ESTRUCTURA DE DATOS

Práctica Laboratorio 3

Grado en Ingeniería

Jordi Lazo Florensa

Alejandro Clavera Poza

Universitat de Lleida
Escola Politècnica Superior

Tarea 1

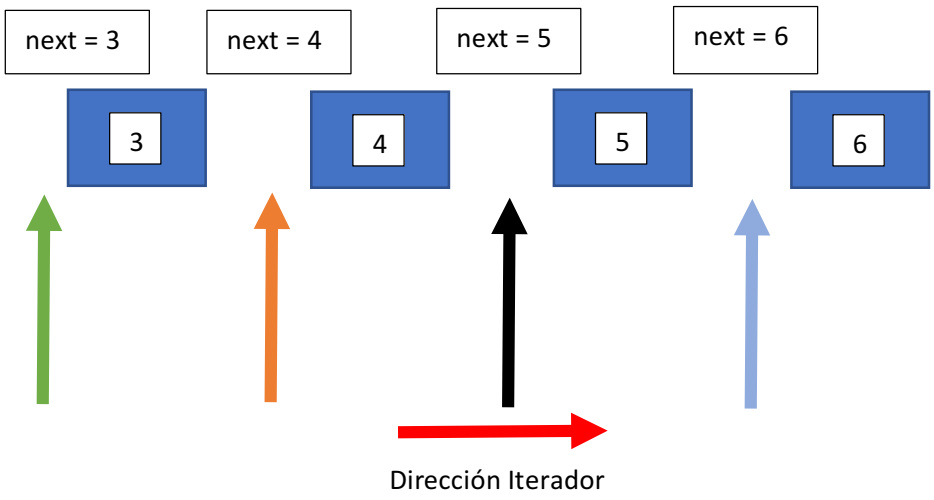
Para la realización de esta tarea primero creamos una interfaz que defina las principales operaciones de nuestra cola (add,remove,element,isEmpty,size) Tras ello creamos la clase *BankQueue* la cual implementara las operaciones de la interfaz nombrada anteriormente.

Para implementar dicha cola usaremos dos arraylist que actuaran como pilas. Una de estas representara la parte trasera de la cola, donde se irán almacenando los últimos elementos que se vayan añadiendo y la otra representara la parte delantera de la cola que es donde se guardara los primeros elementos introducidos.

En cuanto a la implementación de las operaciones la operación add se realiza mediante la inserción de un nuevo elemento en la pila back. En la operación remove primero se comprueba que no esté vacía la cola, si es así, se lanza una exception, de lo contrario comprobamos primero que la pila front no esté vacía, si lo está antes de realizar el remove se mueven todos los elementos en la pila front en la forma inversa y eliminando a su vez todos los elementos de la pila back. Una vez realizada la comprobación anterior se eliminará el último elemento que ha sido colocado en la pila front que representa el primer elemento que entró en la cola. La operación element realiza las mismas operaciones que la función remove con la diferencia que element no elimina el elemento, sino que lo retorna. La operación isEmpty que tan solo comprobamos que el tamaño de la cola no sea 0. Por ultimo en la operación size para dar el tamaño de la cola, sumamos el tamaño de las pilas.

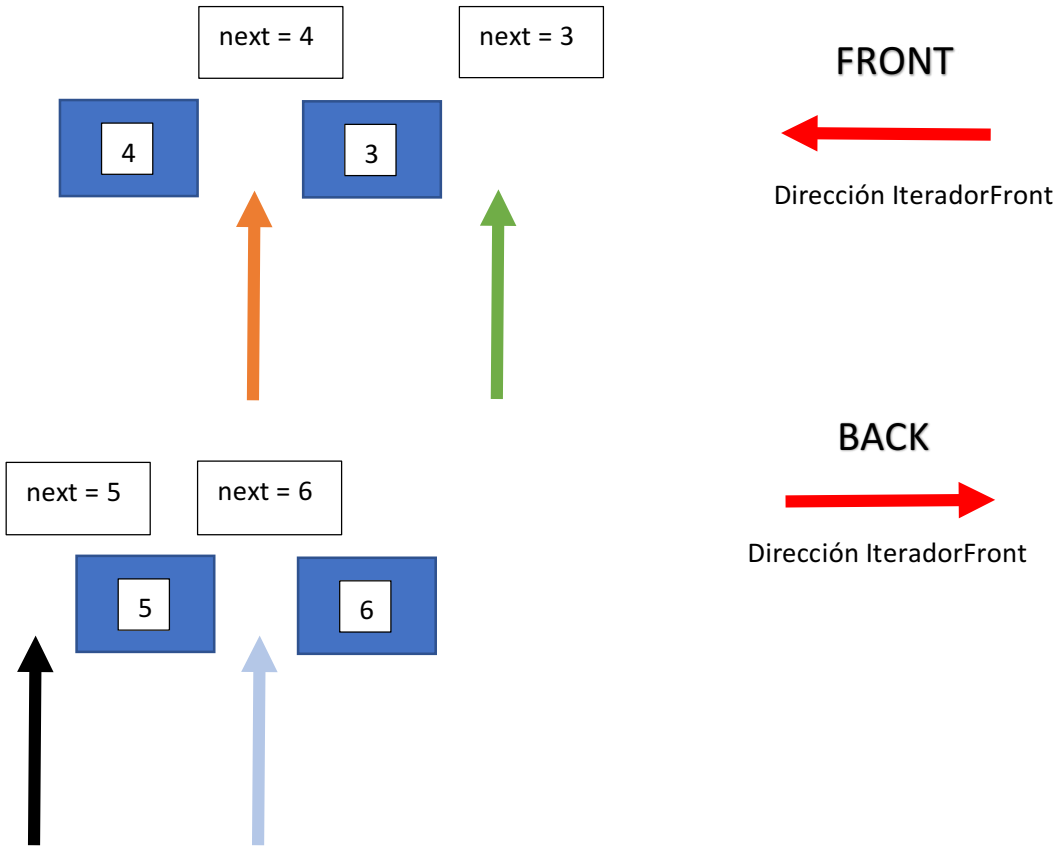
Tarea 2

En esta tarea se nos pide la realización de la cola realizada de la tarea anterior. Para esto nos valemos de los dos listIterators provenientes de los dos ArrayList utilizados anteriormente, donde el cursor del ListIterator del ArrayList front es colocado al final de ella y el cursor ListIterator del ArrayList back es colocado al principio de esta. En cuanto al recorrido por la cola de este iterador comienza moviendo el iterador front decrementando posiciones hasta que este llegue a la primera posición de este , tras ello se mueve el iterador back hasta el final de este.



(cada color de la flecha del diagrama representa una posición diferente del cursor)

Procedimiento interno del iterador



Las principales operaciones de este iterador son hasnext la cual nos dice si existe un elemento siguiente para devolver en la posición del cursor. Para esto comprobamos si el cursor de el ArrayList front no está al principio de este , si no lo está utilizaremos la función hasprevious del iterador de dicho ArrayList, por el contrario, si este ya está en esa posición, emplearemos la función hasnext del ListIterator del ArrayList back.

La operación next, devuelve si es posible el siguiente elemento. Para esto primero comprobamos que no se haya realizado ninguna modificación a la cola, en caso afirmativo lanzaremos una excepción. Por otro lado, también comprobamos que la cola no esté vacía, se esta lo esta se mandara una excepción. Si las comprobaciones anteriores no lanzan ninguna exception podremos devolver dicho elemento, para esto utilizaremos el método previous del iterador del array front decrementando una posición del iterador ArrayList front. En caso que el iterador del ArratList front este al principio de este, se utilizara la operación next del iterador del ArrayList back y se incrementa una unidad el curso de este.

Finalmente, la operación remove, la cual nuestro iterador no realizará, por tanto lanzará la exception UnsupportOperationException ya que esta operación la interfaz iterator obliga implementarla.

Tarea 3

Esta tarea consiste en la simulación de la recepción de clientes en un banco, esto nos permite probar la cola implementada en las tareas realizadas anteriormente. Para esto en cada simulación tendremos en cuenta el número de cajeros y de clientes que llegan a estos. Cada simulación comienza con la llegada del primer cliente asignándole el tiempo de llegada a 0, siendo colocado en la cola del primer cajero. A continuación, el cliente será atendido durante 120 segundos, por otro lado, irán llegando más clientes cada 15 segundos los cuales se irán repartiendo de forma equitativa por todas las colas. Por otro lado, se contabilizará cuánto tiempo lleva cada cliente siendo atendido, si estos clientes han llegado a su límite se les asignará el tiempo de salida de ese momento. Posteriormente, se atenderá al siguiente cliente siempre que exista algún cliente en la cola de dicho cajero. La simulación acabará cuando todos los clientes hayan sido atendidos, mostrando el tiempo medio que ha pasado cada cliente.

Para realizar estas simulaciones, hemos planteado 10 situaciones diferentes, dónde cada situación tiene 1 cajero más que el anterior y en todas las situaciones se atienden 100 clientes.

Tras realizar estas simulaciones como podemos ver en el siguiente grafico observamos que llega un momento que por más que haya más cajeros que se añadan el tiempo medio que pasa cada cliente no disminuye.