UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Estructuras de Datos
2020-2 - Tarea Integradora 2

Objetivos

Unidad 3: Algoritmos y Estructuras Recursivas

OE3.1. Calcular la complejidad temporal de algoritmos recursivos.
OE3.3. Resolver ecuaciones de recurrencia que sean resultado del análisis de complejidad temporal de algoritmos recursivos.
OE3.6. Utilizar estructuras recursivas de datos para representar la información del modelo de datos cuando sea conveniente.
OE3.7. Evaluar la utilidad del concepto de orden en un árbol binario de búsqueda para la solución de problemas.
OE3.8. Evaluar la utilidad del concepto de balanceo en un árbol binario de búsqueda para la solución de problemas.
OE3.9. Desarrollar estructuras de datos recursivas ABB.
Opcional: OE3.10. Desarrollar estructuras de datos recursivas R&N.
OE3.11. Desarrollar estructuras de datos recursivas AVL.
OE3.14. Desarrollar las pruebas unitarias de cada una de las estructuras de datos recursivas implementadas.

Gestión Eficiente de Base de Datos de Personas

Enunciado

Después del excelente rendimiento en sus trabajos académicos previos, sus profesores de algoritmos los han recomendado a diversos profesores de la Facultad de Ingeniería, para que trabajen en sus proyectos de investigación. Su equipo ha sido contratado para una monitoría en un proyecto de investigación interna de la Universidad, como parte del Equipo VIP de Simulación[1].

Generación de los Datos

El sub-proyecto que le ha sido asignado, consiste en el desarrollo de un prototipo de software que permita gestionar eficientemente las operaciones CRUD sobre una base de datos de personas de nuestro continente. La población del continente americano se estima, en 2020, en poco más de mil millones de personas[2], representando cerca del 13% del total mundial. Por tanto, usted debe **simular la creación de** (generar) un número similar de registros de personas, para este continente, con los siguientes datos: código (autogenerado), nombre, apellido, sexo, fecha de nacimiento, estatura, nacionalidad y fotografía.

Usted debe desarrollar un programa que lleve a cabo la generación de todos estos registros de personas de acuerdo con las condiciones que se detallan a continuación. Los datasets que el programa utilizará como entrada para generar los datos de las personas deben ser descargados a un directorio del proyecto (/data).

Para la generación de nombres completos, se deben tomar los nombres de este dataset de nombres de data.world. Los apellidos se deben tomar del archivo completo (el archivo pequeño -1000- debe ser ignorado) de este dataset de apellidos de data.world. La combinación de todos los nombres con todos los apellidos debe producir la cantidad (o similar) de personas que deseamos.

La fecha de nacimiento debe ser generada aleatoriamente, suponiendo una distribución de edad para toda América basada en esta distribución de edad de Estados Unidos. La distribución de la población en sexo indicada en el enlace anterior puede ser ignorada, y se puede asumir una cantidad igual de hombres y mujeres.

La estatura debe ser generada aleatoriamente en un intervalo que tenga sentido. La nacionalidad debe ser asignada a cada persona, generada de tal forma que se mantengan los porcentajes relativos de población de cada país respecto del continente de acuerdo con estos datos de población por países (se puede también asignar exactamente la misma población de cada país, y si hay diferencia en el total, dicha diferencia -positiva o negativa- puede quedar en el país con mayor población).

Usted puede filtrar el dataset para dejar en el archivo únicamente los registros de los países necesarios. Su programa debe basar los cálculos en el archivo y no en condicionales por país quemados en el código (hardcode).

---

[1] VIP Curse in Data Intensive Processing and Simulation
[2] https://www.worldometers.info/geography/7-continents/

Bonus: La fotografía debe ser generada aleatoriamente de este sitio: https://thispersondoesnotexist.com/ sin importar que su imagen no se corresponda exactamente con su edad, sexo u otros.

El programa debe tener una opción para empezar a generar los datos siguiendo las especificaciones anteriores. Debe tener una barra de progreso si el proceso tarda más de 1 segundo en terminar, y debe indicar cuánto tiempo se demoró la operación. La opción de generar debe tener un campo de texto en el cual se pueda digitar cuántos registros se desea generar. Por defecto, debe estar en el campo el máximo valor posible.

Una vez los datos se generan, debe haber una opción para guardarlos en la base de datos del programa, y así poderlos consultar posteriormente. Todos los datos del programa deben ser persistentes (es decir, si se cierra el programa, deben seguir allí una vez se inicie nuevamente).

Responda este par de preguntas: (1) ¿Es posible serializar los datos generados? (2) ¿Cuánto pesa el archivo serializado, cuando se persisten los datos, al generarse el número máximo posible de registros?.

Operaciones CRUD

En programación, se conoce como CRUD (Create, Read, Update y Delete) a las cuatro funciones básicas del almacenamiento persistente[3]. El programa desarrollado por su equipo debe tener la posibilidad de llevar a cabo cualquiera de estas funciones. Por tanto, la interfaz con el usuario deberá contar con un formulario para agregar a una nueva persona, otro para buscar a una persona por los criterios mencionados más adelante y uno más para actualizar los campos de una persona existente. Este último formulario debe también tener una opción para eliminar a una persona.

Crear, Actualizar y Eliminar

El formulario para agregar debe tener todos los campos requeridos (menos el código, que es autogenerado) para la información de una persona y la opción de Guardar.

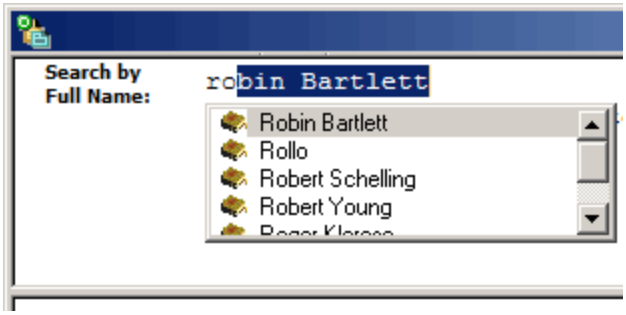
El formulario para actualizar una persona, debe tener todos los campos editables (menos el código, que no se puede actualizar) de información de una persona, cargados con su información actual, la opción de Actualizar (para guardar los cambios, si hubo) y la opción Eliminar (si se desea eliminar a esta persona).

Buscar (la estrella de la solución)

El formulario para buscar debe tener la posibilidad de realizar la búsqueda por cualquiera de los siguientes criterios, de forma excluyente (es decir, buscas por un criterio o por otro, pero no por varios al tiempo):

1. Nombre
2. Apellido
3. Nombre Completo (Nombre + " " + Apellido)
4. Código

El programa debe permitir que, en la medida en que se digite los caracteres de búsqueda en el campo (para los criterios 1 a 3), vayan apareciendo en una lista **emergente** debajo del campo, máximo 100 (parametrizable) nombres de la base de datos, que empiecen con los caracteres digitados hasta el momento.



Ejemplo de la búsqueda y la lista emergente

Para cada una de las cuatro búsquedas, usted debe mantener un árbol binario de búsqueda autobalanceado y persistente. El árbol debe ser implementado completamente por su equipo de desarrollo.

[3] https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

Bonus: si uno de los árboles binarios de búsqueda autobalanceado es un Árbol Rojo y Negro.

Mientras se hace la búsqueda, debe mostrarse al lado del campo donde se digita la cadena, un número que indica la cantidad total de elementos que hasta el momento coinciden con el prefijo digitado.

En el momento en que haya 20 (parametrizable) o menos elementos que coinciden con la búsqueda, debe desplegarse un listado con los registros que coinciden y un botón al lado de cada registro con la opción de Editar, que nos llevará al formulario con la posibilidad de modificar o eliminar ese registro.

El programa debe tener el diseño e implementación de pruebas unitarias automáticas de las estructuras de datos implementadas y de las operaciones principales de las clases del modelo.

El programa debe contar con una interfaz gráfica con el usuario, tener un componente menú en la parte superior de la ventana que permita cambiar todos los elementos de la ventana cada vez que se esté trabajando en opciones diferentes. Es decir, **no** se desea tener una ventana llena al mismo tiempo de elementos que se usarán en momentos diferentes.

Entregas

Usted debe entregar los siguientes artefactos en las condiciones indicadas:

1. **[15%]** Desarrollo completo del Método de la Ingeniería.
    a. La etapa 1 debe incluir la Especificación de Requerimientos Funcionales.
    b. En las etapas 2, 3, 4 y 5 se deben concentrar en el problema de cómo generar sugerencias al digitar texto.
2. **[25%]** Diseño.
    a. **[20%]** Diseño completo del diagrama de clases, incluyendo, las estructuras de datos, el paquete del modelo, de la interfaz con el usuario y pruebas.
    b. **[5%]** Diseño de los casos de la única prueba, incluyendo los escenarios.
3. **[50%]** Implementación.
    a. **[15%]** Implementación completa de las estructuras de datos y sus pruebas.
    b. **[35%]** Implementación completa y correcta del modelo, la ui y las pruebas.
4. **[10%]** Usted debe entregar el enlace del repositorio en GitHub o GitLab con los elementos anteriores. El nombre del repositorio debe estar en inglés, en minúsculas y si tiene varias palabras, éstas van separadas por un guión. Su repositorio debe corresponder con un proyecto de eclipse. Debe tener al menos 10 commits con diferencia de 1 hora entre cada uno de ellos. En el repositorio o proyecto de eclipse debe haber un directorio llamado **docs/** en el cual deberán ir cada uno de los documentos del diseño.

    El **readme.md** del repositorio debe explicar brevemente (en inglés) de qué se trata el proyecto. Es importante que ni en el nombre, ni la explicación del readme hagan referencia a una “tarea” de un curso ni nada parecido, sino que explique de forma independiente al curso, de que se trata el problema y la solución del mismo. Deben enlazar los archivos que documentan el proyecto (en formato pdf) y deben especificar las condiciones técnicas del mismo (lenguaje, sistema operativo, ambiente de desarrollo e instalación). **El último commit en la rama master debe estar marcado con un tag llamado Milestone1**. La fecha del commit marcado con este tag debe ser previo a la fecha y hora límite de la entrega. Usted podrá seguir haciendo commits posteriores a la fecha de entrega, pero la revisión se hará sobre la versión del proyecto marcada con esta etiqueta.

    Recuperación:
    Después de la sustentación de la primera entrega, su equipo tendrá un tiempo (unos días, estipulados en su momento por el profesor), para hacer ajustes y completar los elementos del proyecto que desee si desea que sea evaluado nuevamente. Esa segunda entrega deberá ser etiquetada como **Milestone2**.
    En esta versión recuperatoria del proyecto, al **readme.md** del repositorio debe agregársele una sección que indique concretamente qué elementos fueron agregados o modificados en esta versión (para que quien califique conozca qué es lo nuevo que debe revisar). El título de la sección puede ser changelog. También está la opción de

hacer esta descripción de cambios de la nueva versión en un archivo aparte, **enlazado en el readme**. Así lo haga en un archivo aparte o dentro del mismo readme, por favor siga las recomendaciones en https://keepachangelog.com.

5. **[5%] Bonus** de la imagen de la persona.
6. **[10%] Bonus** de Árbol Rojinegro (incluyendo diagrama de clases, implementación, pruebas y uso en el modelo).

Los requerimientos funcionales, el diagrama de clases y el diseño de casos de prueba deben entregarse en un mismo archivo en formato **pdf**, bien organizado por secciones y títulos, con hoja de portada.

Importante: su repositorio debe ser privado hasta la hora y fecha de entrega, después de la cual usted debe hacerlo público para que pueda ser revisado y compartido a la comunidad que estará ansiosa de conocerlo!.

Nota: La rúbrica con la que se evaluará esta tarea se encuentra en el siguiente enlace Rúbrica TI2. Se recomienda revisar la rúbrica con la que será evaluada su entrega.

Fecha Máxima de Entrega: 2 de Mayo de 2022.