Estructura de Datos
Grado II

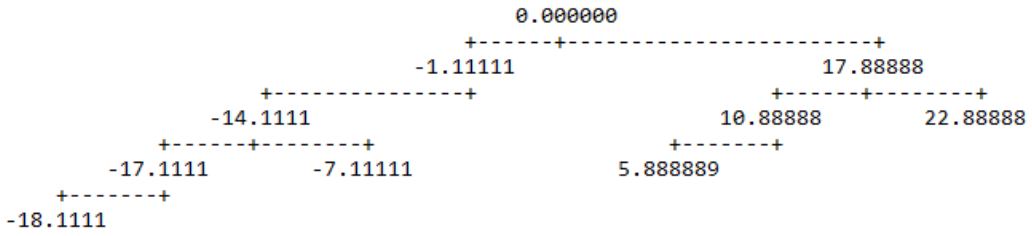
PECL2 2019-20

SIMULACIÓN DEL CONTROL DE ADUANAS EN UN AEROPUERTO

Carlos Javier Hellín Asensio

ÍNDICE

ABB y datos de la práctica 2




```
Satisfaccion media de los pasajeros: 0.000000

Datos de los pasajeros que tienen mayor y menor nivel de satisfaccion:
Mayor: (id: 3, satisfaccion: 22.888889)
Menor: (id: 9, satisfaccion: -18.111111)

Datos de los pasajeros que tienen un nivel de satisfaccion mayor que la media, ordenados de menor a mayor puntuacion:
(id: 8, satisfaccion: 5.888889)
(id: 7, satisfaccion: 10.888889)
(id: 1, satisfaccion: 17.888889)
(id: 3, satisfaccion: 22.888889)

Datos de los pasajeros cuya puntuacion es negativa recorriendo el ABB en preorden:
(id: 2, satisfaccion: -1.111111)
(id: 4, satisfaccion: -14.111111)
(id: 6, satisfaccion: -17.111111)
(id: 9, satisfaccion: -18.111111)
(id: 5, satisfaccion: -7.111111)

Datos de todos los pasajeros ordenados por su puntuacion:
(id: 3, satisfaccion: 22.888889)
(id: 1, satisfaccion: 17.888889)
(id: 7, satisfaccion: 10.888889)
(id: 8, satisfaccion: 5.888889)
(id: 2, satisfaccion: -1.111111)
(id: 5, satisfaccion: -7.111111)
(id: 4, satisfaccion: -14.111111)
(id: 6, satisfaccion: -17.111111)
(id: 9, satisfaccion: -18.111111)
```

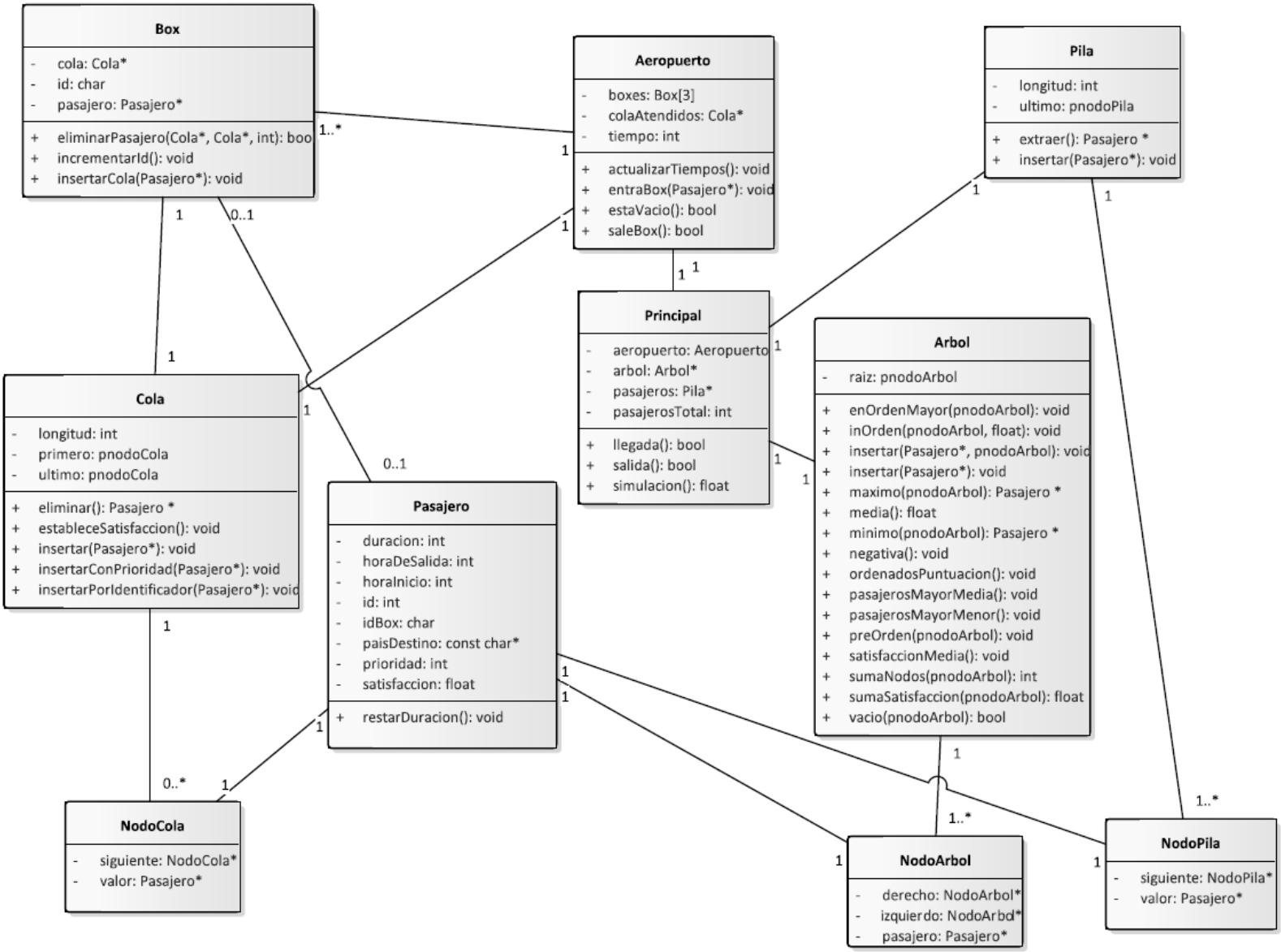
TAD`s creados y definición de las operaciones del TAD

spec ARBOL[PASAJERO]
usa BOOLEANOS, NATURALES
parámetro formal
    géneros pasajero
fparametro
géneros árbol
operaciones
carbol*:* → árbol {crea un árbol vacío}
insertar: pasajero → arbol {poner un pasajero en el árbol}
enOrdenMayor: enlace-arbol {muestra los pasajeros de mayor a menos}
inOrden: enlace-arbol, float {muestra en pantalla en orden de izquierda, raíz, derecha si la satisfacción es mayor que la media}
máximo: enlace-arbol → pasajero {devolver pasajero con la satisfacción máxima}
mínimo: enlace-arbol → pasajero {devolver pasajero con la satisfacción mínima}
mostrar: nada → nada {muestra el árbol en pantalla}
negativa: nada → nada {muestra los pasajeros cuya puntuación es negativa recorriendo el ABB en preorden}
ordenadosPuntuacion: nada → nada {muestra todos los pasajeros ordenados por su puntuación}
pasajerosMayorMedia: nada → nada {muestra los pasajeros que tienen un nivel de satisfacción mayor que la media, ordenados de menor a mayor puntuación}
pasajerosMayorMenor: nada → nada {muestra en pantalla los pasajeros que tienen mayor y menor nivel de satisfacción}
preOrden: enlace-arbol → nada {muestra en pantalla en orden de raíz, izquierda, derecha}
sumaNodos: enlace-arbol → natural {suma los nodos que contiene el árbol}
sumaSatisfaccion: enlace-arbol → float {suma la satisfacción de todos los pasajeros}
media: nada → float {calcula la media de satisfacción usando sumaSatisfaccion y sumaNodos}
satisfaccionMedia: nada {muestra en pantalla la satisfacción media de los pasajeros}
vacio: enlace-arbol → bool {devolver true si el nodo está vacío}
tipos
enlace-arbol = **puntero a** nodo-arbol
nodo-arbol = **reg**
    valor: pasajero
    derecho: enlace-arbol
    izquierdo: enlace-arbol
freg
arbol = **reg**
    raiz: enlace-arbol
freg
ftipos

Solución adoptada: descripción de las dificultades encontradas

- Antes para calcular el tiempo medio se hacía en el Aeropuerto con dos variables float, ahora al terminar la simulación se recorre una cola de pasajeros atendidos y se calcula la media con su tiempo de estancia (hora de salida – hora de llegada) aprovechando también para poder calcular la satisfacción de cada pasajero para esta práctica.
- El algoritmo de pintar el árbol tiene problemas para mostrarse en condiciones cuando se tiene más pasajeros de lo normal (más de 9 pasajeros)

Diagrama UML



Explicación de los métodos más destacados

Clase Arbol
satisfaccionMedia(): recorre el árbol para saber la satisfacción total y el número de nodos para calcular la media, usando los métodos sumaSatisfaccion() y sumaNodos() de forma recursiva.
pasajerosMayorMenor(): con los métodos máximo que recorre hasta el último nodo del lado derecho del árbol y el mínimo que hace lo mismo pero con el lado izquierdo, se muestra los datos de los pasajeros que tienen mayor y menor nivel de satisfacción.
pasajerosMayorMedia(): usando el método inOrden() para que estén ordenados de menor a mayor puntuación y pasándole la media, se obtiene los datos de los pasajeros que tienen un nivel de satisfacción mayor que la media.
negativa(): recorriendo el árbol binario con preOrden() y pasándole el nodo izquierdo de raíz se obtiene los datos de los pasajeros cuya puntuación es negativa.
ordenadosPuntuacion(): llamando al método enOrdenMayor() que recorre el árbol en el orden de izquierda, derecha y raíz se obtiene los datos de todos los pasajeros ordenados por su puntuación.

Clase Box
eliminarPasajero(Cola *cola, Cola *colaAtendidos, int tiempo): en el momento de la eliminación de los pasajeros se establece la hora de salida y se inserta en la cola de atendidos ordenados por su identificador.

Clase Principal
simulacion(): se ha modificado para cuando termine la simulación obtenga la cola de atendidos del aeropuerto y establezca la satisfacción de cada uno de los pasajeros e insertarlo en el árbol binario. Por último, muestra el árbol y todos los datos que pide el enunciado.

Clase Cola
estableceSatisfaccion(): recorre la cola para obtener el tiempo medio calculado con el tiempo de estancia de los pasajeros (hora de salida – hora de entrada) y dicho valor lo establece como satisfacción a cada uno de los pasajeros.

Explicación del comportamiento del programa

Se inicia en el punto de entrada main del programa que crea una pila donde se insertan los pasajeros para realizar las pruebas de la simulación. El último de la pila es el que más tarda en salir y el primero es el que sale antes.

Se inicia la clase Principal pasando al constructor el puntero de la pila y se llamada a la función simulación() que devolverá el tiempo medio de atención en minutos para luego ser mostrado en la consola.

El constructor de la clase Principal obtiene la pila de pasajeros y guarda en una variable la longitud de la pila para luego calcular la media. También tiene como atributo la clase Aeropuerto cuyo constructor crea un nuevo array para los boxes, inicializa el tiempo a 0, crea una cola y otra cola para los pasajeros atendidos.

En simulacion() entra en el bucle while que duerme el proceso mediante sleep para mostrar en tiempo real el resultado de la simulación. Empieza llamando a los métodos llegada() y salida() para simular los eventos del aeropuerto, comprueba que si el aeropuerto está vacío y no hay más pasajeros en la pila entonces devuelve la media. En caso contrario continua en el bucle actualizando los tiempos.

Cuando se produce un evento de llegada entonces se busca un box vacío para posteriormente asociar ese pasajero al box y si no está vacío lo inserta en la cola.

En el evento de salida se van eliminado los pasajeros que han terminado de ser atendidos en los boxes, que esto es cuando la duración del pasajero es igual a 0, se le da la hora de salida al pasajero y se inserta en la cola de atendidos ordenados por su identificador. Una vez hecho esto, se obtiene el siguiente pasajero de la cola principal.

La clase Árbol del tipo binario es la encargada de la gestión de la satisfacción de los pasajeros. Tiene una raíz con un pasajero ficticio de satisfacción 0 y partir de su nodo izquierdo se encuentran los pasajeros con satisfacción negativa y a partir del nodo derecho con satisfacción positiva. Tiene métodos para insertar los pasajeros de forma recursiva, para sumar la satisfacción y el número nodos recorriendo el árbol binario y de forma recursiva para poder calcular la media y otros métodos habituales en este tipo de estructura de datos como inOrden (izquierda, raíz, derecha) y preOrden (raíz, izquierda, derecha). Todo los datos que solicita el enunciado se hacen de forma recursiva y siempre recorriendo el árbol.

Tanto la clase Pila como la Cola tienen su funcionamiento estándar de estas estructuras de datos, exceptuando la Cola que tiene como método insertar con prioridad para que la cola tenga los pasajeros ordenados de mayor a menor prioridad y también insertar por el identificador del pasajero para hacer uso de la cola de atendidos.

Box y Pasajero tienen los atributos necesarios para usar su método mostrar en pantalla el resultado además de los exigidos en el enunciado de la práctica. Box se encarga de asociar un Pasajero a sí mismo, de insertar el Pasajero a la cola principal. Además, el comportamiento de Pasajero es la de reducir la duración del tiempo que está en un Box mientras es atendido además de su método de mostrar en pantalla.

Una vez terminado la simulación, la clase Principal se encarga de obtener la cola de atendidos del Aeropuerto para establecer la satisfacción de los pasajeros recorriendo dicha cola. Se inserta los pasajeros al árbol binario y se procede a mostrar los datos que pide el enunciado: mostrar el árbol, la satis*facción media, pasajeros ordenados de mayor a menor, etc…*