



UNIVERSIDAD DE COSTA RICA

SEDE DE GUANACASTE

Informática Empresarial
IF-2000 Programación I

I Proyecto Programado

Integrantes:

Álvarez Taisigüe Cris
Murillo Vásquez Erick

Carné B97785
Carné B98334

M.C.I Kenneth Sánchez Sánchez

Liberia – Guanacaste
19 de noviembre del 2019
II Ciclo

Índice

Índice.....	2
1. Introducción	3
2. Objetivos.....	4
2.1 Objetivo General:	4
2.2 Objetivo Especifico:	4
3. Descripción del problema y solución.....	5
3.1 Problema:	5
3.2 Solución:	6
4. Herramienta de Desarrollo	10
4.1 Hardware:	10
4.2 Software:.....	10
5. Descripción de Datos.....	11
6. Corrida del programa con datos suministrados	15
7. Código fuente	28
8. Problemas y Limitaciones	62
8.1 Problema:	62
8.2 Limitaciones:.....	62
9. Conclusión	63

1. Introducción

La recaudadora MENOSPAG S.A, recibe gran cantidad de clientes todos los días del año. Con esto hay una clara perspectiva de la dificultad de los funcionarios de la recaudadora para poder recibir un número de clientes exorbitante, y ser atendidos uno por uno para poder solicitar la información correspondiente de cada usuario. Asimismo, los funcionarios de la recaudadora deberán realizar cálculos según el recibo que el usuario desea pagar. Además, se le es difícil la búsqueda de recibos de los clientes por la gran cantidad de usuarios.

Con lo dicho anteriormente, La recaudadora MENOSPAG S.A necesitan de un programa que facilite los cálculos correspondientes de cada servicio públicos que ofrecen. Con el mismo fin de solucionar y facilitar los resultados de los cálculos de los impuestos y la solicitud de la información personal del usuario en un tiempo más eficaz. Igualmente, para agilizar el tiempo, desean que el programa tenga una implementación para poder buscar recibos tanto por NIS como por el monto más alto del recibo de servicio público.

Con el problema anterior, se elaboró un programa el cual soluciona las necesidades de la recaudadora, con el objetivo de tener eficacia a la hora de atender el usuario y a la vez proporcionar un sistema sencillo de entender para los funcionarios de la recaudadora.

El método utilizado para la elaboración de este programa fue desarrollado con el lenguaje de programación Java, mediante un entorno de programación de llamado Eclipse. Se busco que el código fuera claro y entendible para cualquier persona que lo interprete. Se tratará de comprender y analizar las funcionalidades del programa a continuación.

2. Objetivos

2.1 Objetivo General: Explicar el funcionamiento del código del programa realizado para facilitar las necesidades de la recaudadora MENOSPAG S.A por medio de un manual al usuario.

2.2 Objetivo Especifico:

- Comprender el problema y buscar la solución a la necesidad de la recaudadora MENOSPAG S.A.
- Comprender y describir las variables propuestas en el código.
- Explicar el funcionamiento del sistema de manera sencilla.

3. Descripción del problema y solución

3.1 Problema:

La recaudadora MENOSPAG S.A requiere un programa que le permita automatizar todos sus cobros en cuanto a emisión, cálculo y entrega de recibos por servicios públicos (agua, electricidad, teléfono (celular y residencial), servicio de cable TV e Internet).

Se deberá diseñar un programa, para ello se debe tomar en cuenta que los recibos están compuestos de: Nombre de usuario, DNI, Dirección, “encabezado” Factura por “Nombre del Servicio”, mes al cobro, NIS (número de identificación del servicio), Fecha de pago.

El AYA incluye costo hidrante (¢3525), consumo diario (m2) alquiler del medidor (¢2500), Número de días demora, y el Monto a pagar (consumo diario * 30 días + hidrante + alquiler) + (demora). La demora se calcula como (Número de días de demora *2800), el monto del m2 es de ¢155.

El ICE por electricidad incluye: consumo en Kws(kilowatts) cuyo costo individual es de ¢1125, además alumbrado público (¢950), impuesto bomberos (2.5% del consumo del producto de kws) atraso al pagar (S/N), si se atrasa debe cancelar Reconexión (¢21000), y Monto a pagar igual a (Kws*1125+alumbrado público + (Reconexión (si atrasa))).

Por Teléfono Residencial incluye Tarifa fija, nº de impulsos, Cantidad Minutos Plenos (costo ¢12), Minutos Reducidos (costo ¢7), tarifa internacional (¢9100), roaming(valor), costo de impulso ¢75, Monto a Pagar es igual (tarifa fija + (nº

impulsos *costo de impulso) +monto de producto minutos plenos+ monto producto de minutos reducidos + tarifa internacional + roaming).

Para el Teléfono Celular se toma en cuenta la Tarifa fija, Cantidad Minutos Plenos, Minutos Reducidos, Costo plenos ¢35, Costo reducidos ¢25, Cantidad de mensajes de texto, Costo mensaje 9,25, Monto a pagar es igual a (Tarifa fija + ((Minutos plenos * Costo plenos) + (Minutos reducidos * Costo reducidos) + (Mensajes * Costo mensaje)).

El servicio de Cable TV e internet incluye Tarifa fija (¢15800), Alta definición, Servicio digital, Cable modem (¢10500), Gigas (Paquetes (PKT) de 50G ¢30000, 10G ¢12000, 4GB ¢6000) (Monto a pagar es igual a (Tarifa fija + Alta definición + Servicio digital + Cable modem+ Pkt de Internet).

3.2 Solución:

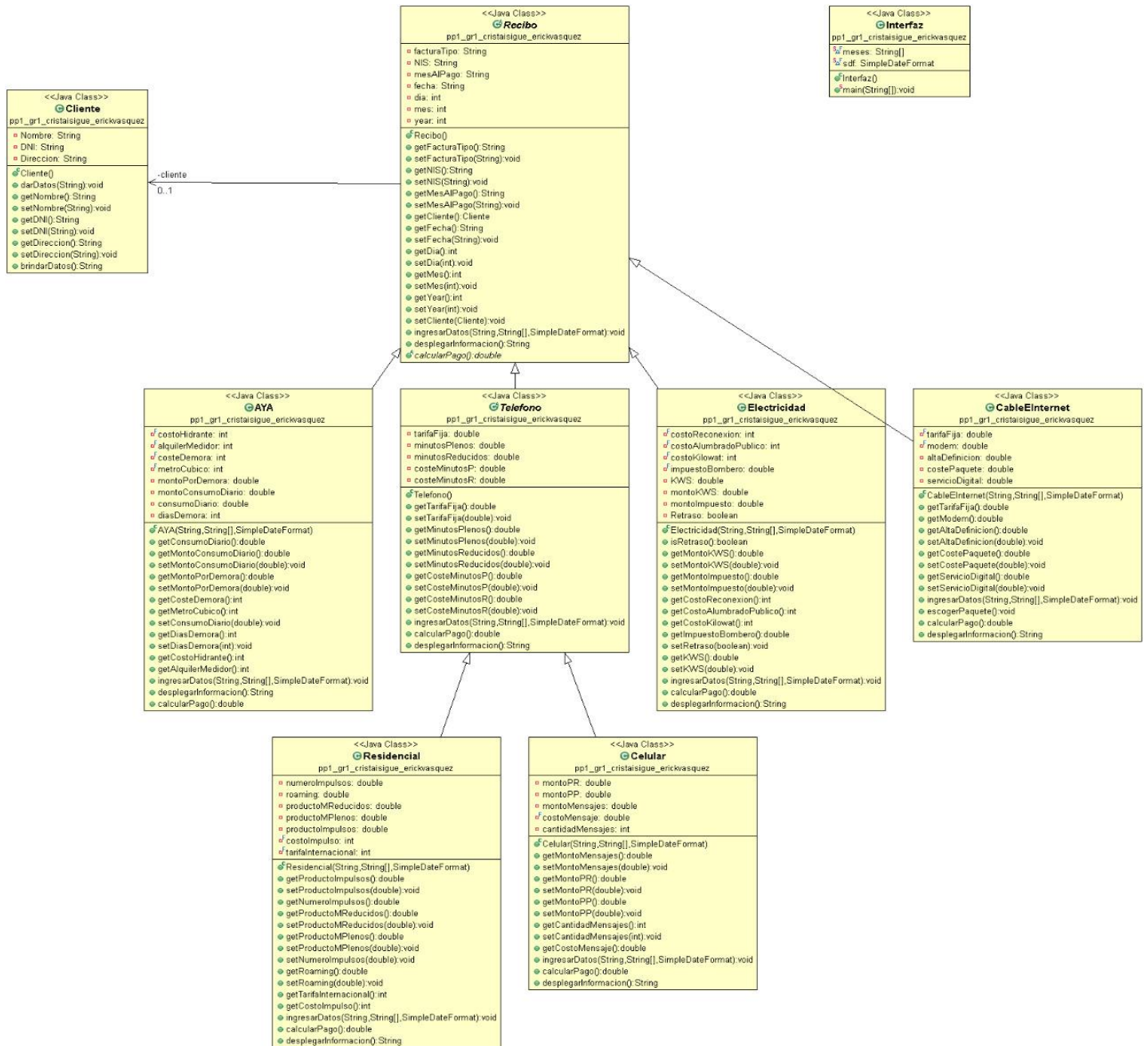
Nuestra solución consiste en la implementación de la programación orientada a objetos, por medio de clases heredadas, agregada y abstractas, sin descuidar el encapsulamiento, polimorfismo para facilitar y respetar el POO, utilizando ligamentos dinámicos, array de objetos y a la vez constantes y variables para el almacenamiento de distintos valores a presentar por cada categoría que se nos especificó, es decir que para cada categoría existirá una clase con sus respectivos atributos, tomando en cuenta el enunciado y con este el problema, se llegó a determinar que la opción más eficaz fue la implementar la separación de los distintos servicios en clases separadas (Servicio ICE, Servicio AYA, Servicio Cable E Internet, servicio Teléfono Residencial, servicio teléfono celular) estas heredan de

una clase común la de Recibo, esto se hace de esta manera, para poder implementar correctamente la característica de polimorfismo a la hora de guardar los distintos recibos en un solo arreglo el cual se emplea en la “Interfaz” de 15 posiciones “Recibo recibos[]=new Recibo[15];” , algo muy importante a mencionar es que se emplean métodos para ser sobrescritos en la clase Recibo, en dicha existen los métodos de ingresarDatos(), desplegarInformacion() y calcularPago(), el primero pedirá los datos generales que constituyen un recibo , este será sobrescrito por las distintas clases pidiendo los datos generales del recibo y los que constituyen a dicho servicio dependiendo de sus atributos y cálculos a efectuar, el segundo método mostrara la información pertinente a la clase recibo la cual es constituida por datos generales , en cada servicio se sobrescribirá este método para así también además de mostrar los datos generales del recibo se muestren los del servicio pertinente, el tercer método abstracto, se encargara de facilitarnos el despliegue de cálculos, los cuales dependiendo del servicio se les sobrescribirá con distintos criterios, para luego poder acceder a estos cálculos desde la superclase.

En lo que respecta la interfaz y la presentación e interacción con el usuario, mediante el empleo de ciclos y selectores “switch” se desarrolló un menú conformado por 5 opciones (1-Emitir,2-Visualizar recibos,3-Buscar Recibo por NIS,4-Recibo con el monto más alto a pagar), en la primera opción se desarrolló un submenú el cual le da la posibilidad al usuario de elegir que recibo emitir y si hace lo desea puede regresarse al menú principal, independientemente del servicio que se escoja el usuario deberá ingresar los datos pertenecientes a los atributos de la clase Recibo y su agregación Cliente, después la clase perteneciente al servicio

seleccionado determinara que datos pedir en función de poder realizar sus cálculos y desplegar información con los métodos sobrescritos “desplegarInformacion()” y “calcularPago()”, para lo que concierne la opción 2 de visualizar todo los recibos se utilizó un ciclo “for ” que recorrerá el arreglo “Recibos recibos[]” y en función de cada posición llamara a los método sobrescritos “desplegarInformacion()” y “calcularPago()” perteneciente a la clase Recibo, la opción 3 la cual es la de buscar un recibo por NIS específico se desarrolló mediante la obtención del valor a buscar (dado por el usuario) y el uso de un ciclo “for ” que fuese recorriendo el arreglo “recibos[]” y obteniendo el atributo NIS perteneciente a la superclase mediante su respectivo método Getter, si este valor es igual al provisto por el usuario se le mostrara que lo ha encontrado y desplegara su información usando el método sobrescrito “desplegarInformacion()” y su respectivo monto a pagar con el método “calcularPago()”, para la opción 4 donde se muestra el monto más alto pagar, se empleó un algoritmo en el cual por defecto se cree que el valor mayor se encuentra en el recibo con la posición 0 (variable “major”) en el arreglo “recibo[]” se recorre el recibo mediante un ciclo “for ”, y en cada posición existe una condicional que determina si la posición en el arreglo “recibos[]” determina por la variable incremental del ciclo y llamando al método sobrescrito “calcularPago()” es mayor al mismo método pero desde la posición supuesta (“major”) que es la mayor, si esto es así la variable “major” cambiara su valor al de la variable incremental del ciclo “for”, al final se visualizara el recibo con la posición determinada después de los cálculos realizados en el ciclo for “recibos[major].desplegarInformacion()” y su respectivo monto a pagar “recibos[major].calcularPago()”

La última opción (5) finalizara el programa mediante un “System.exit(0)”.



(Diagrama de clases Empleado)

4. Herramienta de Desarrollo

4.1 Hardware:

Computador: HP 240 G5 Notebook PC

Procesador: Intel(R) Celeron(R) CPU N3060

Velocidad del procesador: 1.60 GHz

Tipo de Sistema: 64 bits

Memoria RAM: 4,00GB (3,86GB utilizables)

Memoria de almacenamiento: (C:133GB/199GB) (D:159GB/199GB)

4.2 Software:

Sistema Operativo: Windows 10 home Single Language

Versión: 1803

Fabricante del sistema: Microsoft Corporation

Office: Microsoft Office Professional Plus 2019

Editor de texto: Word

Versión de editor de texto: 1905

Programa Utilizado: Eclipse 2019-09

Versión de Java: Jdk 1.8.0_212

5. Descripción de Datos

En este apartado se demostrarán y explicaran algunas de las variables y métodos que poseen una alta relevancia en este sistema.

1- “Recibo recibo[]=new Recibo[15]” , ”recibo[]” (Interfaz.java)este arreglo se encargara de organizar y almacenar los distintos servicios emitidos, gracias a la característica de polimorfismo las clases derivadas que corresponden a los servicios pueden sobrescribir sus métodos para un fácil manejo en lo que respecta la recepción, procesamiento y visualización de datos.

2- “sdf” “SimpleDateFormat” (Interfaz.java) esta variable dará un formato a las fechas que posteriormente ingresemos.

3-metodo “ingresarDatos()” (Recibo.java) este método posibilita ser sobrescrito en las clases derivadas “servicios”, en dicha se piden los datos para rellenar los atributos pertenecientes a la clase de Recibo, cuando se sobrescribe estos mismos datos han de pedirse sumado a los que la clase derivada ocupo para su pertinente función.

4-metodo “desplegarInformacion()” (Recibo.java) este método desplegara la información general (atributos Recibo) así como también los atributos pertenecientes a la clase Cliente , al poder ser sobrescrito, posibilita mostrar los anteriores datos más los que la clase derivada determine.

5-metodo “calcularPago()” (Recibo.java) el motivo por el cual se implementa este método abstracto con la capacidad de sobrescribirse por las clases derivadas “servicios”, es para conferir una versatilidad a la hora de calcular el monto total a pagar de los recibos por emitir, esto porque cada servicio presenta distintos criterios que influyen en la determinación del monto total a pagar, esto facilita el

manejo de datos, por el hecho que independientemente del servicio se podrá obtener su monto total a pagar mediante la superclase "Recibo.java".

6-Atributos por clase (cada uno presenta su propio método set y get en la clase determinada)

a)Recibo (superclase, datos generales, abstracta):

- facturaTipo,:Se guarda el tipo de servicio por emitir.
- NIS:NIS del servicio
- mesAlPago; mes al pago de ese servicio
- Cliente cliente: contendrá los métodos y atributos perteneciente al cliente.
- fecha: se guarda la fecha con el formato implementado
- dia: dia cuando se pagó el servicio.
- mes: mes cuando se pagó el servicio.
- year; año cuando se pago el servicio.

b)Cliente(información acerca del cliente, agregación a Recibo):

- Nombre: Se guarda el nombre del cliente.
- DNI: se guarda el DNI del cliente
- Direccion: se guarda la dirección del cliente.

c)AYA(Servicio AYA):

- costoHidrante=3525: costo del hidrante.
- alquilerMedidor=2500: costo por alquiler de medidor.
- costeDemora=2800: costo por si existe demora en el pago.
- metroCubico=155: costo del metro cubico de agua.
- montoPorDemora: monto calculado si existe demora en el pago.
- montoConsumoDiario: monto calculado producto del consumo diario.
- consumoDiario: consumo diario.
- diasDemora: días de demora en caso de una demora en el pago.

d)Electricidad (Servicio ICE):

- costoReconexion=21000: costo de reconexión del servicio.
- costoAlumbradoPublico=950: costo por el alumbrado público.
- costoKilowat=1125: costo individual del kilowatt.
- impuestoBombero=0.025: porciento a cobrarse por impuesto del servicio de bomberos.
- KWS, montoKWS: cantidad de kilowatts consumidos.
- montoImpuesto: monto calculado por el porciento de los impuestos por el servicio de bomberos
- Retraso: determina se existió retraso en el pago del servicio.

e)Telefono(clase abstracta de la cual derivan “Celular” y “Residencial”):

- tarifaFija: se guarda el coste de la tarifa fija.
- minutosPlenos: se guarda la cantidad de minutos plenos.
- minutosReducidos: se guarda la cantidad de minutos reducidos.
- costeMinutosP: se determina el coste del minuto pleno.
- :costeMinutosR: se determina el coste de minuto reducido

f)Celular(Servicio telefónico celular, derivada de la clase Telefono):

- montoPR: monto calculado producto de los minutos reducidos.
- montoPP: monto calculado producto de los minutos plenos.
- montoMensajes: monto calculado producto de los mensajes
- costoMensaje=9.25: coste del mensaje.
- cantidadMensajes: cantidad determinada de mensajes

g)Residencial (Servicio teléfono residencial, derivada de la clase Telefono):

- numeroImpulsos: se determina el número de impulsos.
- Roaming: valor del Roaming.
- productoMReducidos: monto calculado producto de los minutos reducidos.
- productoMPlenos: monto calculado producto de los minutos plenos.
- productoImpulsos; monto calculado producto del número de impulsos.
- costoImpulso=75: el coste del impulso.

-tarifaInternacional=9100: costo de tarifa internacional.

h) CableEInternet (servicio cable e internet):

-tarifaFija=15800: coste de tarifa fija.

-modem=10500: coste por alquiler del modem.

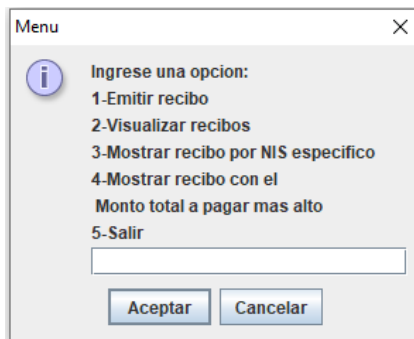
-altaDefinicion: costo determinado por alta definición.

-costePaquete: costo determinado por el paquete de internet suscrito

-servicioDigital: coste determinado por el servicio digital

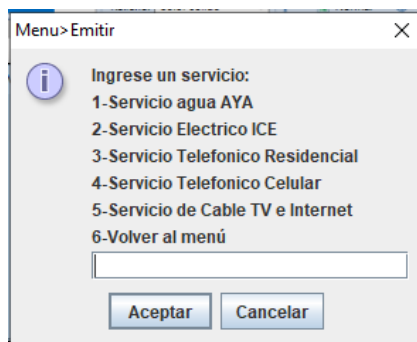
6. Corrida del programa con datos suministrados

1-Una vez iniciado el sistema le aparece un menú como este dispone de 5 opciones estas son:



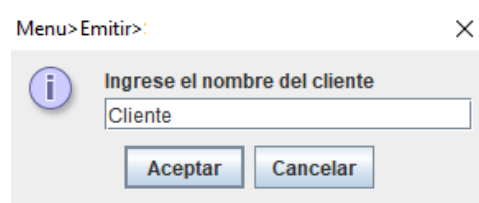
A screenshot of a Windows-style dialog box titled "Menu". It contains an information icon (i) and the text "Ingrese una opcion:". Below this, there is a list of five options: "1-Emitir recibo", "2-Visualizar recibos", "3-Mostrar recibo por NIS especifico", "4-Mostrar recibo con el Monto total a pagar mas alto", and "5-Salir". At the bottom of the list is a text input field. Below the input field are two buttons: "Aceptar" and "Cancelar".

2-Si usted selecciono la primera opción se llevará a un menú como este donde deberá seleccionar el servicio para efectuar un recibo.



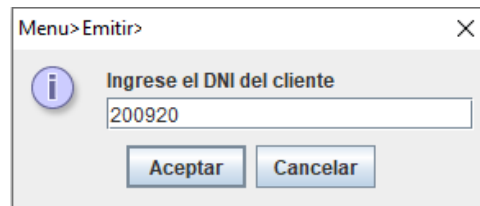
A screenshot of a Windows-style dialog box titled "Menu> Emitir". It contains an information icon (i) and the text "Ingrese un servicio:". Below this, there is a list of six options: "1-Servicio agua AYA", "2-Servicio Electrico ICE", "3-Servicio Telefonico Residencial", "4-Servicio Telefonico Celular", "5-Servicio de Cable TV e Internet", and "6-Volver al menú". At the bottom of the list is a text input field. Below the input field are two buttons: "Aceptar" and "Cancelar".

3-Notese que independientemente de la opción que usted seleccione se le pedirá lo siguiente:



A screenshot of a Windows-style dialog box titled "Menu> Emitir>". It contains an information icon (i) and the text "Ingrese el nombre del cliente". Below this text is a text input field with the placeholder text "Cliente". At the bottom of the dialog are two buttons: "Aceptar" and "Cancelar".

(a-Nombre del cliente)



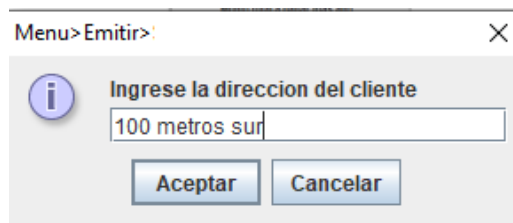
Menu> Emitir>

i Ingrese el DNI del cliente

200920

Aceptar Cancelar

(b-DNI del cliente)



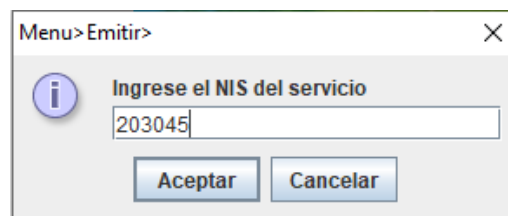
Menu> Emitir>

i Ingrese la direccion del cliente

100 metros sur

Aceptar Cancelar

(c-Dirección del cliente)



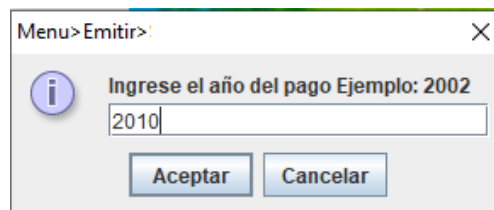
Menu> Emitir>

i Ingrese el NIS del servicio

203045

Aceptar Cancelar

(d-NIS del servicio)



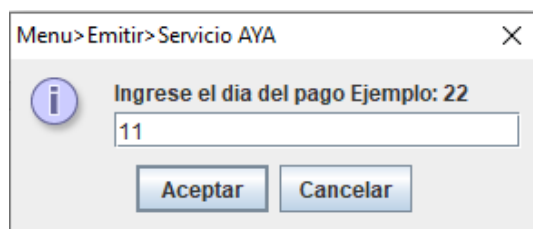
Menu> Emitir>

i Ingrese el año del pago Ejemplo: 2002

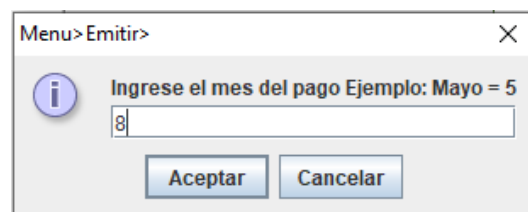
2010

Aceptar Cancelar

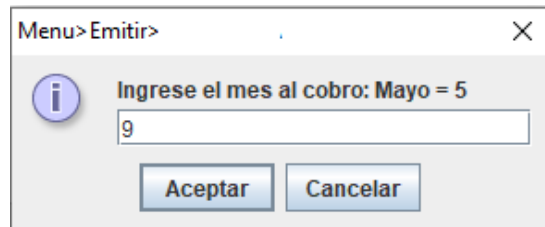
(e-El año cuando se pagó)



(f-El día en el cual se pagó)

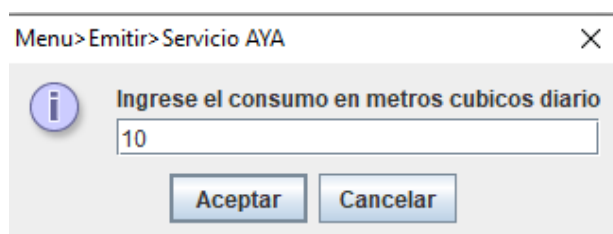


(g-El mes cuando se pagó)

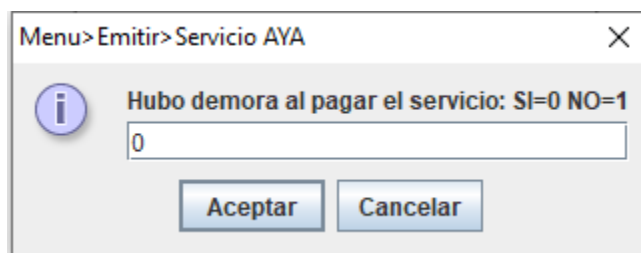


(h-El mes al cobro)

4-Si usted eligió la primera opción para emitir un recibo del servicio del AYA adjuntada mente a los anteriores se le pedirá que ingrese los siguientes datos



(a-Consumo en metros cúbicos)



(b-Se le preguntara si existió demora en el pago, si fue así se le pedirá que indica los días de demora)

Menu>Emitir>Servicio AYA

Número de días de demora

10

Aceptar Cancelar

(c-Por si hubo demora en el pago del servicio)

Factura emitida

Factura por Servicio AYA

Nombre del cliente:Cliente
 DNI del cliente:200920
 Direccion del cliente:100 metros sur
 Identificacion NIS:203045
 Fecha de pago:11/08/2010
 Mes al cobro:septiembre
 Consumo diario:10.0
 Número de días de demora:10
 Monto a pagar por demora:28000.0
 Monto producto del consumo diario:300.0
 Costo por alquiler de hidrante:3525
 Costo por alquiler de medido:2500
 Monto total a pagar:34325.0

Aceptar

(d-Se le presentara una factura como esta con los datos pertinentes (Cliente, monto total a pagar), los cálculos los valores ingresados y los criterios acerca de ese servicio)

5-Si usted eligió la opción (2) para emitir un recibo del servicio ICE, se le pedirá que ingrese los siguientes datos

Menu>Emitir>Servicio ICE

Ingrese la cantidad de KWS consumidos

6

Aceptar Cancelar

(a-Kilowatts consumidos)

Menu>Emitir>Servicio ICE

¿Existió retraso en el pago? :Ingrese 0 =SI , 1=NO

0

Aceptar Cancelar

(b-Se le preguntara si existió retraso en el pago, si fue así se le cobra la reconexión de dicho)

Factura emitida

Factura por Servicio Electrico ICE

Nombre del cliente:Cliente
 DNI del cliente:200920
 Direccion del cliente:100 metros sur
 Identificacion NIS:203046
 Fecha de pago:11/08/2010
 Mes al cobro:septiembre
 Retraso en el pago:Si, si se le cobra la reconexion de (21000)
 Consumo de KWS:6.0
 Impuesto bombero:168.75
 Monto producto del consumo de KWS:6750.0
 Cobro por alumbrado publico:950
 Costo de KiloWatt individual:1125
 Monto total a pagar:28868.75

Aceptar

(c-Se le presentara una factura como esta con los datos pertinentes (Cliente, monto total a pagar), los cálculos los valores ingresados y los criterios acerca de ese servicio)

6-Si usted eligió la opción (3) para emitir un recibo del servicio Teléfono Residencial, se le pedirá que ingrese los siguientes datos

Menu> Emitir> Telefono

i Ingrese el coste de la tarifa fija

1000

Aceptar Cancelar

(a-Coste de la tarifa fija)

Menu> Emitir> Telefono

i Ingrese la cantidad de minutos Plenos

10

Aceptar Cancelar

(b-la cantidad de minutos plenos)

Menu> Emitir> Telefono

i Ingrese la cantidad de minutos Reducidos

5

Aceptar Cancelar

(c-La cantidad de minutos reducidos)

Menu> Emitir> Telefono> Telefono Residencial

i Ingrese el número de impulsos

3

Aceptar Cancelar

(d-El número de impulsos)

Menu>Emitir>Telefono>Telefono Residencial

i Ingrese el valor del Roaming

2000

Aceptar Cancelar

(e-El valor del Roaming)

Factura emitida

i Factura por Servicio Telefonico Residencial

Nombre del cliente:Cliente
 DNI del cliente:200920
 Direccion del cliente:100 metros sur
 Identificacion NIS:203047
 Fecha de pago:11/08/2010
 Mes al cobro:septiembre
 Cantidad minutos reducidos:5.0
 Cantidad minutos plenos:10.0
 Tarifa fija:1000.0
 Costo del minuto pleno:12.0
 Costo del minuto reducido:7.0
 Valor de roaming:2000.0
 Número de impulsos:3.0
 Monto por minutos reducidos:49.0
 Monto por minutos plenos:120.0
 Monto por numero de impulsos:225.0
 Monto tarifa internacional:9100
 Monto total a pagar:12494.0

Aceptar

(f-Se le presentara una factura como esta con los datos pertinentes (Cliente, monto total a pagar), los cálculos los valores ingresados y los criterios acerca de ese servicio)

7-Si usted eligió la opción (4) para emitir un recibo del servicio Teléfono celular, se le pedirá que ingrese los siguientes datos

Menu>Emitir>Telefono


i Ingrese el coste de la tarifa fija

1000

Aceptar Cancelar


(a-Coste de la tarifa fija)

Menu>Emitir>Telefono

 Ingrese la cantidad de minutos Plenos


(b-la cantidad de minutos plenos)

Menu>Emitir>Telefono

 Ingrese la cantidad de minutos Reducidos

(c-La cantidad de minutos reducidos)

Menu>Emitir>Telefono>Telefono Celular

 Ingrese la cantidad de mensajes

(d-La cantidad de mensajes)

Factura emitida

Factura por Servicio Telefono Celular

Nombre del cliente: Cliente
DNI del cliente: 200920
Direccion del cliente: 100 metros sur
Identificacion NIS: 203049
Fecha de pago: 11/08/2010
Mes al cobro: septiembre
Cantidad minutos reducidos: 5.0
Cantidad minutos plenos: 10.0
Tarifa fija: 1000.0
Costo del minuto pleno: 35.0
Costo del minuto reducido: 25.0
Cantidad de mensajes: 4
Costo por mensaje: 9.25
Monto por minutos plenos: 350.0
Monto por minutos reducidos: 125.0
Monto por mensajes: 37.0
Monto total a pagar: 1512.0

Aceptar

(f-Se le presentara una factura como esta con los datos pertinentes (Cliente, monto total a pagar), los cálculos los valores ingresados y los criterios acerca de ese servicio)

8-Si usted eligió la opción (5) para emitir un recibo del servicio Cable E Internet, se le pedirá que ingrese los siguientes datos

menu> Emitir> Servicio Cable TV e Internet

Ingrese el coste del servicio digital

1000

Aceptar Cancelar

(a-Coste servicio digital)

menu>Emitir>Servicio Cable TV e Internet

i Ingrese el coste por alta definicion

1000

Aceptar Cancelar

(b-Coste por alta definición)

Entrada

? Ingrese el paquete suscrito:

1- Paquete 50GB (30000 colones)
2- Paquete 10GB (12000 colones)
3- Paquete 4GB (6000 colones)

2

Aceptar Cancelar

(c-Deberá seleccionar el paquete suscrito)

Factura emitida

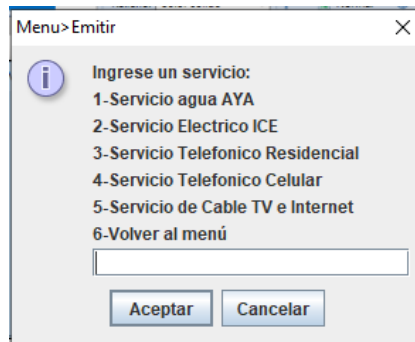
i Factura por Servicio Cable e Internet

Nombre del cliente:Cliente
DNI del cliente:200920
Direccion del cliente:100 metros sur
Identificacion NIS:203050
Fecha de pago:11/08/2010
Mes al cobro:septiembre
Coste de alta definicion:1000.0
Coste del paquete elegido:12000.0
Coste de servicio digital:1000.0
Tarifa fija:15800.0
Coste cable modem:10500.0
Monto total a pagar:40300.0

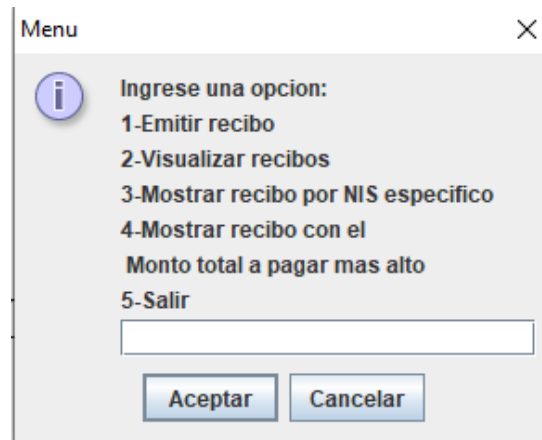
Aceptar

(f-Se le presentara una factura como esta con los datos pertinentes (Cliente, monto total a pagar), los cálculos los valores ingresados y los criterios acerca de ese servicio)

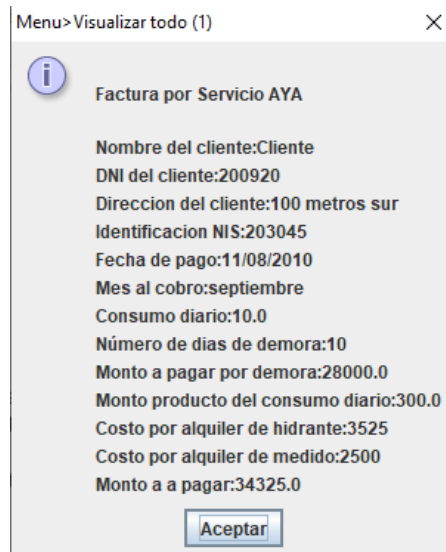
9-Si su opción fue el número 6, se volverá al menú principal.



(Ingresando la opción 6)



10-La opción 2 del menú llevara a la visualizan de los hasta ahora emitidos recibos



Menu>Visualizar todo (2) X

i Factura por Servicio Electrico ICE

Nombre del cliente:Cliente
 DNI del cliente:200920
 Direccion del cliente:100 metros sur
 Identificacion NIS:203046
 Fecha de pago:11/08/2010
 Mes al cobro:septiembre
 Retraso en el pago:Si, si se le cobra la reconexion de (21000)
 Consumo de KWS:6.0
 Impuesto bombero:168.75
 Monto producto del consumo de KWS:6750.0
 Cobro por alumbrado publico:950
 Costo de KiloWatt individual:1125
 Monto a pagar:28868.75

Aceptar

11-La opción 3 del menú le permitirá encontrar un recibo por el número de NIS

Menu>NIS Especifico X

i Ingrese el NIS a buscar:

203046

Aceptar Cancelar

(Se el NIS para buscar ese recibo)

Menu>NIS Especifico X

i Se encontro:

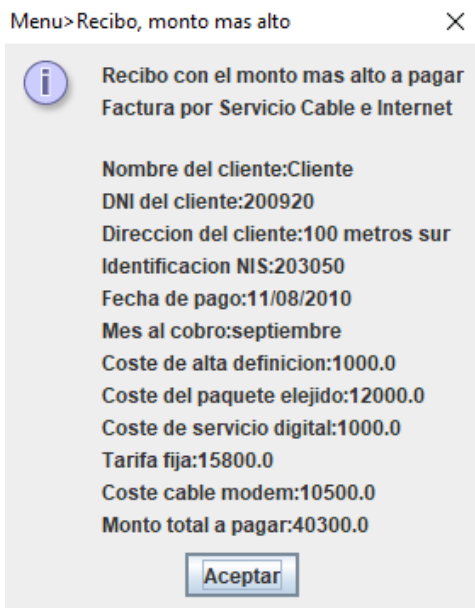
Factura por Servicio Electrico ICE

Nombre del cliente:Cliente
 DNI del cliente:200920
 Direccion del cliente:100 metros sur
 Identificacion NIS:203046
 Fecha de pago:11/08/2010
 Mes al cobro:septiembre
 Retraso en el pago:Si, si se le cobra la reconexion de (21000)
 Consumo de KWS:6.0
 Impuesto bombero:168.75
 Monto producto del consumo de KWS:6750.0
 Cobro por alumbrado publico:950
 Costo de KiloWatt individual:1125
 Monto total a pagar28868.75

Aceptar

(Se encuentra el recibo)

12-La opción número cuatro le permitirá determinar cuál recibo tiene el mayor monto a pagar de los actuales emitidos



(Se muestra el recibo con el monto a pagar más alto)

13-La opción número cinco (salir) finalizara el sistema.

7. Código fuente

```
1. package pp1_gr1_cristaisigue_erickvasquez;
2. import javax.swing.JOptionPane;
3.
4. public class Cliente {
5.
6.     //Atributos
7.     private String Nombre;
8.     private String DNI;
9.     private String Direccion;
10.
11.     //Constructor vacio
12.     public Cliente() {
13.
14.     }
15.
16.     //Metodo que se utilizar en el metodo "ingresarDatos()" de la supercl
17.     ase "recibo" para brindar los datos a los atributos de esta clase Cliente
18.     public void darDatos(String tipo) {
19.
20.         setNombre(JOptionPane.showInputDialog(null,"Ingrese el nombre del
21. cliente","Menu>Emitir>" + tipo, JOptionPane.INFORMATION_MESSAGE));
22.         setDNI(JOptionPane.showInputDialog(null,"Ingrese el DNI del clien
23. te","Menu>Emitir>" + tipo, JOptionPane.INFORMATION_MESSAGE));
24.         setDireccion(JOptionPane.showInputDialog(null,"Ingrese la direcci
25. on del cliente","Menu>Emitir>" + tipo, JOptionPane.INFORMATION_MESSAGE));
26.     }
27.
28.     //Metodos SET y GETTER pertenecientes a esta clase para obtener y mod
29. ificar atributos de esta clase
30.     public String getNombre() {
31.         return Nombre;
32.     }
33.
34.     public void setNombre(String nombre) {
35.         Nombre = nombre;
36.     }
37.
38.     public String getDNI() {
39.         return DNI;
40.     }
41.
42.     public void setDNI(String dNI) {
43.         DNI = dNI;
44.     }
45. }
```

```

39.     }
40.
41.     public String getDireccion() {
42.         return Direccion;
43.     }
44.
45.     public void setDireccion(String direccion) {
46.         Direccion = direccion;
47.     }
48.
49.     //Metodo utilizado en el metodo de la super clase "desplegarInformacion()" para visualizar los atributos de esta clase
50.     public String brindarDatos() {
51.         String texto="";
52.         texto+="Nombre del cliente:"+getNombre()+
53.             "\nDNI del cliente:"+getDNI()+
54.             "\nDireccion del cliente:"+getDireccion();
55.         return texto;
56.     }
57.}
58.package pp1_gr1_cristaisigue_erickvasquez;
59.import java.text.SimpleDateFormat;
60.import java.util.GregorianCalendar;
61.import javax.swing.JOptionPane;
62.
63.//SuperClase abstracta Recibo
64.public abstract class Recibo {
65.
66.    private String facturaTipo,NIS,mesAlPago;
67.    private Cliente cliente;//Agregacion del objeto Cliente
68.    private String fecha;
69.    private int dia,mes,year;
70.
71.    //Constructor vacio
72.    public Recibo() {
73.
74.    }
75.
76.
77.
78.    //Metodos SET y GETTER pertenecientes a esta clase para obtener y modificar atributos de esta clase
79.    public String getFacturaTipo() {
80.        return facturaTipo;
81.    }

```

```
82.

83.     public void setFacturaTipo(String facturaTipo) {
84.         this.facturaTipo = facturaTipo;
85.     }
86.

87.     public String getNIS() {
88.         return NIS;
89.     }
90.

91.     public void setNIS(String nIS) {
92.         NIS = nIS;
93.     }
94.

95.     public String getMesAlPago() {
96.         return mesAlPago;
97.     }
98.

99.     public void setMesAlPago(String mesAlPago) {
100.         this.mesAlPago = mesAlPago;
101.     }
102.

103.     public Cliente getCliente() {
104.         return cliente;
105.     }
106.
107.
108.
109.     public String getFecha() {
110.         return fecha;
111.     }
```

```
112.

113.     public void setFecha(String fecha) {
114.         this.fecha = fecha;
115.     }
116.

117.     public int getDia() {
118.         return dia;
119.     }
120.

121.     public void setDia(int dia) {
122.         this.dia = dia;
123.     }
124.

125.     public int getMes() {
126.         return mes;
127.     }
128.

129.     public void setMes(int mes) {
130.         this.mes = mes;
131.     }
132.

133.     public int getYear() {
134.         return year;
135.     }
136.

137.     public void setYear(int year) {
138.         this.year = year;
139.     }
140.
```

```

141.     public void setCliente(Cliente cliente) {
142.         this.cliente = cliente;
143.     }
144.
145.     /*El metodo "ingresarDatos" se brindaran Los datos para llenar Los
146.     atributos (facturaTipo,NIS,mesALPago
147.     *fecha,dia,mes,year) en esta clase y tambien se brindaran Los dato
148.     s pertenecientes a La agregacion de la
149.     *clase "Cliente cliente"
150.     *Este metodo sera sobrescrito dependiendo de la clase hija (aadi
151.     endo al existente metodo, La recepcion de valores dependiendo de Las clas
152.     es hijas)
153.     */
154.     public void ingresarDatos(String tipo,String meses[],SimpleDateFor
155.     mat sdf) {
156.         //Creacion del objeto cliente
157.         cliente=new Cliente();
158.         int mesAux=0;
159.
160.         cliente.darDatos(tipo);//Brindamos Los datos a la clase "Clien
161.         te" mediante su propio metodo definido
162.
163.         //Datos pertenecientes a esta clase
164.         setNIS(JOptionPane.showInputDialog(null,"Ingrese el NIS del se
165.         rvicio","Menu>Emitir>" + tipo,JOptionPane.INFORMATION_MESSAGE));
166.         setYear(Integer.parseInt(JOptionPane.showInputDialog(null,"Ing
167.         rese el año del pago Ejemplo: 2002","Menu>Emitir>" + tipo,JOptionPane.INFO
168.         RMATION_MESSAGE)));
169.         setDia(Integer.parseInt(JOptionPane.showInputDialog(null,"Ingr
170.         ese el dia del pago Ejemplo: 22","Menu>Emitir>" + tipo,JOptionPane.INFORMAT
171.         ION_MESSAGE)));
172.         setMes(Integer.parseInt(JOptionPane.showInputDialog(null,"Ingr
173.         ese el mes del pago Ejemplo: Mayo = 5","Menu>Emitir>" + tipo,JOptionPane.IN
174.         FORMATION_MESSAGE)));
175.         setFecha(sdf.format(new GregorianCalendar(year,mes-
176.         1,dia).getTime()));/*
177.         *Se utiliza el objeto GregorianCalendar para obtener una fecha
178.         con Los valores proporcionados y ademas
179.         *se Le da un formato con el parametro sdf(Simple date format)
180.         */
181.         setFacturaTipo(tipo);

```



```

167.         mesAux=Integer.parseInt(JOptionPane.showInputDialog(null,"Ingr
ese el mes al cobro: Mayo = 5","Menu>Emitir">"+tipo,JOptionPane.INFORMATIO
N_MESSAGE));
168.         setMesAlPago(meses[mesAux-
1]);//Se determina el mes al cobro, usando arreglo perteneciente a la interfaz "meses[]"
169.     }
170.
171.
172.     /*El metodo "desplegarInformacion" muestra los atributos perteneci
entes a esta clase (facturaTipo,NIS,mesAlPago
173.     *fecha,dia,mes,year) asi como tambien los atributos al objeto "Cli
ente cliente" por medio de su propio metodo
174.     *Este metodo sera sobrescrito segun la clase hija (añadiendo al e
xistente metodo, los atributos a desplegarse)
175.     */
176.     public String desplegarInformacion() {
177.         return "\n\tFactura por "+getFacturaTipo()+"\n"+"
\n"+cliente.b
rindarDatos()+"\nIdentificacion NIS:"+getNIS()
178.         +"\nFecha de pago:"+getFecha()+"\nMes al cobro:"+getMesAlPago(
)+"\n";
179.     }
180.
181.     public abstract double calcularPago();/*Metodo "calcularPago" abst
racto, para ser sobrescrito en las clases hijas,
182.     donde se determinaran el monto a pagar dependiendo de los criteri
os de cada clase hija*/
183.
184.
185. }
186.
187.
188. package pp1_gr1_cristaisigue_erickvasquez;
189. import java.text.SimpleDateFormat;
190. import javax.swing.JOptionPane;
191.
192. public class AYA extends Recibo{
193.
194.     //Atributos
195.     private final int costoHidrante=3525,alquilerMedidor=2500,cost
eDemora=2800,metroCubico=155;
196.     private double montoPorDemora,montoConsumoDiario;
197.     private double consumoDiario;
198.     private int diasDemora;
199.

```

```

200.
201.     public AYA(String tipo,String meses[],SimpleDateFormat sdf) {
202.         ingresarDatos(tipo,meses,sdf);//Llamada al metodo sobrescr
           ito "ingresarDatos" durante la creacion de este objeto
203.     }
204.
205.     //Metodos SET y GETTER pertenecientes a esta clase para obtene
           r y modificar atributos de esta clase
206.
207.     public double getConsumoDiario() {
208.         return consumoDiario;
209.     }
210.
211.
212.
213.
214.     public double getMontoConsumoDiario() {
215.         return montoConsumoDiario;
216.     }
217.
218.     public void setMontoConsumoDiario(double montoConsumoDiario) {
219.         this.montoConsumoDiario = montoConsumoDiario;
220.     }
221.
222.     public double getMontoPorDemora() {
223.         return montoPorDemora;
224.     }
225.
226.     public void setMontoPorDemora(double montoPorDemora) {
227.         this.montoPorDemora = montoPorDemora;
228.     }
229.
230.     public int getCosteDemora() {
231.         return costeDemora;
232.     }
233.
234.     public int getMetroCubico() {
235.         return metroCubico;
236.     }
237.
238.     public void setConsumoDiario(double consumoDiario) {
239.         this.consumoDiario = consumoDiario;
240.     }
241.

```

```

242.         public int getDiasDemora() {
243.             return diasDemora;
244.         }
245.
246.         public void setDiasDemora(int diasDemora) {
247.             this.diasDemora = diasDemora;
248.         }
249.
250.         public int getCostoHidrante() {
251.             return costoHidrante;
252.         }
253.
254.         public int getAlquilerMedidor() {
255.             return alquilerMedidor;
256.         }
257.
258.         /*Se sobrescribe el metodo "ingresarDatos" de la superclase "Recepcion"
259.         * se sobrescribe llamando al existente en la superclase
260.         * y añadiendole la recepcion de datos para los atributos de
261.         esta clase (montoPorDemora,
262.         * montoConsumoDiario, consumoDiario, diasDemora)
263.         */
264.         @Override
265.         public void ingresarDatos(String tipo, String meses[], SimpleDateFormat sdf) {
266.             super.ingresarDatos(tipo, meses, sdf); //Llamada para brindar
267.             los datos para los atributos de la superclase
268.
269.             //Añadida de la recepcion de datos para los atributos de esta clase
270.             setConsumoDiario(Double.parseDouble(JOptionPane.showInputDialog(null, "Ingrese el consumo en metros cubicos diario", "Menu>Emitir>Servicio AYA", JOptionPane.INFORMATION_MESSAGE)));
271.             setMontoConsumoDiario(getConsumoDiario()*30);
272.
273.             if(Integer.parseInt(JOptionPane.showInputDialog(null, "Hubo demora al pagar el servicio: SI=0 NO=1", "Menu>Emitir>Servicio AYA", JOptionPane.INFORMATION_MESSAGE))==0) {
274.                 setDiasDemora(Integer.parseInt(JOptionPane.showInputDialog(null, "Numero de dias de demora", "Menu>Emitir>Servicio AYA", JOptionPane.INFORMATION_MESSAGE)));
275.                 if(getDiasDemora()>0) {

```

```

275.             setMontoPorDemora(getCosteDemora()*getDiasDemora()
    );
276.         }else {
277.             setDiasDemora(0);
278.             setMontoPorDemora(0);
279.         }
280.     }else {
281.         setDiasDemora(0);
282.         setMontoPorDemora(0);
283.     }
284. }
285.
286.     /*Se sobrescribe el metodo "desplegarInformacion" de la superc
    lase "Recibo"
287.     * Se despliengan los atributos de la superclase y los pertene
    cientes a esta clase
288.     * */
289.     @Override
290.     public String desplegarInformacion() {
291.
292.         return super.desplegarInformacion()+"Consumo diario:"+getC
    onsumoDiario()+
293.             "\nNúmero de dias de demora:"+getDiasDemora()+"\nMonto a
    pagar por demora:"+getMontoPorDemora()+
294.             "\nMonto producto del consumo diario:"+getMontoConsumoDiar
    io()+"\nCosto por alquiler de hidrante:"+
295.             getCostoHidrante()+"\nCosto por alquiler de medido:"+getAl
    quilerMedidor();
296.     }
297.
298.     /*Se sobrescribe el metodo abstracto "calcularPago()" de la su
    perclase "Recibo"
299.     * usando el criterio brindado para el servicio de AYA
300.     * (consumo diario * 30 días + hidrante + alquiler) + (demora
    ).
301.     * ((getConsumoDiario()*30)+getCostoHidrante()+getAlquilerMedi
    dor())+(getDiasDemora()*getCosteDemora())
302.     * */
303.     @Override
304.     public double calcularPago() {
305.         double total=0;
306.
307.         if(getDiasDemora()>0) {
308.             setMontoPorDemora(getDiasDemora()*getCosteDemora());
309.             total+=getMontoPorDemora();

```

```

310.         }
311.
312.         setMontoConsumoDiario(getConsumoDiario()*30);
313.         total+=getMontoConsumoDiario();
314.
315.         total+=getAlquilerMedidor();
316.
317.         total+=getCostoHidrante();
318.
319.         return total;
320.     }
321.
322.
323. }
324.
325. package pp1_gr1_cristaisigue_erickvasquez;
326. import java.text.SimpleDateFormat;
327.
328. import javax.swing.JOptionPane;
329.
330. public abstract class Telefono extends Recibo{
331.
332.     //Atributos
333.     private double tarifaFija,minutosPlenos,minutosReducidos,costeMinu
tosP,costeMinutosR;
334.
335.     //Constructor vacio
336.     public Telefono() {
337.     }
338.
339.     //Metodos SET y GETTER pertenecientes a esta clase para obtener y
modificar atributos de esta clase
340.
341.     public double getTarifaFija() {
342.         return tarifaFija;
343.     }
344.
345.     public void setTarifaFija(double tarifaFija) {
346.         this.tarifaFija = tarifaFija;
347.     }
348.
349.     public double getMinutosPlenos() {
350.         return minutosPlenos;
351.     }
352.

```

```

353.     public void setMinutosPlenos(double minutosPlenos) {
354.         this.minutosPlenos = minutosPlenos;
355.     }
356.
357.     public double getMinutosReducidos() {
358.         return minutosReducidos;
359.     }
360.
361.     public void setMinutosReducidos(double minutosReducidos) {
362.         this.minutosReducidos = minutosReducidos;
363.     }
364.
365.
366.
367.     public double getCosteMinutosP() {
368.         return costeMinutosP;
369.     }
370.
371.     public void setCosteMinutosP(double costeMinutosP) {
372.         this.costeMinutosP = costeMinutosP;
373.     }
374.
375.     public double getCosteMinutosR() {
376.         return costeMinutosR;
377.     }
378.
379.     public void setCosteMinutosR(double costeMinutosR) {
380.         this.costeMinutosR = costeMinutosR;
381.     }
382.
383.     /*Se sobrescribe el metodo "ingresarDatos" de la superclase "Recib
o"
384.     * se sobrescribe llamando al existente en la superclase
385.     * y añadiendole la recepcion de datos para los atributos de esta
clase (tarifaFija,minutosPlenos,minutosReducidos, costeMinutosP, costeMinu
tosR)
386.     */
387.     @Override
388.     public void ingresarDatos(String tipo,String meses[],SimpleDateFor
mat sdf) {
389.         super.ingresarDatos(tipo,meses,sdf);
390.         setTarifaFija(Double.parseDouble(JOptionPane.showInputDialog(n
ull,"Ingrese el coste de la tarifa fija","Menu>Emitir>Telefono",JOptionPa
ne.INFORMATION_MESSAGE)));

```

```

391.         setMinutosPlenos(Double.parseDouble(JOptionPane.showInputDialog(
392.             null,"Ingrese la cantidad de minutos Plenos","Menu>Emitir>Telefono",JOptionPane.INFORMATION_MESSAGE)));
393.     }
394.
395.     /*Se sobrescribe el metodo abstracto "calcularPago()" de la superclase "Recibo"
396.     * esta modificacion sera representativa ya que esta clase a su vez posee herencia
397.     * */
398.     @Override
399.     public double calcularPago() {
400.         return 0;
401.     }
402.
403.     /*Se sobrescribe el metodo "desplegarInformacion" de la superclase "Recibo"
404.     * Se despiengan los atributos de la superclase y los pertenecientes a esta clase
405.     * */
406.     @Override
407.     public String desplegarInformacion() {
408.         return super.desplegarInformacion()+"Cantidad minutos reducidos:"+getMinutosReducidos()+"\nCantidad minutos plenos:"+getMinutosPlenos()+"\nTarifa fija:"+getTarifaFija()+"\nCosto del minuto pleno:"+getCosteMinutosP()+"\nCosto del minuto reducido:"+getCosteMinutosR();
409.     }
410. }
411. }
412.
413. package pp1_gr1_cristaisigue_erickvasquez;
414. import java.text.SimpleDateFormat;
415.
416. import javax.swing.JOptionPane;
417.
418. public class Celular extends Telefono{
419.
420.     //Atributos
421.     private double montoPR,montoPP,montoMensajes;
422.     private final double costoMensaje=9.25;
423.     private int cantidadMensajes;
424.

```

```

425.     public Celular(String tipo,String meses[],SimpleDateFormat sdf) {
426.         //Se le provee el coste minutos plenos y reducidos a la clase
         padre Telefono
427.         super.setCosteMinutosP(35);
428.         super.setCosteMinutosR(25);
429.         ingresarDatos(tipo,meses,sdf);//Llamada al metodo sobrescrito
         "ingresarDatos" durante la creacion de este objeto
430.     }
431.
432.
433.     //Metodos SET y GETTER pertenecientes a esta clase para obtener y
         modificar atributos de esta clase
434.     public double getMontoMensajes() {
435.         return montoMensajes;
436.     }
437.
438.
439.     public void setMontoMensajes(double montoMensajes) {
440.         this.montoMensajes = montoMensajes;
441.     }
442.
443.     public double getMontoPR() {
444.         return montoPR;
445.     }
446.
447.     public void setMontoPR(double montoPR) {
448.         this.montoPR = montoPR;
449.     }
450.
451.     public double getMontoPP() {
452.         return montoPP;
453.     }

```



```

454.     public void setMontoPP(double montoPP) {
455.         this.montoPP = montoPP;
456.     }
457.
458.     public int getCantidadMensajes() {
459.         return cantidadMensajes;
460.     }
461.
462.     public void setCantidadMensajes(int cantidadMensajes) {
463.         this.cantidadMensajes = cantidadMensajes;
464.     }
465.
466.     public double getCostoMensaje() {
467.         return costoMensaje;
468.     }
469.
470.     /*Se sobrescribe el metodo "ingresarDatos" de la superclase "Recib
o"
471.     * se sobrescribe llamando al existente en la superclase y la clas
e Telefono
472.     * y añadiendole la recepcion de datos para los atributos de esta
clase (montoPR,montoPP,montoMensajes,cantidadMensajes)
473.     */
474.     @Override
475.     public void ingresarDatos(String tipo,String meses[],SimpleDateFor
mat sdf) {
476.         super.ingresarDatos(tipo,meses,sdf);
477.         setCantidadMensajes(Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese La cantidad de mensajes","Menu>Emitir>Telefono>Telefono
Celular",JOptionPane.INFORMATION_MESSAGE)));
478.         setMontoPP(super.getCosteMinutosP()*super.getMinutosPlenos());
479.         setMontoPR(super.getCosteMinutosR()*super.getMinutosReducidos(
));

```

```

480.         setMontoMensajes(getCostoMensaje()*getCantidadMensajes());
481.     }
482.
483.     /*Se sobrescribe el metodo abstracto "calcularPago()" de la superclase "Recibo"
484.     * usando el criterio brindado para el servicio de Telefono Celular
485.     * Tarifa fija + ((Minutos plenos * Costo plenos)+ (Minutos reducidos * Costo reducidos) + (Mensajes * Costo mensaje)
486.     *getTarifaFija()+getMontoPP()+getMontoPR()+getMontoMensajes()
487.     * */
488.     @Override
489.     public double calcularPago() {
490.         double total=0;
491.         total+=super.getTarifaFija();
492.         total+=getMontoPP();
493.         total+=getMontoPR();
494.         total+=getMontoMensajes();
495.
496.         return total;
497.     }
498.
499.     /*Se sobrescribe el metodo "desplegarInformacion" de la superclase "Recibo"
500.     * Se despliengan los atributos de la superclase, la clase telefono y los pertenecientes a esta clase
501.     * */
502.     @Override
503.     public String desplegarInformacion() {
504.         return super.desplegarInformacion()+"\nCantidad de mensajes:"+
getCantidadMensajes()+"\nCosto por mensaje:"+getCostoMensaje()+
505.             "\nMonto por minutos plenos:"+getMontoPP()+"\nMonto por minutos reducidos:"+getMontoPR()
506.             +"\nMonto por mensajes:"+getMontoMensajes();
507.     }
508.
509. }
510.
511. package pp1_gr1_cristaisigue_erickvasquez;
512. import java.text.SimpleDateFormat;
513.
514. import javax.swing.JOptionPane;
515.
516. public class Residencial extends Telefono{
517.

```

```

518.     private double numeroImpulsos,roaming,productoMReducidos,productoM
        Plenos,productoImpulsos;
519.     private final int costoImpulso=75,tarifaInternacional=9100;
520.
521.     public Residencial(String tipo,String meses[],SimpleDateFormat sdf
        ) {
522.         //Se le provee el coste minutos plenos y reducidos a la clase
padre Telefono
523.         super.setCosteMinutosP(12);
524.         super.setCosteMinutosR(7);
525.         ingresarDatos(tipo,meses,sdf);//Llamada al metodo sobrescrito
"ingresarDatos" durante la creacion de este objeto
526.     }
527.
528.
529.
530.
531.     //Metodos SET y GETTER pertenecientes a esta clase para obtener y
modificar atributos de esta clase
532.     public double getProductoImpulsos() {
533.         return productoImpulsos;
534.     }
535.
536.     public void setProductoImpulsos(double productoImpulsos) {
537.         this.productoImpulsos = productoImpulsos;
538.     }
539.
540.     public double getNumeroImpulsos() {
541.         return numeroImpulsos;
542.     }
543.
544.
545.     public double getProductoMReducidos() {
546.         return productoMReducidos;
547.     }
548.
549.

```

```
550.     public void setProductoMReducidos(double productoMReducidos) {
551.         this.productoMReducidos = productoMReducidos;
552.     }
553.
554.     public double getProductoMPlenos() {
555.         return productoMPlenos;
556.     }
557.
558.     public void setProductoMPlenos(double productoMPlenos) {
559.         this.productoMPlenos = productoMPlenos;
560.     }
561.
562.     public void setNumeroImpulsos(double numeroImpulsos) {
563.         this.numeroImpulsos = numeroImpulsos;
564.     }
565.
566.     public double getRoaming() {
567.         return roaming;
568.     }
569.
570.     public void setRoaming(double roaming) {
571.         this.roaming = roaming;
572.     }
573.
574.     public int getTarifaInternacional() {
575.         return tarifaInternacional;
576.     }
577.
578.     public int getCostoImpulso() {
579.         return costoImpulso;
580.     }
```

```

581.
582.     /*Se sobrescribe el metodo "ingresarDatos" de la superclase "Recib
o"
583.     * se sobrescribe llamando al existente en la superclase y clase T
elefono
584.     * y añadiendole la recepcion de datos para los atributos de esta
clase (numeroImpulsos,roaming,productoMReducidos,productoMPlenos,product
oImpulsos)
585.     */
586.     @Override
587.     public void ingresarDatos(String tipo,String meses[],SimpleDateFor
mat sdf) {
588.         super.ingresarDatos(tipo,meses,sdf);
589.         setNumeroImpulsos(Double.parseDouble(JOptionPane.showInputDialog
(null,"Ingrese el n mero de impulsos","Menu>Emitir>Telefono>Telefono R
esidencial",JOptionPane.INFORMATION_MESSAGE)));
590.         setRoaming(Double.parseDouble(JOptionPane.showInputDialog(null
,"Ingrese el valor del Roaming","Menu>Emitir>Telefono>Telefono Residencia
l",JOptionPane.INFORMATION_MESSAGE)));
591.         setProductoMPlenos(super.getMinutosPlenos()*super.getCosteMinu
tosP());
592.         setProductoMReducidos(super.getCosteMinutosR()*super.getCosteM
inutosR());
593.         setProductoImpulsos(getNumeroImpulsos()*getCostoImpulso());
594.     }
595.
596.     /*Se sobrescribe el metodo abstracto "calcularPago()" de la superc
lase "Recibo"
597.     * usando el criterio brindado para el servicio de Telefono Reside
ncial
598.     * tarifa fija +(n  impulsos *costo de impulso)+monto de producto
minutos plenos+ monto producto de minutos reducidos + tarifa internacion
al + roaming
599.     * getTarifaFija()+getProductoImpulsos()+getProductoMPlenos()+getP
roductoMReducidos()+getTarifaInternacional()+getRoaming()
600.     * */
601.     @Override
602.     public double calcularPago() {
603.         double total=0;
604.         total+=getTarifaFija();
605.         total+=getProductoMPlenos();
606.         total+=getProductoMReducidos();
607.         total+=getRoaming();
608.         total+=getTarifaInternacional();
609.         total+=getProductoImpulsos();

```

```

610.
611.         return total;
612.     }
613.
614.     /*Se sobreescribe el metodo "desplegarInformacion" de la superclase
"Recibo"
615.     * Se despliengan los atributos de la superclase, la clase telefon
o y los pertenecientes a esta clase
616.     * */
617.     @Override
618.     public String desplegarInformacion() {
619.         return super.desplegarInformacion()+"\nValor de roaming:"+getR
oaming()+"\nNumero de impulsos:"+getNumeroImpulsos()
620.         +"\nMonto por minutos reducidos:"+getProductoMReducidos()+"\nM
onto por minutos plenos:"+getProductoMPlenos()
621.         +"\nMonto por numero de impulsos:"+getProductoImpulsos()+"\nMo
nto tarifa internacional:"+getTarifaInternacional();
622.     }
623. }
624.
625. package pp1_gr1_cristaisigue_erickvasquez;
626. import java.text.SimpleDateFormat;
627.
628. import javax.swing.JOptionPane;
629.
630. public class Electricidad extends Recibo{
631.
632.     //Atributos
633.     private final int costoReconexion=21000,costoAlumbradoPublico=950,
costoKilowat=1125;
634.     private final double impuestoBombero=0.025;
635.     private double KWS,montoKWS,montoImpuesto;
636.     private boolean Retraso;
637.
638.     public Electricidad(String tipo,String meses[],SimpleDateFormat sd
f) {
639.         ingresarDatos(tipo,meses,sdf);//Llamada al metodo sobreescrito
"ingresarDatos" durante la creacion de este objeto
640.     }
641.
642.     //Metodos SET y GETTER pertenecientes a esta clase para obtener y
modificar atributos de esta clase
643.     public boolean isRetraso() {
644.         return Retraso;
645.     }

```

```

646.
647.
648.
649.     public double getMontoKWS() {
650.         return montoKWS;
651.     }
652.
653.     public void setMontoKWS(double montoKWS) {
654.         this.montoKWS = montoKWS;
655.     }
656.
657.     public double getMontoImpuesto() {
658.         return montoImpuesto;
659.     }
660.
661.     public void setMontoImpuesto(double montoImpuesto) {
662.         this.montoImpuesto = montoImpuesto;
663.     }
664.
665.     public int getCostoReconexion() {
666.         return costoReconexion;
667.     }
668.
669.     public int getCostoAlumbradoPublico() {
670.         return costoAlumbradoPublico;
671.     }
672.
673.     public int getCostoKilowat() {
674.         return costoKilowat;
675.     }
676.
677.     public double getImpuestoBombero() {
678.         return impuestoBombero;
679.     }
680.
681.     public void setRetraso(boolean retraso) {
682.         Retraso = retraso;
683.     }
684.
685.     public double getKWS() {
686.         return KWS;
687.     }
688.
689.     public void setKWS(double kWS) {
690.         KWS = kWS;

```

```

691.     }
692.
693.
694.     /*Se sobrescribe el metodo "ingresarDatos" de la superclase "Recibo"
695.     * se sobrescribe llamando al existente en la superclase
696.     * y aadienle aadiendole la recepcion de datos para los atributos
697.     * KWS, montoKWS, montoImpuesto)
698.     */
699.     @Override
700.     public void ingresarDatos(String tipo, String meses[], SimpleDateFormat sdf) {
701.         super.ingresarDatos(tipo, meses, sdf); //Llamada para brindar los
702.         datos para los atributos de la superclase
703.         //Aadidura de la recepcion de datos para los atributos de esta
704.         clase
705.         setKWS(Double.parseDouble(JOptionPane.showInputDialog(null, "Ingrese la cantidad de KWS consumidos", "Menu>Emitir>Servicio ICE", JOptionPane.INFORMATION_MESSAGE)));
706.         setMontoKWS(getCostoKilowat()*getKWS());
707.         setMontoImpuesto(getMontoKWS()*getImpuestoBombero());
708.         if(Integer.parseInt(JOptionPane.showInputDialog(null, "¿Existi
709.         re retraso en el pago? :Ingrese 0 =SI , 1=NO", "Menu>Emitir>Servicio ICE",
710.         JOptionPane.INFORMATION_MESSAGE))==0) {
711.             setRetraso(true);
712.         }
713.     }
714.
715.     /*Se sobrescribe el metodo abstracto "calcularPago()" de la superclase "Recibo"
716.     * usando el criterio brindado para el servicio de Electrico ICE
717.     * (Kws*1125+alumbrado p blico + (Reconexi n (si atrasa))+ImpuestoBombero
718.     * (getMontoKWS()+getCostoAlumbradoPublico()+(True:21000)+(getMontoImpuesto()))
719.     * */
720.     @Override
721.     public double calcularPago() {
722.         double total=0;
723.

```



```

724.         if(isRetraso()) {
725.             total+=getCostoReconexion();
726.         }
727.
728.         total+=getCostoAlumbradoPublico();
729.
730.         total+=getMontoImpuesto();
731.
732.         total+=getMontoKWS();
733.
734.         return total;
735.     }
736.
737.     /*Se sobrescribe el metodo "desplegarInformacion" de la superclase
       "Recibo"
738.     * Se despiengan los atributos de la superclase y los pertenecien
       tes a esta clase
739.     * */
740.     @Override
741.     public String desplegarInformacion() {
742.         String retrasado;
743.
744.         if(isRetraso()) {
745.             retrasado="Si, si se le \n\t cobra la reconexion de (" +get
               CostoReconexion()+")";
746.         }else {
747.             retrasado="No, no se le \n\t cobra la reconexion de (" +ge
               tCostoReconexion()+")";
748.         }
749.
750.         return super.desplegarInformacion()+"Retraso en el pago:"+retr
               asado+"\nConsumo de KWS:"+getKWS()+"\nImpuesto bombero:"+getMontoImpuesto
               ()+
751.             "\nMonto producto del consumo de KWS:"+getMontoKWS()+"
               \nCobro por alumbrado publico:"+getCostoAlumbradoPublico()+"\nCosto de Ki
               loWatt individual:"+getCostoKilowatt();
752.     }
753. }
754. package pp1_gr1_cristaisigue_erickvasquez;
755. import java.text.SimpleDateFormat;
756. import javax.swing.JOptionPane;
757.
758. public class CableEInternet extends Recibo{
759.
760.     //Atributos

```

```

761.     private final double tarifaFija=15800,modem=10500;
762.     private double altaDefinicion,costePaquete,servicioDigital;
763.
764.     public CableEInternet(String tipo,String meses[],SimpleDateFormat
      sdf) {
765.         ingresarDatos(tipo,meses,sdf);//Llamada al metodo sobrescrito
      "ingresarDatos" durante la creacion de este objeto
766.     }
767.
768.     //Metodos SET y GETTER pertenecientes a esta clase para obtener y
      modificar atributos de esta clase
769.     public double getTarifaFija() {
770.         return tarifaFija;
771.     }
772.
773.     public double getModem() {
774.         return modem;
775.     }
776.
777.     public double getAltaDefinicion() {
778.         return altaDefinicion;
779.     }
780.
781.     public void setAltaDefinicion(double altaDefinicion) {
782.         this.altaDefinicion = altaDefinicion;
783.     }
784.
785.     public double getCostePaquete() {
786.         return costePaquete;
787.     }
788.

```

```

789.     public void setCostePaquete(double costePaquete) {
790.         this.costePaquete = costePaquete;
791.     }
792.
793.     public double getServicioDigital() {
794.         return servicioDigital;
795.     }
796.
797.     public void setServicioDigital(double servicioDigital) {
798.         this.servicioDigital = servicioDigital;
799.     }
800.
801.     /*Se sobrescribe el metodo "ingresarDatos" de la superclase "Recib
o"
802.     * se sobrescribe llamando al existente en la superclase
803.     * y adaptandole la recepcion de datos para los atributos de esta
clase (altaDefinicion, costePaquete, servicioDigital)
804.     */
805.
806.     @Override
807.     public void ingresarDatos(String tipo, String meses[], SimpleDateFormat
mat sdf) {
808.         super.ingresarDatos(tipo, meses, sdf);
809.         setServicioDigital(Double.parseDouble(JOptionPane.showInputDialog(
log(null, "Ingrese el coste del servicio digital",
810.             "menu>Emitir>Servicio Cable TV e Intenet", JOptionPane.
INFORMATION_MESSAGE))));
811.
812.         setAltaDefinicion(Double.parseDouble(JOptionPane.showInputDialog(
og(null, "Ingrese el coste por alta definicion",
813.             "menu>Emitir>Servicio Cable TV e Intenet", JOptionPane.
INFORMATION_MESSAGE))));
814.
815.         escogerPaquete(); //Llamada del metodo "escogerPaquete()" perte
reciente a esta clase
816.     }
817.
818.     /*Como parte del ingreso de datos para esta clase se necesita que
usuario ingrese el paquete suscrito
819.     (PKT)*/
820.     public void escogerPaquete() {

```

```

821.
822.         int opcion=Integer.parseInt(JOptionPane.showInputDialog(null,"
      Ingrese el paquete suscrito:"
823.             + "\n1- Paquete 50GB (30000 colones)+"\n2-
      Paquete 10GB (12000 colones)"
824.             +"\n3-Paquete 4GB (6000 colones)"));
825.
826.         switch(opcion) {
827.
828.             case 1:{
829.                 setCostePaquete(30000);
830.             }break;
831.
832.             case 2:{
833.                 setCostePaquete(12000);
834.             }break;
835.
836.             case 3:{
837.                 setCostePaquete(6000);
838.             }break;
839.
840.             default:{
841.                 JOptionPane.showMessageDialog(null, "Opcion invalida R
      e-Intente el proceso","Menu>Emitir>Servicio Cable TV e Intenet",
842.                     JOptionPane.WARNING_MESSAGE);
843.                 escogerPaquete();
844.             }break;
845.
846.         }
847.
848.     }
849.
850.
851.     /*Se sobrescribe el metodo abstracto "calcularPago()" de la superc
      lase "Recibo"
852.         * usando el criterio brindado para el servicio de Cable E interne
      t
853.         * (Tarifa fija + Alta definicion + Servicio digital + Cable mode
      m+ Pkt de Internet).
854.         * (getTarifaFija()+getAltaDefinicion()+getServicioDigital()+getMo
      dem()+getCostePaquete())
855.         * */
856.     @Override
857.     public double calcularPago(){
858.

```

```

859.         double total=0;
860.         total+=getModem();
861.         total+=getTarifaFija();
862.         total+=getCostePaquete();
863.         total+=getServicioDigital();
864.         total+=getAltaDefinicion();
865.
866.         return total;
867.     }
868.
869.
870.     /*Se sobreescribe el metodo "desplegarInformacion" de la superclase
"Recibo"
871.     * Se despliengan los atributos de la superclase y los pertenecien
tes a esta clase
872.     * */
873.     @Override
874.     public String desplegarInformacion() {
875.         return super.desplegarInformacion()+"Coste de alta definicion:
            "+getAltaDefinicion()+"\nCoste del paquete elegido:"+
876.             getCostePaquete()+"\nCoste de servicio digital:"+getServicioDi
            gital()+"\nTarifa fija:"+getTarifaFija()
877.             +"\nCoste cable modem:"+getModem();
878.     }
879.
880. }
881.
882. package pp1_gr1_cristaisigue_erickvasquez;
883. import java.text.SimpleDateFormat; //Empleo de Simple date format para
obtener un formato
884. import javax.swing.JOptionPane; //JOptionPane para posibilitar las inte
racciones entre el sistema y el usuario
885.
886. public class Interfaz {
887.
888.     static final String meses[] = {"Enero", "Febrero", "Marzo", "Abril", "M
        ayo", "Junio", "Julio", "agosto", "septiembre", "octubre", "noviembre", "diciemb
        re"};
889.     static final SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy
        y"); //Simple date format (sdf) para dar formato a las fechas.
890.
891.     public static void main(String[] args) {
892.
893.
894.         Recibo recibo[]=new Recibo[15];

```

```

895.         boolean condicion=true; //Condicion inicial para empleada en los
           ciclos de menu y submenu
896.         int opcion, capacidad=0; //Capacidad indica donde se guardara el
           siguiente recibo a emitir
897.         //el uso de la variable opcion nos permitira escoger entre los
           casos del menu principal y submenu
898.
899.         //El siguiente ciclo representa al Menu principal
900.         //Cada caso representa una opcion del total de 5
901.         while(condicion) {
902.
903.             /*Se emplean controladores de excepciones "try,catch" esto
           para evitar la desarmonia del sistema
904.             * al obtener algun valor brindado por el usuario
905.             */
906.             try {
907.                 opcion=0;
908.                 opcion=Integer.parseInt(JOptionPane.showInputDialog(nu
           11,"Ingrese una opcion:\n1-Emitir recibo"+
909.                 "\n2-Visualizar recibos"+ "\n3-
           Mostrar recibo por NIS especifico"+ "\n4-Mostrar recibo con el\n\t"
910.                 + "Monto total a pagar mas alto\n5-Salir"
911.                 , "Menu", JOptionPane.INFORMATION_MESSAGE));
912.
913.                 switch(opcion) {
914.
915.                     case 1:{
916.                         //Antes de tratar de emitir un recibo verifica
           mos si la capacidad no ha llegado a su limite
917.                         if(capacidad==14) {
918.                             //Se ha llegado al limite de 15 objetos en
           el arreglo "recibo[]"
919.                             JOptionPane.showMessageDialog(null,"Se ha
           Llegado a la maxima capacidad para emitir recibos","Aviso",JOptionPane.WA
           RNING_MESSAGE);
920.                         }else {
921.
922.
923.                             /*El siguiente ciclo representa al Sub Men
           u para emitir recibos
924.                             *Cada caso representa un tipo (servicio di
           stinto) de recibo a emitir
925.                             *Al emitirse correctamente el servicio esc
           ogido se guardara en el arreglo "recibos[]" con la poscion del anterior v
           alor de la variable"capacidad"

```

```

926.         *La variable capacidad aumentara un valor,
           para indicar el indice donde se guardara el proximo servicio a emitirse
927.         *se mostrara Los respectivos datos ingresa
           dos y calculos del recibo recién emitido, el monto total a pagar se calcul
           a y se despliega mediante el metodo sobrescrito
928.         *"calcularPago()" perteneciente a la super
           clase "Recibo", al ser sobrescrita por los criterios dependiendo del ser
           vicio seleccionado.
929.         *y se volvera al menu principal al cambiar
           el valor de la variable "boolean condicion"
930.         */
931.         while(condicion) {
932.
933.             //Empleo de control de excepciones "try
           y catch"
934.             try {
935.                 opcion=Integer.parseInt(JOptionPane
           e.showInputDialog(null,"Ingresa un servicio:\n1-Servicio agua AYA"+
936.                 "\n2-
           Servicio Electrico ICE"+ "\n3-Servicio Telefonico Residencial"+ "\n4-
           Servicio Telefonico Celular"
937.                 + "\n5-
           Servicio de Cable TV e Internet\n6-Volver al menu"
           ,"Menu>Emitir",JOptionPane
938.             .INFORMATION_MESSAGE));
939.
940.             switch(opcion) {
941.                 case 1:{
942.                     //Caso en el submenu para
           emitir un recibo por Servicio AYA
943.                     AYA servicioTemporal=new A
           YA("Servicio AYA",meses,sdf);
944.                     recibo[capacidad]=servicio
           Temporal;
945.                     JOptionPane.showMessageDialog
           (null,recibo[capacidad].desplegarInformacion()+"\nMonto total a pagar:
           "+recibo[capacidad].calcularPago(),"Factura emitida",JOptionPane.INFORMAT
           ION_MESSAGE);
946.                     capacidad++;
947.                     condicion=false;
948.                 }break;
949.
950.
951.                 case 2:{

```

```

952.                                     //Caso en el submenu para
emitir un recibo por Servicio Electrico ICE
953.                                     Electricidad temporalServi
cio=new Electricidad("Servicio Electrico ICE",meses,sdf);
954.                                     recibo[capacidad]=temporal
Servicio;
955.                                     JOptionPane.showMessageDialog
log(null,recibo[capacidad].desplegarInformacion()+"\nMonto total a pagar:
"+recibo[capacidad].calcularPago(),"Factura emitida",JOptionPane.INFORMAT
ION_MESSAGE);
956.                                     capacidad++;
957.                                     condicion=false;
958.                                     }break;
959.
960.                                     case 3:{
961.                                     //Caso en el submenu para
emitir un recibo por Servicio Telefono Residencial
962.                                     Residencial servicioTempor
al=new Residencial("Servicio Telefonico Residencial",meses,sdf);
963.                                     recibo[capacidad]=servicio
Temporal;
964.                                     JOptionPane.showMessageDialog
log(null,recibo[capacidad].desplegarInformacion()+"\nMonto total a pagar:
"+recibo[capacidad].calcularPago(),"Factura emitida",JOptionPane.INFORMAT
ION_MESSAGE);
965.                                     capacidad++;
966.                                     condicion=false;
967.
968.                                     }break;
969.
970.                                     case 4:{
971.                                     //Caso en el submenu para
emitir un recibo por Servicio Telefono Celular
972.                                     Celular servicioTemporal=n
ew Celular("Servicio Telefono Celular",meses,sdf);
973.                                     recibo[capacidad]=servicio
Temporal;
974.                                     JOptionPane.showMessageDialog
log(null,recibo[capacidad].desplegarInformacion()+"\nMonto total a pagar:
"+recibo[capacidad].calcularPago(),"Factura emitida",JOptionPane.INFORMAT
ION_MESSAGE);
975.                                     capacidad++;
976.                                     condicion=false;
977.                                     }break;

```



```

978.
979.             case 5:{
980.                 //Caso en el submenu para
emitir un recibo por Servicio Cable E Internet
981.                 CableEInternet servicioTem
poral=new CableEInternet("Servicio Cable e Internet",meses,sdf);
982.
983.                 recibo[capacidad]=servicio
Temporal;
984.                 JOptionPane.showMessageDialog
log(null,recibo[capacidad].desplegarInformacion()+"\nMonto total a pagar:
"+recibo[capacidad].calcularPago(),"Factura emitida",JOptionPane.INFORMAT
ION_MESSAGE);
985.                 capacidad++;
986.                 condicion=false;
987.
988.                 }break;
989.
990.             case 6:{
991.                 //Caso en el submenu para
volver al menu principal
992.                 condicion=false;
993.                 }break;
994.
995.             default:{
996.                 JOptionPane.showMessageDialog
log(null,"Opcion Invalida","Aviso",JOptionPane.WARNING_MESSAGE);
997.                 }break;
998.
999.             }
1000.
1001.             }catch(Exception e) {
1002.                 JOptionPane.showMessageDialog(null
,"Entrada Invalida","Aviso",JOptionPane.WARNING_MESSAGE);
1003.             }
1004.
1005.         }
1006.
1007.         condicion=true;
1008.     }
1009. }break;
1010.
1011.     case 2:{
1012.         //Se muestran todos los datos de cada objeto p
erteneciente al objeto recibo

```

```

1013.          //Se verifica si se ha emitido al menos alguno
1014.          //Si esto es asi, se recorre el arreglo y para
          cada posicion se utiliza el metodo
1015.          //designado para mostrar los datos "desplegarI
          nformacion()"
1016.          if(recibo[0]!=null) {
1017.              for(int cont=0;cont<recibo.length;cont++)
          {
1018.                  if(recibo[cont]!=null) {
1019.                      JOptionPane.showMessageDialog(null
          , recibo[cont].desplegarInformacion()+"\nMonto a a pagar:"+recibo[cont].c
          alcularPago(),"Menu>Visualizar todo ("+(cont+1)+")",JOptionPane.INFORMATI
          ON_MESSAGE);
1020.                  }
1021.              }
1022.          }else {
1023.              JOptionPane.showMessageDialog(null, "No se
          han emitido recibos todavia","Menu>Visualizar todo",JOptionPane.WARNING_
          MESSAGE);
1024.          }
1025.          }break;
1026.
1027.
1028.          case 3:{
1029.
1030.              /*Se verifica si se ha emitido algun recibo
1031.              Se obtiene el atributo NIS de cada objeto in
          cluido en el arreglo de recibo
1032.              Este se compara con el valor dado, y se mues
          tra si hay coincidencia
1033.              Si es asi lo muestra*/
1034.
1035.              if(recibo[0]!=null) {
1036.                  String aBuscar=JOptionPane.showInputDialog
          (null,"Ingrese el NIS a buscar:", "Menu>NIS Especifico",JOptionPane.INFORM
          ATION_MESSAGE);
1037.                  int aux=-5;
1038.                  for(int cont=0;cont<recibo.length;cont++)
          {
1039.                      if(recibo[cont]!=null) {
1040.                          if(recibo[cont].getNIS().equalsIgn
          oreCase(aBuscar)) {
1041.                              aux=cont;
1042.                          }
1043.                      }

```

```

1044.         }
1045.
1046.         //Si se encontro algun recibo concordante
con el NIS se cumple la condicion de lo contrario se informa.
1047.         if(aux>=0) {
1048.             JOptionPane.showMessageDialog(null,"Se
encontro:\n"+recibo[aux].desplegarInformacion()+"\nMonto total a pagar"+
recibo[aux].calcularPago(),"Menu>NIS Especifico",JOptionPane.INFORMATION_
MESSAGE);
1049.         }else {
1050.             JOptionPane.showMessageDialog(null,"No
se encontro ningun recibo con el NIS "+aBuscar,"Menu>NIS Especifico",JOp
tionPane.INFORMATION_MESSAGE);
1051.         }
1052.         }else {
1053.             JOptionPane.showMessageDialog(null, "No se
han emitido recibos todavia","Menu>Visualizar todo",JOptionPane.WARNING_
MESSAGE);
1054.         }
1055.         }break;
1056.
1057.
1058.         case 4:{
1059.             /*El caso cuatro en el menu principal, concist
e en encontrar la factura con el
1060.             * valor mas alto a pagar, para esto primerame
nente se verifica si se ha emitido algun recibo
1061.             * Seguidamente suponemos que recibo con el ma
yor valor a pagar se encuentra en la posicion 0
1062.             * como esta inicializado en la variable "int
major", iniciamos a recorrer el arreglo
1063.             * entonces si la poscion representada en la v
ariable inicialiazada "major" es menor
1064.             * a la posicion X (cont) en el ciclo for , "m
ajor" se le asignara el valor de X (cont) y asi
1065.             * sucesivamente hasta terminar de recorrer el
arreglo
1066.             * */
1067.             if(recibo[0]!=null) {
1068.                 int major=0;
1069.
1070.                 for(int cont=1;cont<recibo.length;cont++)
                {
1071.                     if(recibo[cont]!=null) {

```

```

1072.                                     if(recibo[cont].calcularPago()>rec
    ibo[major].calcularPago()) {
1073.                                     major=cont;
1074.                                     }
1075.                                 }
1076.                            }
1077.
1078.                                //Se imprime el recibo con la posicion "ma
    jor" que se determino en el aterior ciclo for
1079.                                JOptionPane.showMessageDialog(null, "Recib
    o con el monto mas alto a pagar"+recibo[major].desplegarInformacion()+"\n
    Monto total a pagar:"+recibo[major].calcularPago(),
1080.                                "Menu>Recibo, monto mas alto",JOpt
    ionPane.INFORMATION_MESSAGE);
1081.                                }else {
1082.                                JOptionPane.showMessageDialog(null, "No se
    han emitido recibos todavia","Menu>Recibo, monto mas alto",JOptionPane.W
    ARNING_MESSAGE);
1083.                                }
1084.
1085.                                }break;
1086.
1087.
1088.                                case 5:{
1089.                                //EL caso n mero cinco del menu principal rep
    resenta la opcion de salir
1090.                                System.exit(0);
1091.                                }break;
1092.
1093.                                default:{
1094.                                //Se indica que la opcion no existe
1095.                                JOptionPane.showMessageDialog(null,"Opcion Inv
    alida","Aviso",JOptionPane.WARNING_MESSAGE);
1096.                                }break;
1097.
1098.                                }
1099.
1100.                                }catch(Exception er) {
1101.                                JOptionPane.showMessageDialog(null,"Entrada Invali
    da","Aviso",JOptionPane.WARNING_MESSAGE);
1102.                                }
1103.
1104.                                }
1105.
1106.                                }

```

```
1107.  
1108.  
1109. }  
1110.
```

8. Problemas y Limitaciones

8.1 Problema: No se presentaron problemas a la hora de ejecutar el programa.

8.2 Limitaciones: En lo que respecta el ingreso de datos en los diferentes servicios si estos son valores numéricos se deberán ingresar valores positivos, de lo contrario la realización de cálculos será errada.

9. Conclusión

El trabajo realizado determina que si bien algunos de los funcionamientos del código del programa, tales como la descripción del funcionamiento del programa y la descripción de los datos para un mayor entendimiento. A la vez, se comprende el problema que presentaba la recaudadora MENOSPAG S.A, en el departamento de despacho del cliente, el cual era la ineficacia a la hora de los funcionarios poder solicitar la información correspondiente de cada usuario. Se planteo una solución de índole entendible y muy descriptivo con la finalidad de comprender el programa para cualquier funcionario de la recaudadora. Asimismo, se trató de realizar comentarios claros para que cualquier programador entendiera las funciones que realizaba los algoritmos impuestos en el código, si bien se trató de comentar exclusivamente aquellos métodos o ciclos que fueran confusos a simple vista. No obstante, esto no agobia que algún programador no pueda comprenderlo, pero si bien esta diseñado con el fin de ser fácilmente comprensible.

Al mismo tiempo, tiene una interfaz sencilla y legible para que el funcionario comprende. El programa cuenta con una serie de instrucciones, la cual cumple la necesidad de poder evacuar dudas sencillas hacia el funcionario que esté utilizando el programa. En estas instrucciones se especifica con claridad que elementos deberá ingresar en cada opción.

Con respecto al trabajo, se pudo abarcar cada uno de los puntos de los Objetivos satisfactoriamente, creando un programa ordenado y de forma muy descriptiva para cualquier persona que lo manipule.