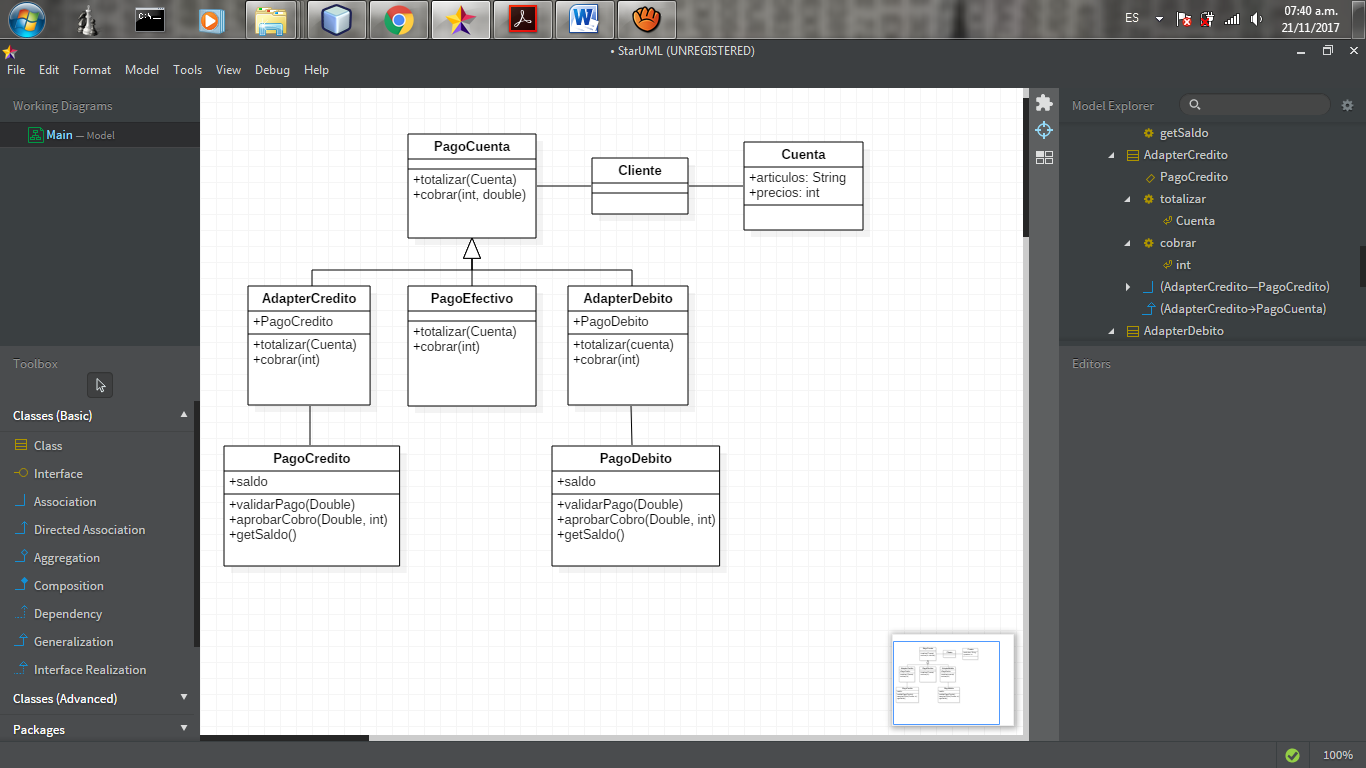
**ESCENARIO:**

Planteamos un escenario que sucedería dentro de un restaurante, la situación problema se presenta para el manejo de los diferentes tipos de pago (Efectivo, Tarjeta De Crédito y Tarjeta Debito), de manera tal que al momento de pagar con tarjetas incluya un aumento y valide las cuotas a diferir el pago.



public class Cliente {

public static void main(String[] args) {

PagoCuenta pagoCuenta;

Cuenta cuenta;

int opcion, cantidad, cuotas;

String producto, productos[];

int salir = 0;

double total, precio, precios[], saldo = 0;

while (salir != 1) {

System.out.println("-------------------------");

System.out.println("BIENVENIDO AL RESTAURANTE");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

cantidad = Integer.parseInt(JOptionPane.showInputDialog(null, "Pago De Cuenta \n"

+ "Digite cantidad de productos: \n"));

productos = new String[cantidad];

precios = new double[cantidad];

for (int i = 0; i < cantidad; i++) {

producto = JOptionPane.showInputDialog(null, "Nombre Del Producto: ");

precio = Integer.parseInt(JOptionPane.showInputDialog(null, "Precio: "));

productos[i] = producto;

precios[i] = precio;

}

cuenta = new Cuenta(productos, precios);

boolean cancelado = false;

do {

opcion = Integer.parseInt(JOptionPane.showInputDialog(null, "Tipo DE Pago: " + "\n"

+ "1.Efectivo" + "\n"

+ "2.Tarjeta De Credito" + "\n"

+ "3.Tarjeta Debito" + "\n"

+ "4.Salir" + "\n"));

switch (opcion) {

case 1:

try {

saldo = Integer.parseInt(JOptionPane.showInputDialog(null, "Efectivo: "));

pagoCuenta = new PagoEfectivo(saldo);

total = pagoCuenta.totalizar(cuenta);

pagoCuenta.cobrar(total, 1);

cuenta.imprimirCuenta();

cancelado = true;

System.out.println("Efectivo:" + saldo);

System.out.println("Cambio:" + pagoCuenta.getEstado());

} catch (Error e) {

JOptionPane.showMessageDialog(null, "Intentelo De Nuevo... Error:" + e);

cancelado = false;

}

break;

case 2:

try {

saldo = Integer.parseInt(JOptionPane.showInputDialog(null, "Saldo: "));

pagoCuenta = new AdapterCredito(saldo);

total = pagoCuenta.totalizar(cuenta);

cuotas = Integer.parseInt(JOptionPane.showInputDialog(null, "# Cuotas: "));

pagoCuenta.cobrar(total, cuotas);

cuenta.imprimirCuenta();

cancelado = true;

System.out.println("Nuevo Saldo:" + pagoCuenta.getEstado());

} catch (Error e) {

JOptionPane.showMessageDialog(null, "Intentelo De Nuevo... Error:" + e);

cancelado = false;

}

break;

case 3:

try {

saldo = Integer.parseInt(JOptionPane.showInputDialog(null, "Saldo: "));

pagoCuenta = new AdapterDebito(saldo);

total = pagoCuenta.totalizar(cuenta);

pagoCuenta.cobrar(total, 1);

cuenta.imprimirCuenta();

cancelado = true;

System.out.println("Nuevo Saldo:" + pagoCuenta.getEstado());

} catch (Error e) {

JOptionPane.showMessageDialog(null, "Intentelo De Nuevo... Error:" + e);

cancelado = false;

}

break;

case 4:

salir = 1;

break;

}

} while (cancelado == false);

}

}

}

public class Cuenta {

public String[] articulos;

public double[] precios;

public Cuenta(String[] articulos, double[] precios) {

this.articulos = articulos;

this.precios = precios;

}

public String[] getArticulos() {

return articulos;

}

public void setArticulos(String[] articulos) {

this.articulos = articulos;

}

public double[] getPrecios() {

return precios;

}

public void setPrecios(double[] precios) {

this.precios = precios;

}

public void imprimirCuenta(){

for (int i = 0; i < articulos.length; i++) {

System.out.println(articulos[i]+" ... "+precios[i]);

}

}

}

public abstract class PagoCuenta {

abstract public double totalizar(Cuenta cuenta);

abstract public void cobrar(double totalizado, int cotas);

abstract public double getEstado();

}

public class AdapterCredito extends PagoCuenta {

private final PagoTCredito tCredito;

public AdapterCredito(double saldo) {

super();

tCredito = new PagoTCredito(saldo);

}

@Override

public double totalizar(Cuenta cuenta) {

double total = 0;

for(int i=0; i<cuenta.articulos.length; i++ ){

total=total+(cuenta.precios[i]);

}

if(tCredito.validarPago(total)==true){

total=total+(total\*0.05);

}else{

total=0;

}

return total;

}

@Override

public void cobrar(double totalizado,int cuotas) {

tCredito.aprobarCobro(totalizado,cuotas);

}

@Override

public double getEstado() {

return tCredito.getSaldo();

}

}

public class PagoTCredito {

public double saldo;

public PagoTCredito(double saldo) {

this.saldo = saldo;

}

public boolean validarPago(Double total) {

boolean valido = false;

if (saldo > (total + (total \* 0.05))) {

valido = true;

}

return valido;

}

public void aprobarCobro(Double cobro, int cuotas) {

setSaldo(saldo - (saldo / cuotas));

}

public double getSaldo() {

return saldo;

}

public void setSaldo(double saldo) {

this.saldo = saldo;

}

}

public class AdapterDebito extends PagoCuenta {

private final PagoTDebito tDebito;

public AdapterDebito(double saldo) {

super();

tDebito = new PagoTDebito(saldo);

}

@Override

public double totalizar(Cuenta cuenta) {

double total = 0;

for (int i = 0; i < cuenta.articulos.length; i++) {

total = total + (cuenta.precios[i]);

}

if (tDebito.validarPago(total) == true) {

total = total + (total \* 0.03);

} else {

total = 0;

}

return total;

}

@Override

public void cobrar(double totalizado, int cuotas) {

tDebito.aprobarCobro(totalizado, cuotas);

}

@Override

public double getEstado() {

return tDebito.getSaldo();

}

}

public class PagoTDebito {

private double saldo;

public PagoTDebito(double saldo) {

this.saldo = saldo;

}

public boolean validarPago(Double total) {

boolean valido = false;

if (saldo > (total + (total \* 0.03))) {

valido = true;

}

return valido;

}

public void aprobarCobro(Double totalizado, int cuotas) {

setSaldo(saldo - totalizado);

}

public double getSaldo() {

return saldo;

}

public void setSaldo(double saldo) {

this.saldo = saldo;

}

}

public class PagoEfectivo extends PagoCuenta {

public double efectivo;

public PagoEfectivo(double efectivo) {

this.efectivo = efectivo;

}

@Override

public double totalizar(Cuenta cuenta) {

double total = 0;

for (int i = 0; i < cuenta.articulos.length; i++) {

total = total + (cuenta.precios[i]);

}

return total;

}

@Override

public void cobrar(double totalizado, int cuotas) {

if (efectivo >= totalizado) {

setEfectivo((efectivo - totalizado) / cuotas);

}

}

public void setEfectivo(double efectivo) {

this.efectivo = efectivo;

}

public double getEfectivo() {

return efectivo;

}

@Override

public double getEstado() {

return getEfectivo();

}

}

RESULTADOS:

