Reading By Abstraction

Agenda

addEntry

Línea 44: asigna el valor first al nodo cur.

Línea 47: a la variable booleana found, le asigna el valor true.

Línea 46-48: si el nombre de la persona que está en el nodo actual (cur) es igual al nombre de la persona que está en el nodo que estamos buscando (p), found pasa a valer true.

Línea 45-51: mientras el nodo actual (cur) sea distinto de null y no se haya encontrado la persona que buscamos en la agenda (found== false), se comprueba si el nombre del nodo actual coincide con el nombre del nodo pasado por parámetro y, si no, se avanza un nodo.

Línea 43-52: si el primer nodo es distinto de null, cur pasa a valer first y mientras el nodo actual (cur) sea distinto de null y no se haya encontrado la persona que buscamos en la agenda (found== false), se comprueba si el nombre del nodo actual coincide con el nombre del nodo pasado por parámetro y, si no, se avanza un nodo.

Línea 55: se crea una nueva entrada en la agenda y se asigna al nodo first.

Línea 56: se devuelve true.

Línea 58: se retorna false.

Línea 54-56: si no se ha encontrado la persona en la agenda, se crea un nuevo nodo con esa persona y se le asigna al nodo first y se devuelve true.

Línea 57-58: si la persona ya está en la agenda, no se añade ningún nodo y se retorna false.

Línea 40: se crea un nodo cur de tipo AgendaNode.

Línea 41: se crea una variable booleana llamada found e inicializada a false.

Línea 39-59: El método addEntry(Entry p) pasa por parámetro una entrada para añadirla a la agenda. Si esa persona ya está en la agenda, la función devuelve false y no se añade, si por el contrario no existe en la agenda, se crea una nueva entrada con los datos de esa persona y se devuelve true.

removeEntry

Línea 70: retorna el booleano false.

Línea 75: asigna al nodo de agenda “prev” el nodo siguiente.

Línea 79: devuelve el booleano false.

Línea 81: asigna al nodo siguiente del nodo agenda “prev” el siguiente a dicho siguiente.

Línea 82: decrementa en 1 la variable “numEntries”.

Línea 83: retorna el booleano true.

Línea 69-71: si el nodo agenda “first” es nulo, se devuelve falso.

Línea 74-76: mientras el nodo siguiente al nodo agenda “prev” no sea nulo y su nombre sea distinto de la variable “name” pasada por parámetro, asignamos a “prev” su nodo siguiente.

Línea 78-79: si el nodo siguiente a “prev” es nulo, retorna falso.

Línea 80-84: si el nodo siguiente a “prev” no es nulo:

- Asigna al nodo siguiente del nodo agenda “prev” el siguiente a dicho siguiente.
- Decrementa en 1 la variable “numEntries”.
- Retorna el booleano true.

Línea 73: Creamos el nodo agenda llamada “prev” e igualamos a “first”.

Línea 68-85: El método removeEntry(String name) pasa por parámetro un nombre para eliminarlo de la agenda. Si la agenda está vacía, el método termina y devuelve false. En caso contrario, comenzará a buscar el nombre en cada uno de los nodos de la agenda. Si lo encuentra, lo elimina, si no, termina y no hace nada.

removeFirst

Línea 97: a la entrada p se le asigna la información del nodo first.

Línea 98: el nodo primero pasa a ser el siguiente.

Línea 99: se disminuye el número de entradas en 1.

Línea 96-100: si el primer nodo existe, a la entrada p se le asigna la información de ese nodo, first apunta a siguiente y se rebaja en 1 el número de entradas.

Línea 94: se crea una entrada llamada p de tipo Entry, que se inicializa a null.

Línea 102: se retorna la entrada p.

Línea 93-103: esta función borra la primera entrada de la agenda. Para ellos, si existe, guarda la información en una entrada p y borra el primer nodo, haciendo que first apunte a siguiente. Por último, devuelve la entrada borrada.

nEntries

Línea 105-107: esta función devuelve el número de entradas que hay en la agenda.

isEmpty

Línea 111: se devuelve true.

Línea 113: se devuelve false.

Línea 110-112: si el primer nodo está vacío (first == null), se devuelve true.

Línea 112-114: si el primer nodo no está vacío, se devuelve false.

Línea 109-115: el método devuelve true si el primer nodo está vacío, y false si no lo está.

saveAgenda

Línea 135: la variable booleana success pasa a valer true.

Línea 134-136: si success vale false, pasa a valer true.

Línea 137: se inserta una nueva entrada en p con la información del nodo cur.

Línea 138: se inserta en la variable línea, una nueva línea.

Línea 139: se imprime la línea.

Línea 133-140: mientras cur sea distinto de null:

- si success vale false, pasa a valer true.
- se inserta una nueva entrada en p con la información del nodo cur.
- se inserta en la variable línea, una nueva línea.
- se imprime la línea.

Línea 124: se crea una variable cur de tipo AgendaNode, y se inicializa al nodo first.

Línea 125: Se crea un string llamado line.

Línea 126: se crea una variable booleana llamada success, inicializada a false.

Línea 127: se crea una variable p de tipo Parser.

Línea 129: se crea un fichero para almacenar la información de la agenda, llamado agendofile.txt.

Línea 130: se crea un buffer de escritura llamado bufferescritura para el fichero.

Línea 131: se crea un PrintWriter llamado output, que imprime lo que hay escrito en el buffer.

Línea 123-143: La función devuelve true si la agenda se ha almacenado correctamente en un fichero .txt, y false en el caso contrario.

loadAgenda

Línea 169: se llama el método addEntry pasándole por parámetro de tipo Entry “auxEntry”.

Línea 160: se llama al método close de la variable “bufferin” de tipo BufferedReader para cerrarlo.

Línea 161: devuelve falso.

Línea 165: se imprime por pantalla el string “cad”.

Línea 166: se llama al método “insertLine” de la variable “p” pasándole por parámetro el string “cad”.

Línea 167: se crea la variable Entry llamada “auxEntry”, asignándole lo que devuelve el método “getEntry” de la variable “p”.

Línea 168-170: si el método “hasData” de la variable “auxEntry” devuelve true, se llama el método addEntry pasándole por parámetro de tipo Entry “auxEntry”.

Línea 159-162: Se le asigna a la variable “cad” lo que devuelve el método “readLine” de la variable “bufferin” y, si es nulo, se llama al método close de la variable “bufferin” de tipo BufferedReader para cerrarlo y se devuelve falso.

Línea 164-171: mientras que haya líneas en el búfer, se imprimirán, se insertarán en un nuevo Parser y, si esa variable tiene asociada una Entry, se la añade a la agenda.

Línea 154: crea la variable de tipo FileReader llamada “filein”, pasándole por parámetro “agendafile.txt”

Línea 155: crea la variable “bufferin” de tipo Bufferedreader asociada a “filein”.

Línea 157: crea la variable “p” de tipo Parser.

Línea 158: crea la variable “cad” de tipo String.

Línea 172: cerramos el fichero “filein”.

Línea 174: devuelve true.

Línea 153-176: el método loadAgenda importa una agenda pasada por un fichero txt. Si ese fichero está vacío, no hará nada y devolverá false. Si no, se irán añadiendo cada una de las líneas de la agenda como un Parser y, si es el caso, también su Entry. Al acabar, cerrará el fichero y devolverá true.

Análisis de removeFirst

Código	Lo que se supone que hace	Lo que creemos que hace

`public Entry removeFirst() {`
 	
`Entry p = null;`
 	
`if (first != null) {`
 	
`  p = first.info;`
 	
`  first = first.sig;`
 	
`  numEntries--;`
 	
`}`
 	
`return p;`
 	
`}`
	La función elimina el primer contacto de la agenda. si está vacía no hace nada. Si solo tiene un contacto, lo borra y deja una lista vacía. La función retorna un objeto de tipo Entry, si ha conseguido borrar el contacto, y null en caso contrario.	Si el primero es distinto de null(es decir, no es una lista vacía) guarda los datos del primer contacto en el objeto Entry que ha declarado, lo borra haciendo que el puntero first apunte al siguiente y disminuye el número de entradas de la agenda.
		
		Por último, retorna p (si ha conseguido entrar en el if porque no es una lista vacía p contendrá el contacto primero y sino contendrá null).