UNIVERSIDAD ESTATAL A DISTANCIA
VICERRECTORÍA ACADÉMICA
ESCUELA DE CIENCIAS EXACTAS Y NATURALES
CARRERA INGENIERÍA INFORMÁTICA

CÁTEDRA INGENIERÍA DE SOFTWARE

ASIGNATURA

00825 ESTRUCTURA DE DATOS

PROYECTO 1

VALOR: 30% (3.0)

I CUATRIMESTRE 2024

Proyecto 1

Objetivo de aprendizaje

- Adquirir los conocimientos básicos sobre las estructuras de datos.
- Analizar la forma en que se implementa la recursión y comprender cuándo se debe utilizar.
- Comprender el funcionamiento de algoritmos de ordenamiento.

Temas de estudio

- La API de colecciones y recursividad.
- Algoritmos de ordenamiento.

Descripción del trabajo

Precondiciones:

1. Como herramienta de desarrollo se deberá utilizar el Netbeans en su versión 19 o superior.
2. El programa debe realizarse en modo gráfico (GUI), es decir, no se permite en modo consola.
3. Cuando se soliciten o se muestren datos al usuario, no se permite el uso de cuadros de diálogo tipo MessageBox. Para ello se pueden utilizar cajas de texto, etiquetas o listas gráficas según sea. Esto a menos que sea para mostrar excepciones de la aplicación o para dar un mensaje al usuario de que omitió algo en la operación. Si la operación se realiza exitosamente, no se deben mostrar mensajes utilizando estos cuadros de diálogo.
4. Los datos deben persistir en memoria en todo momento hasta que se cierre la aplicación.

Instrucciones:

Se deberá implementar un programa para llevar un registro de los eventos deportivos internacionales de mayor relevancia.

El programa deberá contener las siguientes funcionalidades:

Registro de eventos:

En esta pantalla se podrá incluir los datos generales de cada evento deportivo, con los siguientes campos:

Nombre del campo	Tipo de dato
ID del evento	Identificador consecutivo numérico generado por el programa.
Año del evento	Número entero de 4 dígitos mayor a 1900
Mes del evento	Lista tipo DropDownList con los meses del año.
Día del evento	Lista tipo DropDownList con los días del mes seleccionado (cuando cambia el mes, se refresca esta lista y debe considerar un año bisiesto).
Hora del evento	Lista tipo DropDownList con las horas del día desde las 0 horas hasta las 24 horas, divididas en intervalos de 30 minutos. Ejemplo: 00:00 00:30 01:00

	01:30 02:00 … 23:00 23:30
Deporte	Lista tipo DropDownList con las siguientes opciones: • Fútbol. • Béisbol. • Voleibol. • Básquetbol. • Fútbol Americano. • Balonmano. • Rugby. • Hockey.
Nombre de la competición	Texto
País de la competición	Texto
Equipo A	Texto
Equipo B	Texto

La información de los eventos deportivos con los campos anteriormente especificados, se guardarán en una clase "Evento" y cada una de estas se agregará a una única colección utilizando la clase LinkedList de Java.

Listado de eventos:

En esta pantalla se listan todos los eventos deportivos registrados en la colección (con todos sus campos).
Se deben mostrar las descripciones de los campos, no códigos internos (excepto el ID del evento).
En esta opción se debe utilizar una clase de tipo **Iterator** para recorrer la colección.
Los datos deben ser mostrados en un objeto tipo Table.

Búsqueda y edición de un evento:

Para buscar un evento, el usuario debe seleccionar el año y el mes del evento (usando listas tipo DropDownList como en el registro).
El programa muestra los registros encontrados en ese mes y año, desplegándolos en un objeto tipo Table. El usuario seleccionará un único registro para su edición.
Cuando se seleccione un registro para su edición, se deberá mostrar en pantalla los campos del registro a editar con los valores actuales (mostrar descripciones, no códigos internos).
El campo ID del evento no podrá ser editado.
Habrá un botón GUARDAR para guardar los cambios.

Comparación de valores:

Debe utilizar **recursividad** para esta operación.
El usuario deberá ingresar dos horas del día para realizar la comparación (por ejemplo, 12 y 23, que corresponden a las 12 mediodía y las 11 pm).
El programa realizará una búsqueda recursiva para seleccionar los registros cuyo valor en el campo "Hora del evento" se encuentre dentro del rango de horas indicado por el usuario. Siguiendo el

ejemplo anterior, si la Hora del evento es 4 pm (o las 16:00), este elemento sí debe incluirse dentro de los encontrados, ya que las 16 horas está entre las 12 y las 23 horas indicadas por el usuario.
El programa mostrará la lista con los registros encontrados (se muestran todos los campos de la clase Evento) en un objeto tipo Table (solo descripciones, no códigos internos).
Puede utilizar una estructura de datos aparte para realizar la búsqueda recursiva si lo considera necesario (por ejemplo: un arreglo primitivo).

Ordenamiento de la lista de eventos deportivos:

En esta pantalla se mostrarán los eventos registrados ordenados descendentemente por el campo "Equipo A" o por el campo "Equipo B", según lo indique el usuario de la aplicación.
El ordenamiento debe hacerlo utilizando el algoritmo "Selección" o el "QuickSort", según lo indique el usuario.
Previo al ordenamiento, se deben copiar los datos a ordenar del LinkedList a un arreglo para uso del algoritmo de ordenamiento correspondiente.
Si utiliza el método Sort del LikedList obtendrá un cero en este punto automáticamente.
Finalmente se mostrará la lista ordenada de eventos deportivos con todos sus campos en un objeto o control Table o JTable, según el criterio de ordenamiento indicado.

Debe existir un menú con cada una de las opciones.
Los resultados de las opciones se muestran en la pantalla correspondiente, sin usar mensajes de diálogo para ello.

Rúbrica

NO.	INDICADORES POR EVALUAR	CUMPLIMIENTO		PUNTOS
		Cumple	No cumple	
REGISTRO DE EVENTOS				
1.	La pantalla del registro de eventos incluye los 10 campos indicados en el enunciado con sus correspondientes tipos de dato según lo especificado para cada campo. 1 punto por cada uno.			10
2.	Los 10 campos indicados en el enunciado se almacenan en clases llamadas Evento y cada una de estas se agregará a una única colección utilizando la clase LinkedList de Java. 1 punto por cada uno.			10
LISTADO DE EVENTOS				
3.	En esta pantalla se listan todos los eventos deportivos registrados en la colección de tipo LinkedList (con todos sus 10 campos según el enunciado en el Registro de eventos). Se deben mostrar las descripciones de los campos, no códigos internos (excepto el ID del evento). Los datos se despliegan en un objeto o control tipo Table o JTable. Se utiliza una clase de tipo **Iterator** para recorrer la colección LinkedList. Nota: debe cumplir con todos los enunciados anteriores, sino no, la nota será 0.			10
BÚSQUEDA Y EDICIÓN DE UN EVENTO				
4.	Para buscar un evento, el usuario selecciona el año y el mes del evento (usando listas tipo DropDownList como en el registro de eventos). El programa muestra los registros encontrados en ese mes y año, desplegándolos en un control tipo Table o JTable e incluyendo los 10 campos del evento según el enunciado en el Registro de Eventos.			10
5.	Cuando el usuario selecciona un registro para su edición, se muestran en pantalla los 10 campos del registro a editar con los valores actuales (mostrar descripciones, no códigos internos) y el campo ID del evento no podrá ser editado. 1 punto por cada uno.			10
6.	Al hacer clic en el botón “Guardar”, los 10 campos del registro quedan actualizados.			10
COMPARACIÓN DE VALORES				
7.	Usando recursividad, el programa busca los registros cuyo valor en el campo “Hora del evento” se encuentre dentro del rango de horas indicado por el usuario.			10
8.	El programa mostrará la lista con los registros encontrados (se muestran todos los campos de la clase Evento) en un objeto tipo Table (solo descripciones, no códigos internos).			10

ORDENAMIENTO DE LA LISTA DE EVENTOS				
9.	La pantalla hace ordenamiento descendente de los registros usando el valor del campo “Equipo A” utilizando el algoritmo “Selección” (sin usar el método “sort” de la colección LinkedList) y los muestra incluyendo los 10 campos del evento en un control Table o JTable.			10
10.	La pantalla hace ordenamiento descendente de los registros usando el valor del campo “Equipo A” utilizando el algoritmo “Quicksort” (sin usar el método “sort” de la colección LinkedList) y los muestra incluyendo los 10 campos del evento en un control Table o JTable.			10
11.	La pantalla hace ordenamiento descendente de los registros usando el valor del campo “Equipo B” utilizando el algoritmo “Selección” (sin usar el método “sort” de la colección LinkedList) y los muestra incluyendo los 10 campos del evento en un control Table o JTable.			10
12.	La pantalla hace ordenamiento descendente de los registros usando el valor del campo “Equipo B” utilizando el algoritmo “Quicksort” (sin usar el método “sort” de la colección LinkedList) y los muestra incluyendo los 10 campos del evento en un control Table o JTable.			10
GENERALIDADES				
13.	Se utiliza un menú con cada una de las opciones.			5
14.	Se utiliza el modo gráfico para el registro de eventos			5
15.	Se utiliza el modo gráfico para el listado de eventos.			5
16.	Se utiliza el modo gráfico para la búsqueda y edición de un evento.			5
17.	Se utiliza el modo gráfico para la comparación de valores.			5
18.	Se utiliza el modo gráfico para el ordenamiento de la lista de eventos.			5
19.	No utiliza cuadros de diálogo tipo MessageBox para dar ni para solicitar información al usuario, a menos que sean excepciones del programa o que el usuario omitió algo necesario para la operación que quiere realizar.			5