*2doCienciade la Computación*

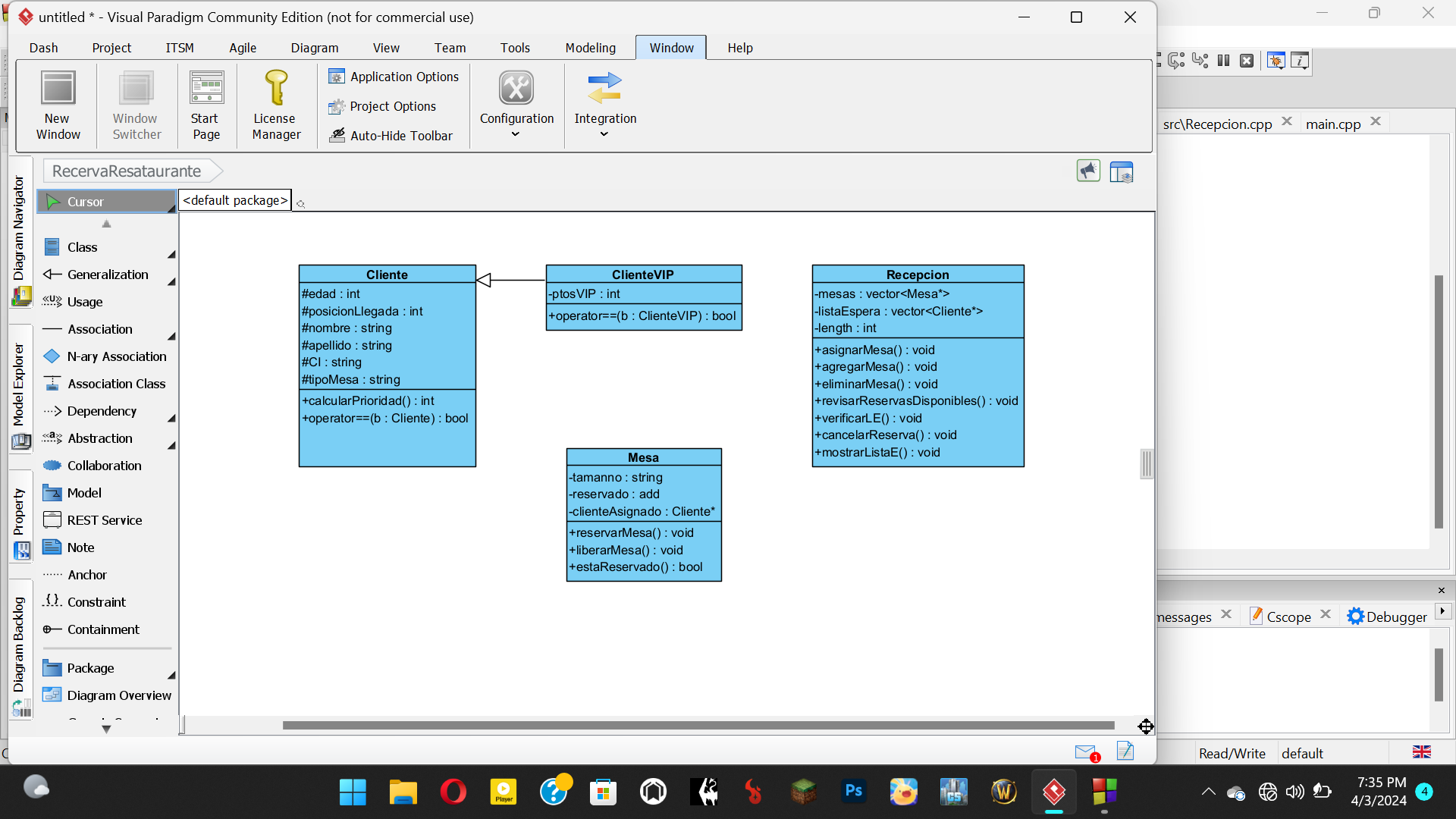


***Proyecto de Programación***

***C++***

**Autor**: *Kamila Veloso Estupiñán*

*UCLV: Marta Abreu de las Villas*



1,m

1,m

UML

**Problemática:**

Se desea un sistema que permita administrar la asignación de mesas a clientes en un restaurante. Las mesas podrán ser reservadas a nombre de un cliente y estas tienen diferentes clasificaciones: personales, normales y grandes. Cuando un cliente se pone en contacto con el restaurante primeramente este es agregado a la lista de espera y posteriormente podrá adquirir una reserva. Hay diferentes tipos de clientes, existen los regulares y los VIP; su diferencia fundamental es que los clientesVIP tienen cierta prioridad cuando entran a la lista de espera, esta prioridad se ve modificada por la cantidad de puntos VIP que posea. Cada cliente quiere reservar un tipo de mesa (personal, normal o grande) y esto debe tenerse en cuenta al hacer una reservación. El Sistema debe ser capaz de introducir nuevos clientes a la lista de espera, hacer reservaciones, eliminar reservaciones, mostrar disponibilidad y confirmar que un cliente dado esta en la lista de espera.

**Respuesta:**

El código proporciona una implementación de las clases **Cliente, ClienteVIP, Mesa,** y **Recepción.** La clase **Cliente** tiene una subclase **ClienteVIP** para proporcionar funcionalidades específicas para clientes VIP, incluyendo un cálculo de prioridad diferente. La clase **Mesa** modela una mesa en el establecimiento, con métodos para reservar y liberar mesas, así como para verificar su disponibilidad. Por último, la clase **Recepción** se encarga de gestionar las mesas y la lista de espera de clientes, proporcionando métodos para asignar mesas, agregar clientes a la lista de espera, eliminar reservas y revisar las reservas disponibles.

En la clase Cliente se definen las características (atributos) de un cliente (nombre, apellido, edad, tipo de mesa, carnet de identidad y posición de llegada) además sus funcionalidades: la función virtual que calcula la prioridad, en este caso solo devuelve el valor de la posición de llegada y que luego se sobrescribe en ClienteVIP donde este valor se ve modificado por la cantidad de puntos VIP.

class Cliente{ class ClienteVIP : public Cliente {

protected: private:

string nombre; int ptosVIP;

string apellido; …};

int edad;

string CI;

string tipoMesa;

int posicionLlegada…};

Método de sobrecarga de operadores:

bool Cliente::operator ==(Cliente b){

return this->CI==b.getCI();

}

Con la utilización de este método es posible comparar clientes de forma directa.

En la clase Mesa se define los atributos: tamaño (aquí se especifica si la mesa es personal, normal o grande) además tiene un bool que especifica si la mesa está reservada o no y un puntero a cliente que apunta al cliente que la reservo. El constructor de mesa tiene un solo argumento puesto que al crearse nunca están reservadas y el puntero de cliente apunta a null. La función reservarMesa permite reservar la mesa asignando un cliente a la misma. La función liberarMesa cambia a false el bool que indica si la mesa está reservada o no y el atributo clienteReservado apunta nuevamente a null.

class Mesa {

private:

string tamano;

bool reservado;

Cliente\* clienteAsignado;

…};

La clase Recepción es aquella que contiene la lista de mesas con que cuenta el restaurante y maneja la lista de espera de los clientes. Esta tiene varias funcionalidades entre ellas la de agregar a un cliente a la lista de espera. Dentro de esta función también se organiza la lista en función a la prioridad de cada cliente. Para realizar esta operación se pasa un puntero a cliente. Otra de sus funciones es asignarMesa la cual recorre las mesas y a cada mesa vacía le asigna el primer cliente que sea compatible siempre y cuando no se agoten las mesas o se vacíe la lista de espera. Esta clase también permite eliminar todas las reservas realizadas, liberando todas las mesas. También es posible cancelar una reserve introduciendo el carnet de identidad del cliente a nombre del cual fue realizada. La clase reserve también da la opción de ver la disponibilidad de las mesas y de mostrar la lista de espera de los clientes. Otra función que también pose es la de verificar si un cliente dado esta en la lista de espera.

class Recepcion {

private:

vector<Mesa\*> mesas;

vector<Cliente\*>listaEspera;

int length;

…};

Los datos son introducidos por consola:

