GastroEdi

Estudiante: Guillermo Barco Muñoz
DNI: 76126510-Y

Curso: 2014/2015
Grupo de prácticas:
Profesor de prácticas:

Fecha de entrega:
Convocatoria:

Contenido:

Descripción del problema

Con motivo del nombramiento de Cáceres de capital gastronómica española en 2015, los alumnos de Estructura de Datos y de la Información se va a encargar de hacer un gestor de reservas para los restaurantes de la localidad.

Los visitantes se registrarán e informaran de los restaurantes en los que desean comer y los acompañantes que llevan.

Se asignaran los visitantes a los restaurantes en orden de llegada y en función de sus prioridades.

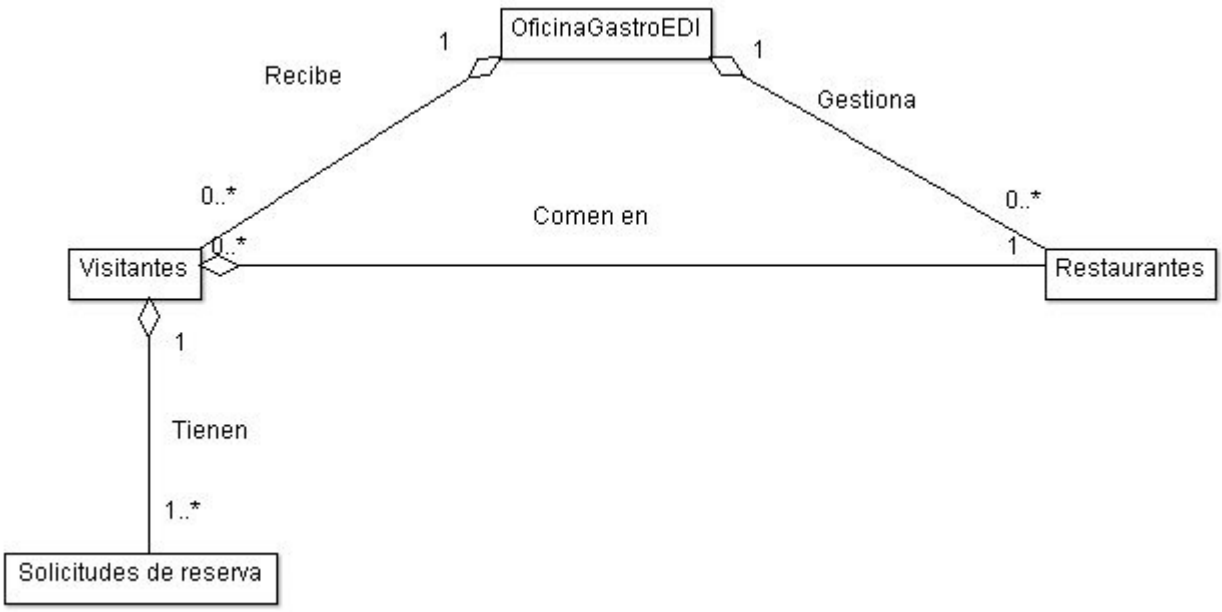
Después de comer, los clientes dejarán una valoración numérica de su experiencia en el restaurante.

Descripción de la solución

GASTROEDI

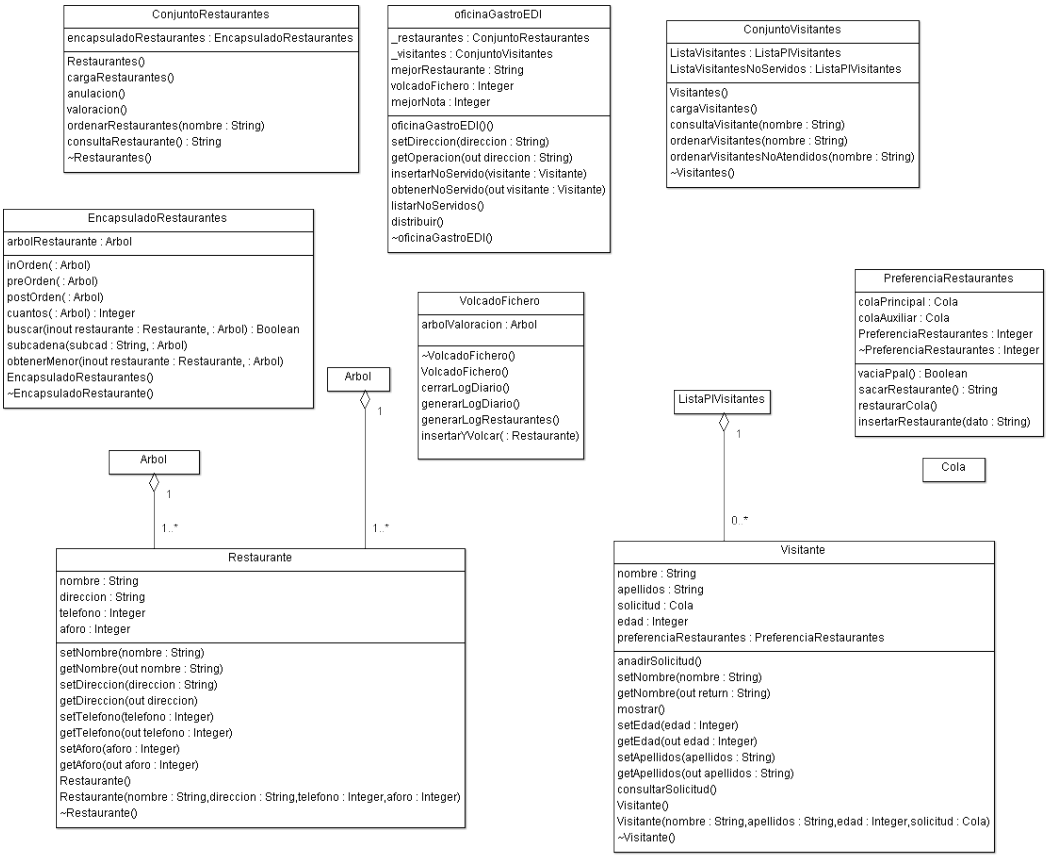
Metodología de desarrollo

Análisis



En el análisis las cosas parecen bastante sencillas y claras. La implementación se complicará mucho más y diferirá bastante de esté.

Diseño del proyecto



Implementación del proyecto

Comparar.h : es una clase que sirve como criterio de comparación por nombre para el árbol de restaurantes.

CompararValoracion.h : es una clase que sirve como criterio de comparación por nota obtenida para el árbol de restaurantes (Para cumplir con el requerimiento de que en el .log salgan por nota obtenida, que aparece en el documento fé de erratas).

ConjuntoRestaurantes.h : es una clase que interactúa con la clase que encapsula el árbol de restaurantes y la clase GastroEDI, gestiona todas las operaciones relacionadas con los restaurantes.

Su método más complicado es procesarVisitante, que comprueba si un determinado visitante puede caber en un restaurante.

ConjuntoVisitantes.h : Gestiona todas las operaciones relacionadas con los visitantes.

Estructuras de datos:

Lista visitantes
Lista visitantesNoServidos

Ambas son listas puesto que es necesario recorrerlas de una forma eficiente.

EncapsuladoRestaurantes.h : Recubre el árbol de restaurantes y permite realizar operaciones como recorrerlo o contar sus elementos.

Estructuras de datos:

Arbol arbolRestaurante;

Un árbol es la estructura de datos más útil porque es muy eficiente al consultar un dato en concreto ya que tiene complejidad logaritmica.

GastroEDI.h: Con esta clase se comunica la interfaz de usuario, posibilita la carga de ficheros y coordina a las clases ConjuntoVisitantes y ConjuntoRestaurantes.

PreferenciaRestaurantes.h : Todo visitante cuenta con una instancia de esta clase, cuenta con dos colas, lo que permite no perder la información de los restaurantes preferidos y poder recuperarla de la cola auxiliar en caso de que no se asigne ningún restaurante al visitante.

Estructuras de datos:

Cola colaPrincipal
Cola colaAuxiliar

Una cola es la mejor opción porque la asignación de un restaurante se hace en el orden en el que se insertaron en la estructura de datos, esto es por definición para lo que está pensada una cola.

Restaurante.h : Clase que contiene información del restaurante y de sus comensales.

Estructura de datos:

Lista clientes

Una lista es buena opción ya que es necesario recorrer varias veces todos los elementos en métodos como, por ejemplo, el de mostrar.

ui.h : Clase que contiene el main del proyecto y actua como interfaz de la aplicación.

Visitante.h : Clase que contiene información del visitante y sus preferencias para comer.

VolcadoFichero.h : Clase que se encarga de la generación de dos de los tres ficheros .log de la aplicación. Además ordena los restaurantes por valoración. También se encarga de liberar la mayor parte de la memoria procedente de los restaurantes y visitantes.

Estructura de datos:

Arbol arbolValoración

Es una estructura de datos útiles para poder comparar por la nota y extraer los datos de mayor a menor.

Batería de pruebas

En la clase EncapsuladoRestaurantesPrueba, se han probado todos los recorridos del árbol, el contado, la búsqueda por nombre y subcadena asi como la inserción.

Para la clase ConjuntoRestaurantes no han sido necesarias pruebas puesto que es una clase intermedia y toda la complejidad de los métodos se encuentra en EncapsuladoRestaurantes.

En la clase ConjuntoVisitantesPrueba, se ha comprobado el correcto funcionamiento de la consulta de un visitante, la extracción, inserción en visitantes no servidos y el mostrado de los no servidos.

Para la clase Visitante no son necesarias pruebas ya que no alberga ninguna estructura de datos y sus métodos son simples.

En la clase RestaurantePrueba, se ha probado la inserción, mostrado y extracción de clientes.

*Nota, el proyecto que contiene las pruebas muestra un error a veces en algunas clases. El error no afecta para nada, se puede ejecutar sin problemas y en el proyecto principal no pasa.

Lista de errores

Implementación de métodos con poca planificación y sobre la marcha: Es una mala práctica de programación y tengo que reconocer que en algunas ocasiones lo he hecho. Debido a esto he tenido que rehacer parte del código.

No leer con suficiente antención las especificaciones del proyecto: Otra mala práctica. Aunque sí me he leido la documentación varias veces, no la he tenido presente durante todo el desarrollo, lo que me ha llevado a errores a lo largo de la implementación.

Hacer el delete de los restaurantes antes de tiempo: En un principio eliminé los restaurantes al hacer la valoración. Cuando los necesité al final para generar el log, me di cuenta de mi error y lo cambié volcandolo en un nuevo árbol que los ordena por valoración para cumplir con las especificiaciones del proyecto.

Implantación o puesta en marcha

Se debe importar el proyecto GastroEDI en un entorno de desarrollo (preferentemente eclipse).

Para la ejecución de el proyecto es recomendable tener un fichero llamado restaurantes.txt y otro llamado visitantes.txt, según lo indicado por los profesores de la asignatura. Sin embargo, si no existen estos ficheros, la aplicación funcionara con los datos por defecto.

El proyecto se debe compliar (icono del martillo) y ejecutar (botón verde con el icono de play).

Para interactual con él, se deben pulsar los números y después la tecla enter. En algunos casos será necesario introducir algunos datos.

Se debe finalizar la aplicación correctamente (pulsando la tecla 10), para que se generen los ficheros log correctamente.

Se podrá ver información de la aplicación en los ficheros:

“gastroedi_diames.log”
“visitantesnoatendidos.log”
“restaurantesmasvalorados.log”

Impresiones personales

Historial de desarrollo y planificación

Generalmente he ido desarrollando la aplicación en el orden en que se debe ejecutar, lo que me ha permitido probar mas exhaustivamente cada opción.

Desde el principio he creado las clases conjuntoRestaurantes y conjuntoVisitantes para ir desarrollando la aplicación en conjunto, en lugar de completar primero unas clases y luego otras.

Para la tercera entrega decidí incluir una clase que encapsulara el árbol, que aunque en algunos casos se limita a llamar a un método del propio árbol, me permite incluir ahí alguno de los métodos que otorgan funcionalidades adicionales.

La última clase desarrollada ha sido la del volcado en fichero. Donde se incluye un nuevo árbol que ordena por nota obtenida.

Fortalezas y debilidades

La mayor fortaleza del programa es que es muy estable. Tras hacer varias pruebas, no he detectado ningún tipo de detención indebida.

Cumple con todos los requisitos a la perfección, genera ficheros de log idénticos a los facilitados por los profesores.

Puede ejecutarse utilizando los datos de visitantes y restaurantes que se encuentran en la clase gastroEDI si no encuentra los ficheros restaurantes.txt y visitantes.txt.

Respeta en todo momento la encapsulación de las estructuras de datos.

Incluye algunas funciones adicionales para el árbol de cara a una posible modificiación del proyecto (como el contado de los elementos).

No hay perdidas de memoria, todo new lleva su delete correspondiente.

Como debilidad, puede que algunos metodos se limiten únicamente a devolver un valor calculado en otra clase y funcionen solo como conexiones entre clases. Aunque he intentado solventar esto lo mejor que he podido y sabido.

Pese a que hubiera sido más correcto hacer todo el volcado de ficheros en la misma clase, el fichero de visitantes no atendidos se genera en conjuntoVisitantes por no complicar más el código.

Opinión personal y conclusiones

En mi caso, tras haber realizado también el proyecto el año pasado, tengo que decir que este me ha gustado bastante más. Queda todo bastante claro y al tener unos datos con los que comparar, permite saber si el proyecto funciona correctamente.

Quizás haya echado en falta un poco más de documentación sobre como implementar el proyecto.

Tras realizar el proyecto, creo que he aprendido bastante sobre como usar estructuras de datos respetando los principios de la programación orientada a objetos.