Estructuras de Datos
Práctica 1
Laboratorio

Realizado por:
Javier Alonso Lledó 09109855S
Daniel Verduras Gallego 09102432K

1)Especificación concreta de la interfaz de los TAD'S implementados

1.1)TADS´s creados

Las operaciones se encuentran especificadas en el siguiente apartado

espec PEDIDO
usa BOOLEANOS, NATURALES, STRING
var
p: pedido
nombre, ncliente, direccion, tipo, tarjeta : string
tiempo, prioridad: natural
erroneo: booleano
generos pedido

espec NODO[PEDIDO]
usa PEDIDO
parametro formal
generos pedido
fparametro
var
n:nodo
ped: pedido
siguiente: puntero a pnodo
generos pnodo

espec PILA[PEDIDO]
usa BOOLEANOS, PEDIDO
parametro formal
generos pedido
fparametro
var
p:pila
cima: puntero a pnodo
generos pila

ecuaciones de definitud
esVacia(p) = F → Def(desapilar())
esVacia(p) = F → Def(mostrarCima())

espec COLA[PEDIDO]
usa BOOLEANOS, PEDIDO, NATURALES
parametro formal
generos pedido
fparametro
var
c:cola
primero, ultimo: puntero a pnodo
longitud: natural
generos cola
ecuaciones de definitud
esVacia(c) = F → Def(desencolar())
esVacia(c) = F → Def(prim())
esVacia(c) = F → Def(ult())

espec LISTA[PEDIDO]
usa BOOLEANOS , PEDIDO, NATURALES
parametro formal
generos pedido
fparametro
var
l:lista
primero, ultimo: puntero a pnodo
longitud: natural
generos lista
esVacia(l) = F → Def(resto())
esVacia(l) = F → Def(eult())
esVacia(l) = F → Def(prim())
esVacia(l) = F → Def(ult())

espec WEB[PEDIDO]
usa BOOLEANOS, PEDIDO, PILA, COLA y LISTA
parametro formal
generos pedido
fparametro
var
w:web
pilaErroneos: pila
colaReg: cola
colaNR: cola
listaEnviar: lista
generos web

1.2) Definición de las operaciones del TAD (Nombre, argumentos y retorno)

A continuación se mostrará las operaciones de las TADS divididas por pertenencia:
PEDIDO:
Pedido:string → pedido
Pedido:string, string, string, string, string, int→ pedido
getTipo: → string
getPrioridad: → string
toStr: → void
NODO:
Nodo: pedido, pnodo → pnodo
PILA:
Pila: → pila
apilar: pedido→ void
apilarOrdenado:pedido→ void
parcial desapilar: → void
parcial mostrarCima→ pedido
esVacia:→ booleano
verPila: → void

COLA:
Cola:→ cola
encolar: pedido → void
parcial desencolar: pedido → void
parcial prim: → pedido
parcial ult: → pedido
esVacia: → booleano
verCola → void
LISTA:
Lista: → lista
insertarIzq : pedido → void
insertarDer: pedido → void
insertarOrdenado : pedido → void
parcial resto: → void
parcial eult: → void
parcial prim: → pedido
parcial ult: → pedido
lon: → int
es_vacia: → bool
verLista: → void
WEB:
Web: → web
introducirPedido: pedido → void
introducirTxt: → int
pasarTiempo: → void
mostrarColas: → void
mostrarPila: → void
mostrarLista: → void

2)Solución adoptada: Descripción de las dificultades encontradas

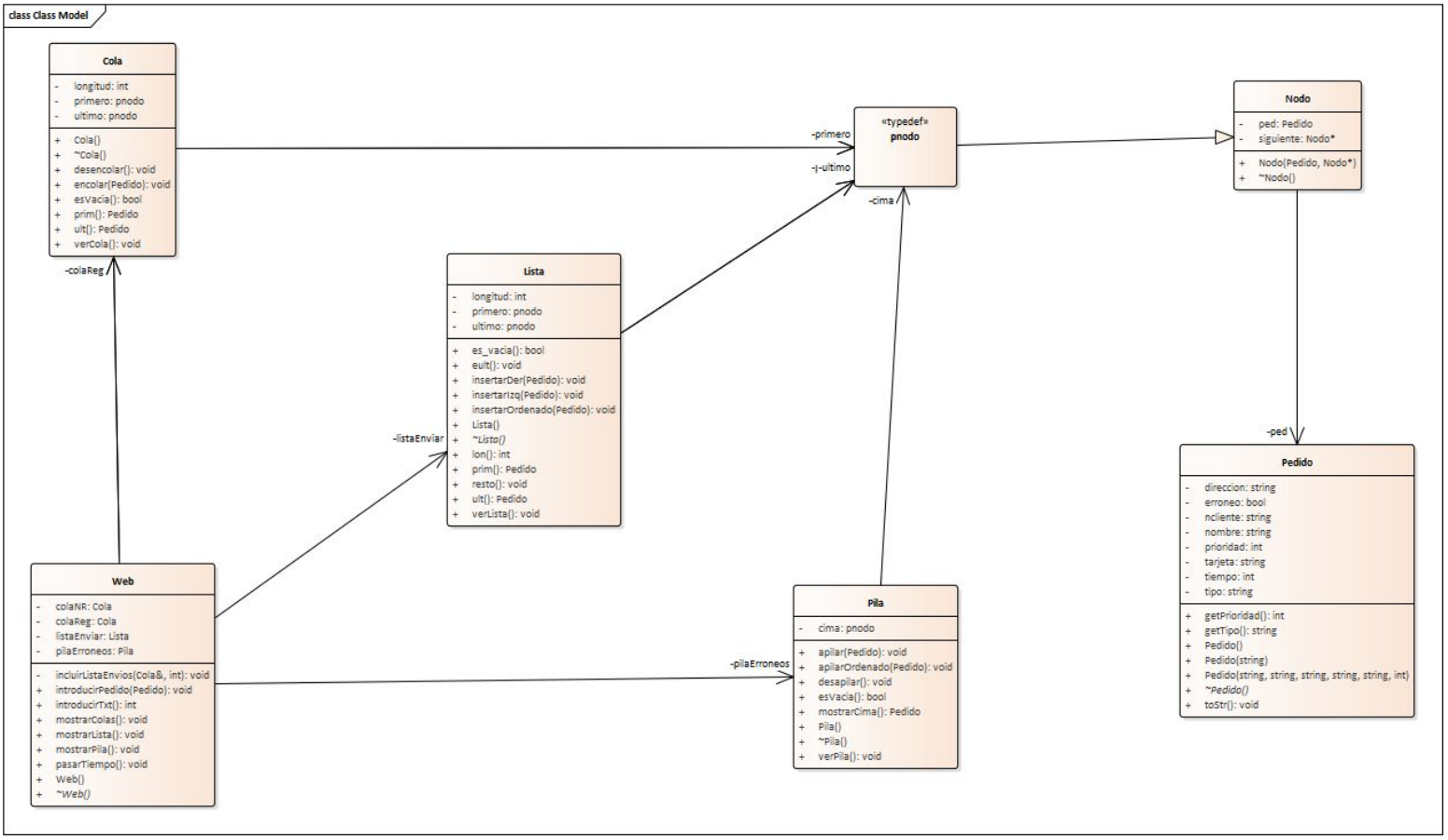
Las mayores dificultades que hemos enfrentado a lo largo del proyecto fueron principalmente relacionadas con las clases Pedido y Web.
En la clase Pedido tuvimos que documentarnos sobre los strings como tratarlos y dividirlos. Todo esto para poder dividir los strings con los datos de pedidos y colocarlos en sus respectivos atributos.
La segunda y mayor dificultad de los pedidos fue a la hora de diseñar el constructor con un tratamiento de errores que cumpliera con los siguientes estándares:

- Comprobación de que el tiempo de preparación fuera válido.
- Comprobar la falta de datos introducidos y autocompletar los pedidos
- Comprobar la validez del tipo de cliente introducido y en caso contrario establecerlo como no registrado(NR).

Continuando con las dificultades de la clase Web encontramos que todas ellas iban en torno a la operación encargada de realizar la simulación. Para su correcto funcionamiento fue necesario establecer una serie de operaciones adicionales(introducirPedido y incluirListaEnvios). Además del conjunto de errores que se desarrollaron a lo largo del proyecto y que se destacaron en este punto avanzado del desarrollo.
La suma de todas estas dificultades ralentizaron el desarrollo del proyecto sobretodo en las etapas más tardías.

3) Diseño de la relación entre las clases de los TAD implementados

3.1)Diagrama UML



3.2)Explicación de los métodos más destacados

En las estructuras de datos, los métodos más importantes son:

- En las listas, el método insertar ordenado. Cuando se pasan los pedidos de las colas a la lista para procesarlos, este método los inserta según su tipo(VIP los primeros,NVIP y NR los últimos).
- En las pilas, el método apilar ordenado. Apila los pedidos según su tipo. Encima de la pila los VIP, luego los NVIP y al fondo los NR.
- En la clase pedido, el método más a destacar es el constructor de Pedido, al que se le pasa el string en bruto y lo tiene que limpiar (las //) y depurar, por si es un pedido que le falten datos, que el tiempo esté mal introducido o que sea un tipo distinto de VIP, NVIP y NR. Básicamente tiene un contador que si no cuenta los 6 elementos necesarios, detecta que es un pedido erróneo automáticamente y genera los datos que faltan. Si el

tiempo es, por ejemplo, un string (que no se puede convertir a int), genera el tiempo aleatoriamente entre 1 y 10 y lo marca como erróneo. Lo mismo si detecta un tipo distinto a los preestablecidos.

El otro gran método es la función pasarTiempo en la clase Web. Es básicamente la función que simula todo el programa. Principalmente, la función coge los pedidos de las colas y los pasa a la lista o a la pila de erróneos según el momento. También se encarga de procesar los pedidos, es decir, simula el paso del tiempo para que, cuando se termine un pedido, añada los correspondientes a la lista de envíos. La función no se termina hasta que el sistema se queda vacío. Esta función viene ayudada por la función auxiliar introducirListaEnvios, que lo que hace es, de una cola, pasar los pedidos a la lista de envíos, pero si el pedido es erróneo, lo pasa a la pila.
Todas estas operaciones de insertar en la lista o apilar en la pila se hacen con los métodos nombrados anteriormente(apilarOrdenado e insertarOrdenado).

3.3)Explicación del comportamiento del programa

El programa comienza mostrando los nombres de los creadores y, a continuación, pregunta al usuario si quiere que los pedidos sean leidos de un fichero de texto o cogerlos del propio código(en el código hay implementados 10 pedidos, dos de ellos erróneos, para así tener el mínimo requerido en caso de que no se decida leer del fichero de texto.)

Si se elige leer del fichero de texto, si hay menos de 10 pedidos en el fichero de texto se introducirán pedidos del código hasta llegar al mínimo que son 10.

Una vez teniendo todos los pedidos se introducen al sistema Web y se muestra por pantalla la situación de las estructuras de datos del mismo. Como es lógico, en este punto sólo tendrán pedidos las colas, ya que aún no ha comenzado la simulación y no se han pasado los pedidos erróneos a la pila ni los pedidos normales a la lista de envíos.

En este punto ya comienza la simulación. Se muestra el estado de las EEDD una vez más, pero esta vez la lista de envíos ya tiene pedidos y la pila puede que tenga o no.

A partir de aquí se empieza a procesar el primer pedido, simulando el paso del tiempo. Cada vez que se pulse una tecla del teclado avanzará el tiempo en 1 minuto. Cuando ha pasado el tiempo necesario para que se procese el pedido actual, te muestra los datos del pedido, se elimina de la lista y pasan a añadirse nuevos pedidos(según el enunciado de la práctica).

Cada vez que se procesa un pedido se muestran todas las estructuras de datos para ver la evolución y el funcionamiento interno de estas.

Cabe destacar que en múltiples puntos del programa se usa la función system("cls") para borrar la consola y que no se almacenen demasiados datos a la vez, ya que podría ser confuso.

4)Código fuente

El código fuente viene adjunto con la memoria pero añadimos el enlace de Github que utilizamos para tener registro de nuestro trabajo y poder acceder a todos los cambios introducidos en cualquier momento.
El repositorio de GitHub se ha mantenido privado hasta el momento del envío, para prevenir plagios.
https://github.com/javierxxal/EDWeb

5)Bibliografía

Las diapositivas de clase sobre Pilas , Colas y Listas tanto de teoría como de laboratorio.

http://www.cplusplus.com/reference/fstream/fstream/
http://www.cplusplus.com/reference/string/string/
http://www.cplusplus.com/reference/cstring/

StackOverflow(en general)