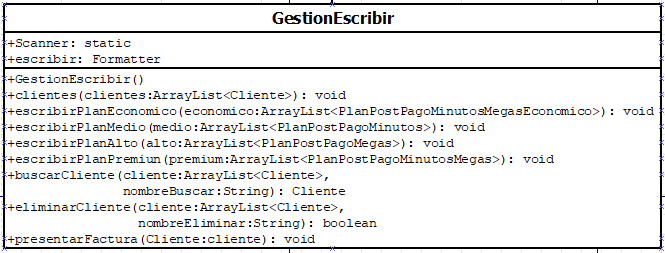
1. Este igual que los 3 anteriores aplica herencia y el polimorfismo.

<https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo_1b-7/blob/main/2_SOLUCION/ProyectoPOO/src/MODEL/PlanPostPagoMegas.java>



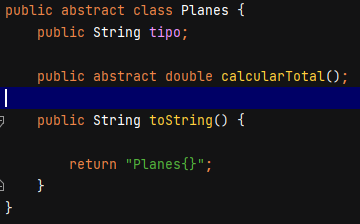
1. Esta es una clase muy importante ya que es donde el usuario va a dar todos sus datos y el Plan que eligieron los ArrayList se crearon para poder guardar todos los usuarion que ingresen sus datos.

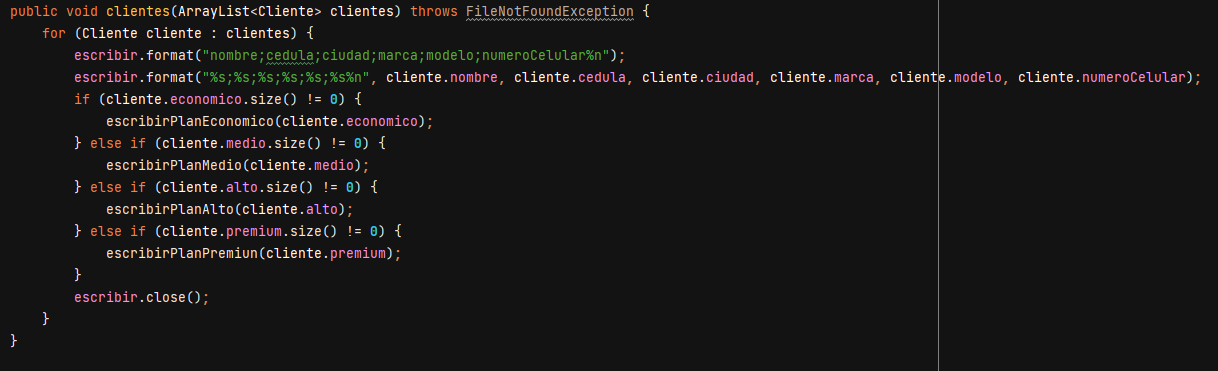
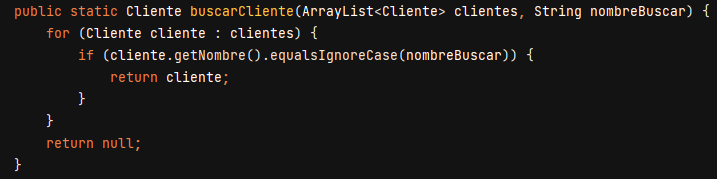
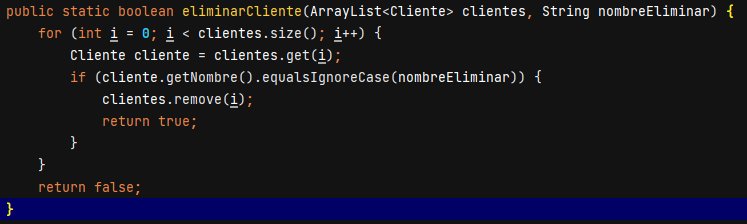
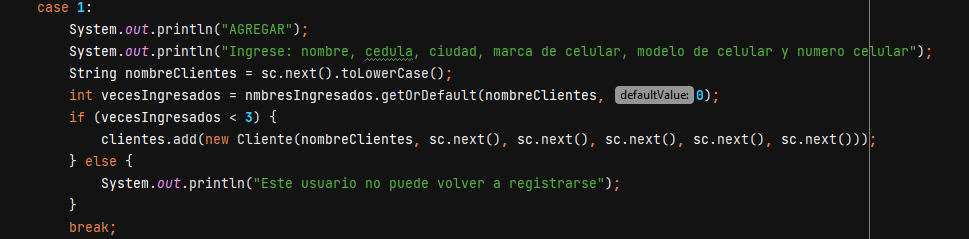
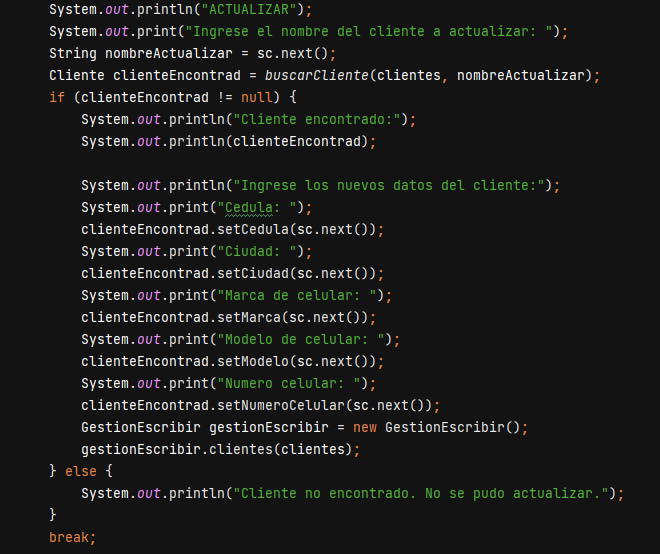
<https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo_1b-7/blob/main/2_SOLUCION/ProyectoPOO/src/MODEL/Cliente.java>

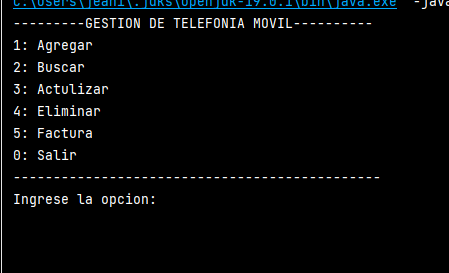
1. esta también es una clase muy importante ya que en esta guardamos todos los usuarios y hacemos todas las demás operaciones CRUD.

<https://github.com/ProOrientadaObjetos-P-B-AA2023/aa-trabfinal-poo-2b-grupotrabajo-final-poo_1b-7/blob/main/2_SOLUCION/ProyectoPOO/src/CONTROLER/GestionEscribir.java>

1. Explicación de partes importantes del código



1. Esta parte es importante ya que es la clase padre aquí heredamos a las demás clases el atributo tipo que mas a delante va a ser importante y también el método abstract que se llama calcular ya que nos sirve para hacer en cada uno de las demás clases su cálculo correspondiente.
2. En este método lo que hacemos es crear un arrayList para cada una de las clases o planes móviles y irlos guardando cada vez que ingresen alguna información sobre algún usuario.
3. En este método creamos un apartado llamado buscarCliente este método nos sirve para el caso de que el usuario le de en la opción de buscar lo llamamos a este método y lo que hace es buscar en el archivo si hay algún nombre comparado con lo que ingreso el usuario, en caso de que si encuentre el nombre lo retorna y muestra la información sobre lo que decidieron buscar.
4. Este es el método de eliminar en si lo que hace es pedirle al usuario que ingrese un nombre y ese lo busca en el ArrayList general y si lo encuentra lo elimina y vuelve a escribir el archivo, pero ya sin el usuario eliminado. En caso de que no este el nombre ingresado lanza un mensaje diciendo que el nombre no fue encontrado.
5. Este es el método de agregar esto en si lo que hace es pedirle al usuario ingresar todos sus datos y el plan elegido una vez ingresado todo lo guarda en su respectivo arrayList.
6. este es el último método y es uno de gran importancia ya que este sirve para poder actualizar algún dato de algún usuario, este pide el nombre y el resto lo manda para que el usuario ingrese todos los datos modificados excepto el nombre.
7. **Resultados**

****

