Estructuras de Datos
PECL

SIMULACIÓN DEL CONTROL DE ACCESO A UN EVENTO DEPORTIVO

PARTE 1

En esta práctica se pretende simular el funcionamiento del control de acceso a un evento deportivo que cuenta con dos colas para la entrada de los aficionados:

- Una cola para aficionados socios que tienen prioridad para acceder primeros al estadio.
- Una cola para aficionados simpatizantes que entrarán después de los socios.

El evento deportivo comenzará a las 19:00 y los aficionados pueden llegar al estadio desde las 18:00 y hasta las 18:59 horas:

- La hora de llegada de un aficionado es indicada en minutos. Los minutos se cuentan desde las 18:00 horas. Por ejemplo, tiempo de llegada igual a 27 quiere decir que el aficionado llegará a las 18:27.
- Cada aficionado tendrá un identificador único.

Se pide simular la cola de espera con **dos colas implementadas dinámicamente**, y que el sistema esté gestionado en función de los dos siguientes eventos:

- Llegada de un aficionado a las colas en la puerta de acceso al estadio.
- Entrada del aficionado al estadio (los socios tienen prioridad para entrar primero).

En nuestra práctica, inicialmente se generará un conjunto de aficionados de forma aleatoria que se almacenarán en **una pila implementada dinámicamente**.

Para cada aficionado se almacena un identificador ID que es único, una hora de llegada que es indicada en minutos y un booleano que nos indica si el aficionado es socio o no. Se debe usar el ID del aficionado y en caso de que éste sea un número par, establecer el aficionado como socio y en caso contrario establecerlo como simpatizante.

Una vez generados los aficionados y almacenados en una pila, estos se dividirán en dos colas en función de si son socios o simpatizantes.

La simulación de acceso de los aficionados al estadio se hará mediante **una lista implementada dinámicamente** (puede ser simplemente enlazada o doblemente enlazada), **en la que se inserten los elementos de forma ordenada** y se hará de la siguiente forma:

Empezarán a entrar al estadio los aficionados que son socios en función de su hora de llegada (entrará primero el socio cuya hora de llegada indicada en minutos es la menor) y posteriormente entrarán los simpatizantes también en función de su hora de llegada (el último aficionado en entrar al estadio será el simpatizante que tiene mayor hora de llegada).

DESARROLLO PARTE 1

Se pide implementar una clase *Aficionado*, una clase *NodoPila*, una clase *NodoCola*, una clase *NodoLista*, una clase *Pila*, una clase *Cola*, una clase *Lista* (con sus operaciones habituales y el método de insertar en orden), una clase *Gestor* en la que se hará la simulación del control de acceso de los aficionados en una clase *main* que mostrará un Menú con las siguientes opciones:

Opción A: cada vez que se use esta opción el programa generará 10 aficionados aleatorios con sus correspondientes ID único, hora de llegada y el tipo de aficionados que son (socio o simpatizante). La primera vez que se use esta opción el ID del aficionado será un número aleatorio entre 1 y 10, la segunda vez entre 11 y 20 y así sucesivamente.

Opción B: esta opción mostrará todos los aficionados almacenados en la pila.

Opción C: esta opción borrará todos los aficionados almacenados en la pila y generados en la opción A.

Opción D: esta opción extraerá los aficionados de la pila y los almacenará en la cola de socios o en la cola de simpatizantes en función del tipo de aficionado.

Opción E: esta opción mostrará la cola de aficionados que son socios.

Opción F: esta opción mostrará la cola de aficionados que son simpatizantes.

Opción G: esta opción borrará los aficionados almenados en ambas colas.

Opción H: esta opción extraerá los aficionados de las colas de aficionados y los almacenará en una lista de forma ordenada. Es decir, se empezará extrayendo los aficionados que hay en la cola de socios y se almacenarán en la lista de forma ordenada en función de la hora de llegada de cada socio. Una vez se hayan insertado todos los socios se hará lo mismo con los simpatizantes que también estarán ordenados en función de su hora de llegada.

Opción I: esta opción buscará en la lista y mostrará los siguientes 4 aficionados:

- El primer aficionado en acceder al estadio.
- El último socio en acceder al estadio.
- El primer simpatizante en acceder al estadio.
- El último aficionado en acceder al estadio.

Opción J: Reiniciar el programa a su estado inicial.

Opción S: Salir del programa.

PARTE 2

En esta segunda parte de la práctica se completará la simulación realizada en la Parte I de la práctica anterior con la creación de un árbol binario de búsqueda (ABB), implementado dinámicamente, que almacenará los aficionados que han acudido al evento deportivo.

Para la creación del ABB se partirá de los aficionados almacenados en la lista creada en la parte I de la práctica. **El nodo raíz del árbol almacenará un aficionado ficticio** que es socio, de manera que los aficionados que son socios se insertarán, ordenados por sus IDs, en el subárbol izquierdo del nodo raíz y los que son simpatizantes en el subárbol derecho, también ordenados por sus IDs.

DESARROLLO PARTE 2

Se pide añadir a la implementación de la Parte 1 las siguientes dos clases con sus correspondientes métodos y operaciones: una clase *NodoArbol* y una clase *Árbol*.

Se deberá ampliar el menú de la Parte 1 para poder mostrar la información que se indica a continuación, **siempre partiendo del ABB creado y recorriéndolo de la forma más eficiente posible (evitando recorrer partes del árbol cuando no sea necesario):**

Opción K: crear y dibujar el ABB en consola.

Opción L: Mostrar los datos de todos los socios ordenados por sus IDs de menor a mayor (sin incluir el aficionado almacenado en nodo ficticio).

Opción M: Mostrar los datos de todos los simpatizantes ordenados por sus IDs de menor a mayor (sin incluir el aficionado almacenado en nodo ficticio).

Opción N: Mostrar los datos de todos los aficionados recorriendo el ABB en inorden.

Opción O: Buscar en el ABB y mostrar los siguientes 4 aficionados:

- El primer aficionado en acceder al estadio.
- El último socio en acceder al estadio.
- El primer simpatizante en acceder al estadio.
- El último aficionado en acceder al estadio.

Opción P: Contar el número de aficionados almacenados en el ABB cuyos ID's son pares.

Opción Q: mostrar los aficionados que se encuentran almacenados en un nodo hoja.

Opción R: Eliminar un aficionado indicado por su ID (que se pide desde consola). **Mostrar el árbol antes y después** tras la eliminación de dicho aficionado.