Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI-0112 Programación 1
Examen #2, I-2021
Sábado 17 de julio del 2021
Docente Sivana Hamer

Indicaciones generales

Entrega:

- Cuenta con 3 horas para solucionar el examen.
- Debe desarrollar el examen (desde el inicio hasta fin) en el siguiente repositorio de Github classroom (https://classroom.github.com/a/IN9s0bqu).
- Una vez transcurridas las 3 horas tendrá 5 minutos adicionales para subir un archivo (.pdf) a Mediación Virtual que documente el hash (identificación del commit) que contenga la versión final del examen. El enlace de mediación virtual se cerrará en punto. Los 5 minutos adicionales se otorgan para solventar cualquier problema técnico para la entrega del archivo. En caso de no entregar el archivo, se considerará el examen sin entregar.
- La entrega final debe compilar, tal como está establecido en la carta al estudiante. De lo contrario, será calificado con nota cero.
- Debe subir el examen en Mediación Virtual al enlace correspondiente. En caso de problemas con la plataforma de Mediación Virtual debe mandar el examen por correo (sivana.hamer@ucr.ac.cr). En caso excepcional que fallen ambos medios, debe enviar la docente por medio de un mensaje privado en Telegram. Se debe subir el examen siempre durante el período de entrega del examen.
- Si realiza el examen en una computadora de escritorio que depende de energía eléctrica y sucede un fallo en el suministro, debe continuar su examen a papel. En caso de realizar el examen a papel, debe tomar fotografías de su solución. Toda página debe estar debidamente enumerada e incluir su identificación (e.g., cédula) en las fotografías. Debe subir el examen siempre durante el período de entrega del examen.

Desarrollo:

- Para el examen es obligatorio que se conecten por medio de Zoom con un dispositivo que tenga cámara y audio (como indica la Resolución de la Vicerrectoría de Docencia No. VD-11502-2020). Debe compartir pantalla durante la realización de la prueba (salvo casos previamente definidos con la docente). Debe mantenerse en Zoom durante toda la duración del examen. Se tomará asistencia.
- Durante el examen, la docente indicará cuando se deben realizar commits y el mensaje respectivo. Los commits no tienen que compilar. Realizar el commit correspondiente es obligatorio. Además de hacer commit al cambio tienen que realizar el push respectivo. El examen se considerará inválido si no viene con los commits-push solicitados y tendrá un cero como nota.
- El examen es estrictamente individual. Está prohibido interactuar con cualquier otra persona que no sea la docente. En caso de sospechar lo contrario, se aplicará el debido proceso estipulado en el Reglamento de orden y disciplina de los estudiantes de la Universidad de Costa Rica.
- Debe hacer uso de buenas prácticas para la programación orientada a objetos, por ejemplo: nombres significativos, indentación adecuada, clases con responsabilidades separadas, breve documentación de métodos y clases, y convenciones de nombres.
- Está prohibido utilizar librerías externas o instrucciones de programación no vistas en clases si la docente no lo ha indicado explícitamente. Su uso tendrá una penalización en la calificación final.
- En este examen se permite la reutilización de código propio o aquel provisto en el curso (pero debe acreditarse la autoría). También se tiene derecho a consultar material escrito en internet. Todo código no realizado por él o la estudiante debe encontrarse debidamente referenciado.

- La comunicación entre docente y estudiante durante la realización de la prueba es por medio de Zoom. Para solicitar a la docente, de clic en el botón de "Pedir ayuda".
- Cualquier imprevisto técnico durante el examen deberá comunicarlo inmediatamente a la docente por medio de un mensaje privado en Telegram.
- Si normalmente utiliza el internet de la casa con señal wifi y se va la electricidad, pero posee internet en el celular comuniquese inmediatamente a la docente por un mensaje privado de Telegram.

Recomendaciones:

- Antes de comenzar a implementar código, preocúpense por tener total claridad sobre cada pregunta y solo responda lo que se le pida, ya que responder otros aspectos no permite otorgarle puntuación alguna. Ejemplo: si no se les pide leer, usen constantes, si no se les pide usar el patrón MVC, no tienen por qué hacerlo.
- No se olviden de guardar el código antes de subirlo (en Visual Studio Code es: ctrl + s).
- Si posee una computadora portátil, es una buena alternativa que la utilice para realizar el examen, asegurándose de que la batería está cargada al 100 %, para que en caso de una falla eléctrica pueda continuar el examen. Tengan disponible siempre el cargador.
- Garanticen que su teléfono está con la batería cargada o tiene disponible el cargador.
- Eviten distracciones. Vayan al baño y a comer antes del examen. Intenten encontrarse en una zona con silencio. Avisen a sus familiares que van a realizar un examen.

Pregunta 1 (40 %)

Suponga que poseen dos listas **m** y **n**, conteniendo la cantidad total de campos (celdas) en buffers a manejarse en cada lista, así como un nodo definido de la siguiente forma:

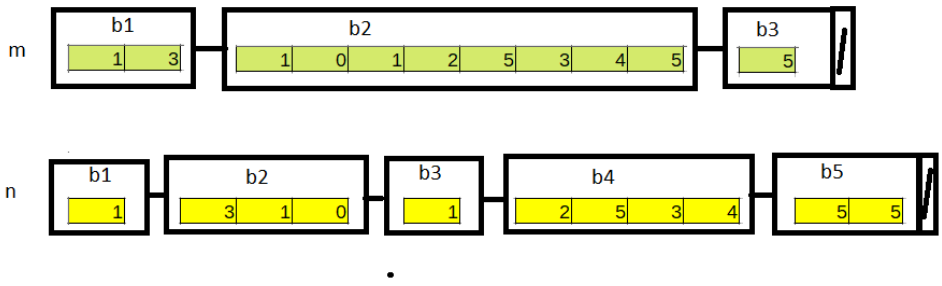
```
class Nodo{
    private String[] buffer; // arreglo de hileras de caracteres
    Nodo next;
    //setters y getters para cada atributo
    //método toString()
}
```

Una vez creado el **buffer** de un nodo, su tamaño **no** se puede modificar de ninguna forma. De igual forma, no se permite cambiar la estructura de la clase nodo.

Cada lista se crea con un número cualquiera e invariable de nodos, y cada nodo puede tener un **buffer** de un tamaño distinto al del **buffer** de otros nodos. Independientemente de lo anterior, se tiene que dos listas son **iguales en tamaño** si la suma del tamaño de sus buffers son iguales. Es decir, para que se cumpla la igualdad de tamaño con las listas **m** y **n**, se cumple que la cantidad total de campos en buffers en esas listas coinciden, es decir:

$$\sum_{i=0}^{\text{nodos de m}} nodo_i(\text{tamaño buffer}) = \sum_{j=0}^{\text{nodos de n}} nodo_j(\text{tamaño buffer})$$

Por ejemplo:



Como parte de este examen, se le pide que programe las siguientes funcionalidades:

1. (5 %) Creación de los atributos, constructores y métodos de las clases requeridas. Su método constructor debe ser general, y debe permitir crear una lista de cualquier tamaño de nodos y cada nodo puede tener buffers con cualquier cantidad de campos. Lo importante acá será no exceder la cantidad total de campos en buffer que se indica como parámetro de entrada al crear la lista.
2. (15 %) Creación de métodos para definir la cantidad de nodos de cada lista, instanciar la lista inicial y mostrar la lista.
3. (20 %) Creación de un método que reciba una lista y copie los contenidos de la lista actual (m) en la lista recibida como parámetro (n). Cualquier método que requiera para copiar la lista lo debe hacer en la clase de la lista.

Pregunta 2 (60 %)

De un grupo de personas se conoce el nombre y la profesión. Se desea implementar una aplicación que permita tener un árbol de búsqueda de nombres de personas que ejercen una profesión determinada, así que debe crearse un árbol de profesiones, donde cada nodo del árbol apunte a una lista simplemente enlazada dinámica de objetos de tipo Persona.

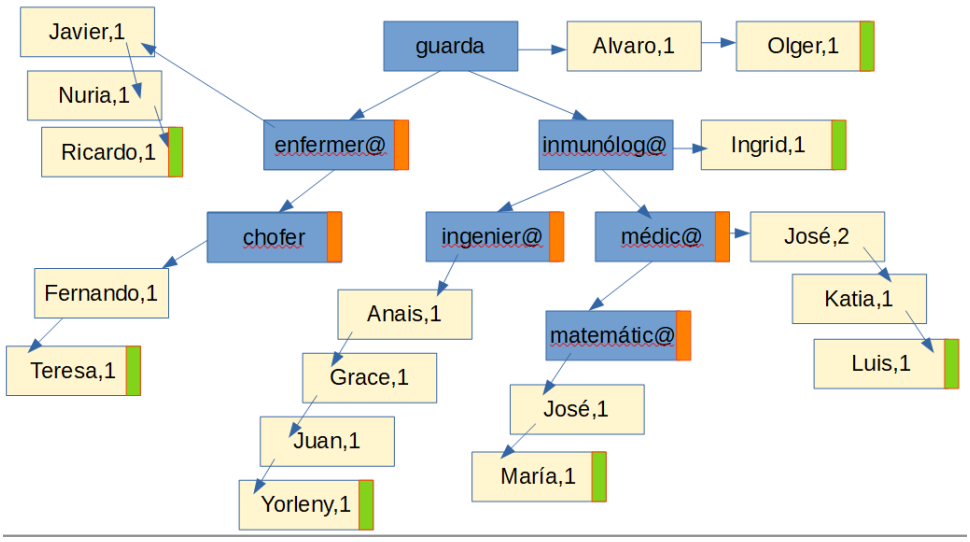
1. (5 %) Cada persona estará compuesta por un nombre y un contador de veces que se encuentra en la lista, y los métodos necesarios para poder interactuar con la clase.

Al insertar las personas en el árbol, las mismas deben **agregarse ordenadamente** en la lista del respectivo nodo de profesión. Si se inserta una persona duplicada para una profesión, entonces deberá incrementarse el contador en el objeto persona para saber, ¿cuántas personas hay con el mismo nombre?. El árbol sólo debe tener las profesiones que aparecen en la lista, sin embargo, su implementación no deberá restringirse a los datos de esta lista. Es decir, si la lista recibida es distinta, se podrá generar el árbol sin problemas.

2. (20 %) Cree un proceso que genere el árbol junto con sus listas, use como base la siguiente matriz:

```
String m[][]=new String[][]{
    {"Alvaro", "guarda"},  {"Katia","médic@"},
    {"Ricardo","enfermer@"},  {"Juan","ingenier@"},
    {"Teresa","chofer"},  {"Maria","matemátic@"},
    {"Ingrid","inmunólog@"},  {"Olger","guarda"},
    {"Luis","médic@"},   {"Nuria","enfermer@"},
    {"Grace","ingenier@"},  {"Fernando","chofer"},
    {"José","matemátic@"},  {"José","médic@"},
    {"Javier","enfermer@"},  {"Yorleny","ingenier@"},
    {"José","médic@"}, {"Anais","ingenier@"}
};
```

A modo de ejemplo, usando los datos en la matriz anterior, las estructuras de datos, se podrían representar de la siguiente manera:



3. (15 %) Usando la información en el árbol binario de búsqueda, programe un método capaz de retornar la lista de personas profesión X.

4. (20 %) Programe la funcionalidad que, a partir del **nombre de una persona** junto con su **profesión**, sea capaz de modificar los datos del nodo del árbol o el nodo mismo. El algoritmo debe buscar dentro del elemento respectivo del árbol si el nombre existe en la lista de personas y proceda a decrementar en uno el contador de personas. Si sólo hay una persona con ese nombre, deberá eliminarse de la lista. A su vez, si el nodo de profesiones queda sin personas, deberá eliminar también el nodo del árbol.