**Objetivos**

**Unidad 4: Estructuras y Algoritmos Recursivos**

OE4.1 Emplear el concepto de recursividad como una alternativa a la estructura de control iterativa.

OE4.2 Aplicar la computación recursiva en la solución de problemas de naturaleza inherentemente autocontenida.

OE4.3 Utilizar árboles binarios de búsqueda para representar grupos de objetos que mantienen entre ellos una relación de orden.

OE4.4 Escribir algoritmos recursivos para manipular estructuras de información recursivas y explicar las ventajas que, en este caso, estos algoritmos tienen sobre los algoritmos iterativos.

**Enunciado**

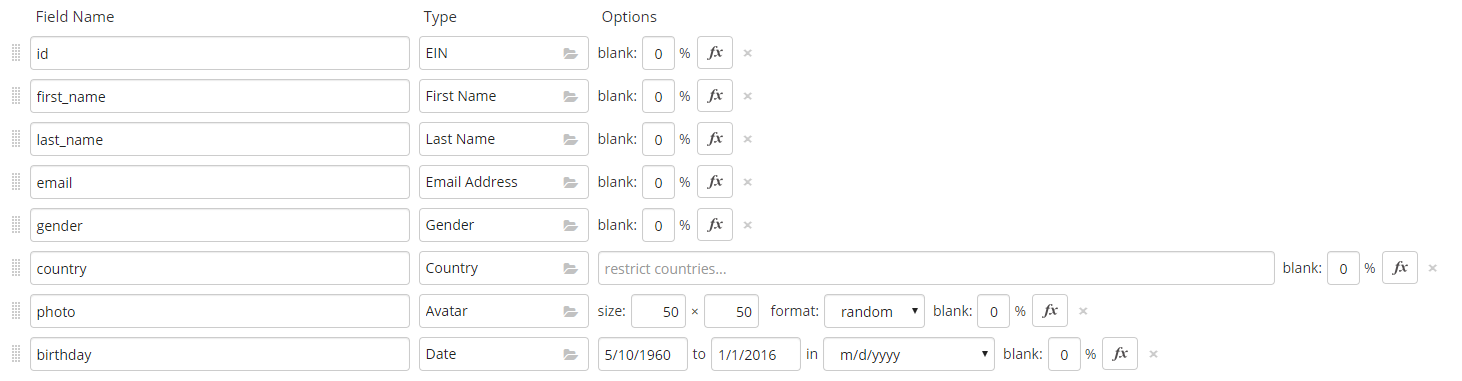
La [IV Copa Panamericana de Voleibol Masculino Sub-21](https://es.wikipedia.org/wiki/Copa_Panamericana_de_Voleibol_Masculino_Sub-21_de_2019) se celebrará del 5 al 11 de mayo de 2019 en Lima, (Perú) y la federación tiene la necesidad de registrar algunos datos de los espectadores que desean asistir al evento y por motivo de algunas labores de logística durante el evento, el personal de registro y control estará permanentemente realizando búsquedas de espectadores y participantes en el software, utilizando como criterio el id de registro del evento.

Usted ha sido elegido para desarrollar un programa que permita leer la información de los posibles asistentes al evento desde un archivo de texto delimitado por comas y cargar la información de manera que se puedan hacer búsquedas a través del identificador del registro del evento.

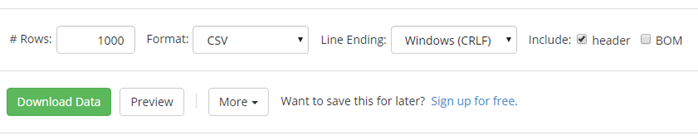
El programa debe permitir leer el archivo de texto con la información de todos los posibles participantes. Esta información se cargará en una estructura de datos de tipo árbol binario de búsqueda (ABB) cuya relación de orden estará dada por el id de registro del evento. Una vez cargada esta información en el árbol de espectadores, se seleccionará aleatoriamente el 50% de esos espectadores para agregarlos a una lista doblemente enlazada, que será la estructura de datos que almacenará los inscritos al evento o participantes al evento.

Luego de la lectura del archivo y la creación de las dos estructuras de datos: una para los posibles participantes y otra para los inscritos al evento, se debe permitir hacer búsquedas por id, para cada una de las estructuras creadas. Su programa debe indicar, en cualquier caso, si el espectador se encuentra o no entre los buscados así como el tiempo que tardó la búsqueda. Si el espectador fue encontrado, deben visualizarse sus datos en un panel a la izquierda.

Los datos de entrada al programa pueden ser generados utilizando la herramienta web [**https://www.mockaroo.com**](https://www.mockaroo.com/)que permite generar datos aleatorios de forma personalizada. La configuración y orden de los datos generados debe ser la que se muestra a continuación:



Se debe generar el máximo de filas posible para hacer buenas pruebas y el formato en que se debe exportar debe ser CSV el cual delimita los archivos de texto con coma por defecto.

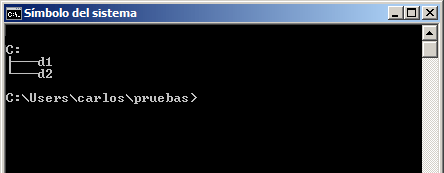


Ya que el máximo es 1000, realice pruebas del tiempo de la consulta para archivos de entrada mas grande como 100000, por ejemplo (generando 100 veces 1000 y uniéndolos). Compare los tiempos de las búsquedas cuando se realizan sobre todos los programadores y los participantes.

La interfaz del programa debe cumplir con los lineamientos:

* Permite cargar los datos de un archivo plano (ejemplo: archivo.csv) y confirmar la correcta carga, la ruta que se entra por teclado puede ser relativa o absoluta, debe especificarle al usuario final cual debe ingresar.
* Buscar espectador por id, debe cuantificar cuánto tiempo tarda la búsqueda y mostrar los datos del espectador (la foto en esta versión no será visualizada).
* Buscar participantes por id, debe cuantificar cuánto tiempo tarda la búsqueda y mostrar los datos del participante (la foto en esta versión no será visualizada).P

El programa también contará con una opción que podrá visualizar los datos de las dos estructuras de un país determinado, donde la estructura del árbol binario de búsqueda se verá como un árbol y la lista se visualizará en forma consecutiva. La estructura árbol puede ser vista de la siguiente manera:



Donde c: es la raíz del árbol y d1 y d2 son hijos de c:,para cada opción se visualizará sólo los datos del país seleccionado (puede ser el componente JOptionPane de Java), eso quiere decir, que para la estructura de espectadores el árbol pintado solo debe tener espectadores de la ciudad seleccionada e igual para la estructura de los participantes.

**Entregables.** **1.** Requerimientos. **2.** Diagrama de clases de modelo e interfaz (no generado automáticamente). **3.** Implementación completa de todos los requerimientos en Java. **4.** Diseño de casos de pruebas unitarias de todas las funcionalidades relacionadas con los árboles. **5.** Implementación de todas las pruebas unitarias diseñadas y corriendo perfectamente. **6**. Todos los métodos para administrar las estructuras deben ser implementados usando recursividad. **7**. La estructura de los espectadores deben ser un árboles binarios de búsqueda y los participantes una lista doblemente enlazada, tal como se explicaron en el tema visto en clase y no podrán usar otro tipo de contenedor como array, ArrayList, etc.

**Fecha de Entrega:** martes 15 de octubre de 2019 a las 11:55 p.m. a través de Moodle.

**Integrantes**: el laboratorio debe realizarse individualmente.

**Desarrollo:**

3.

|  |  |
| --- | --- |
| Nombre: | **RF1** Permitir leer la información asistentes al evento en un CSV desde una ruta. |
| Resumen: | Cargar la información del CSV al árbol binario de búsqueda de espectadores |
| Entradas: | Ninguna |
| Salida: | Se carga la información al árbol ABB |

|  |  |
| --- | --- |
| Nombre: | **RF2** Seleccionar aleatoriamente el 50% de los espectadores para volverlos competidores |
| Resumen: | Selecciona aleatoriamente el 50% de los espectadores para ingresarlos a la lista doblemente enlazada de competidores |
| Entradas: | Ninguna |
| Salida: | Se agrega aleatoriamente el 50% de los espectadores |

|  |  |
| --- | --- |
| Nombre: | **RF3** Buscar id para cada una de las estructuras dadas. |
| Resumen: | Busca el id de cada una de las estructuras ya sea el árbol ABB o la lista doblemente enlazada |
| Entradas: | -Nombre de la persona |
| Salida: | Se busca a la persona por el id y retorna la información si se llega a encontrar. |

|  |  |
| --- | --- |
| Nombre: | **RF4** Pintar el árbol de acuerdo al país seleccionado |
| Resumen: | Permite pintar el árbol de acuerdo al país que el usuario desee dentro del programa. |
| Entradas: | -El nombre del país |
| Salida: | Se muestra el país sí se encontró. |

4.

|  |  |  |
| --- | --- | --- |
| **RF** | **Código** | |
| **Clase** | **Método** |
| Permitir leer la información asistentes al evento en un CSV desde una ruta. | worldCup,Spectator | +addSpectatorCVS(Spectator):void  +insertSpectator(Spectator):void |
| Seleccionar aleatoriamente el 50% de los espectadores para volverlos competidores | worldCup | +loadSpectatorAndSpectator(String,String, String,String,String,String,String,String):void  +addCompetitorOrdenate(Competitor):void |
| Buscar id para cada una de las estructuras dadas. | worldCup,Spectator,Competitor | +searchCompetitor(String):String +searchSpectator1(String):Spectator +searchSpectator2(String):String +showInfo():String  +searchCompetitor(String):String |
| Pintar el árbol de acuerdo al país seleccionado | worldCup | +paintCountry(String)String +pintarArbol(Spectator,int,String):void |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Clase** | **Escenario** | **Valores de Entrada** | **Resultado** |
| +searchSpectator1(String):Spectator | worldCup | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id: “123” | Se devuelve el objeto de Robinson |
| +searchSpectator1(String):Spectator | worldCup | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id:”321” | No se devuelve ningún objeto , puesto que no se encontró a ninguno con ese id |

**Objetivo:** Saber si se ejecuta de manera correcta la búsqueda el id por espectador .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Clase** | **Escenario** | **Valores de Entrada** | **Resultado** |
| +addSpectatorCVS(Spectator):void | worldCup | El árbol inicialmente está vacío | Spectador:”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” | Lo agrega como la raíz del árbol |
| +addSpectatorCVS(Spectator):void | worldCup | El árbol ya tiene un objeto que es la raíz | Spectador:”90”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” | Lo agrega en la parte izquierda del árbol |
| +addSpectatorCVS(Spectator):void | worldCup | El arbol ya tiene dos objetos, una raíz y un subarbol izquierdo | Spectador:”200”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” | Lo agrega en la parte derecha del árbol |

**Objetivo:** Saber si se ejecuta de manera correcta la agregación de elementos en el árbol binario.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Clase** | **Escenario** | **Valores de Entrada** | **Resultado** |
| +pintarArbol(Spectator,int,String):void | worldCup | El árbol inicialmente tiene  unos países como e1= “Colombia”,e2=”México”,  e3 =“Uruguay” | String: “Colombia” | Pinta a colombia de manera correcta |
| +pintarArbol(Spectator,int,String):void | worldCup | El árbol inicialmente tiene  unos países como e1= “Colombia”,e2=”México”,  e3 =“Uruguay” | String: “Wakanda” | No imprime nada, puesto que no hay ningún país con ese nombre |

**Objetivo:** Saber si se ejecuta de manera correcta el pintado de los árboles de acuerdo al país.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Clase** | **Escenario** | **Valores de Entrada** | **Resultado** |
| +searchSpectator1(String):Spectator | Spectator | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id: “123” | devuelve el objeto de Robinson al método principal |
| +searchSpectator1(String):Spectator | Spectator | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id:”200” | No devuelve ningún objeto(null). |

**Objetivo:** Saber si el método auxiliar de buscar el espectador por id funciona de manera adecuada.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Clase** | **Escenario** | **Valores de Entrada** | **Resultado** |
| +searchSpectator2(String):String | worldCup | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id = “132” | Devuelve la información de Juan puesto que encontró su id |
| +searchSpectator2(String):String | worldCup | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id=”1000” | No devuelve ninguna información, puesto que no se encontró ningún objeto con esta información |

**Objetivo:** Saber si el método devuelve la información de los espectadores de manera adecuada.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Clase** | **Escenario** | **Valores de Entrada** | **Resultado** |
| +searchSpectator3(String):String | worldCup | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” | id: “123” | Devuelve la información de Robinson al método principal |
| +searchSpectator3(String):String | worldCup | El escenario tiene unos objetos por defecto unos objetos como e1 =”123”, “Robinson”,”Jhon”,”[juan.restrepo@correo.com](mailto:juan.restrepo@correo.com)”,”Hombre”,”Colombia”,”asdasd.com”,”13/02/14” y e2 = “132”, “Juan” , “Sebastian”,”callipop@yahoo.com”,”Hombre”,”Uganda”,”asdasdasd.com”,”14/07/98” |  | No devuelve ninguna información puesto que no se encontro ningun objeto con ese id. |

**Objetivo:** Saber si el método auxiliar de buscar la información por id, funciona de manera adecuada.