Proyecto 1er parcial POO-Agenda de contactos

El objetivo es implementar un administrador básico de contactos. Es decir un sistema donde una persona pueda guardar datos básicos de otras personas.

El sistema debe permitir dar de alta, borrar y buscar un contacto.

Implementación. A continuación se describen algunas de los elementos clave que debe tener el programa. **OJO** es altamente importante que se respete los nombres (Inclusive mayúsculas y minúsculas) de clases, atributos y métodos que se indiquen. Cada clase se debe implementar en un archivo diferente.

Clase Contacto.

Contacto
- nombre: String - apellidos: String - categoria: String - correo: String - telefono: String - celular: String - direccion: String - nota: String
+ Contacto(String, String, String, String, String, String, String, String) + getter de cada atributo + toString( ): String + contiene(String):boolean

Esta clase representa a cada una de las personas que se van a guardar en el sistema. Ningún atributo deberá validarse y el atributo ***nota*** se refiere simplemente a cualquier comentario libre que quiera escribir y el atributo ***categoria*** se puede utilizar para clasificarlos como “Familia”, “Amigos”, etc.

Método **toString**: Regresaría un String con los datos del contacto. Un campo por línea escribiendo primero la descripción del campo, luego dos puntos y finalmente el valor capturado, si algún campo está vacío entonces ese campo no se muestra. Ejemplo de un String a regresar:
Nombre: Juan
Apellidos: Pérez
Categoría: Amigos
Celular: (33)11330011

En este ejemplo no se muestran los campos correo, teléfono, dirección y nota porque esos campos están vacíos.

Método **contiene**: Esté método busca si un String está contenido en alguno de los campos del contacto. Si está en cualquiera de los campos regresa true caso contrario false.

Clase Agenda

Agenda
- contactos: Contacto[ ] - numContactos: int
+ Agenda( ) + Agenda(int) + agregarContacto(Contacto): void + borrarContacto(int): boolean + buscarContacto(String):void

Esta clase representa al lugar donde se almacenan todos los contactos con los que cuenta el sistema.

El atributo ***numContactos*** almacena cuántos contactos hay dados de alta en el sistema. Recuerda que no necesariamente el arreglo está lleno, es decir puede tener espacios vacíos. Por ejemplo, suponga que el arreglo es de tamaño 10 pero apenas he dado de alta 1 contacto, en ese caso ***numContactos*** vale 1. Y luego si doy de alta otro contacto entonces ***numContactos*** se incrementa y vale ahora 2.

El constructor por default inicializa ***contactos*** en un tamaño de 10 y ***numContactos*** en 0. Es decir, la agenda se acaba de crear y está vacía pero tiene espacio suficiente para guardar 10 contactos.

El constructor con parámetro recibe la capacidad inicial de la agenda es decir en lugar de inicializar contactos en 10 se inicializa en el tamaño que nos hayan pasado como parámetro.

Método **agregarContacto**: Recibe como parámetro el objeto contacto a dar de alta en la agenda. Este contacto deberá almacenarse en orden alfabético respecto al ***apellido*** de los contactos. Es decir el contacto deberá insertarse en la posición que le corresponda en el arreglo por orden alfabético y si hay dos personas con el mismo ***apellido*** entonces deberá considerar también el ***nombre*** del contacto. En el caso que el arreglo ya esté lleno se deberá aplicar la técnica de ampliación dinámica del arreglo.

Método **borrarContacto**: borra un contacto de la lista. El contacto a borrar es el que se localice en la posición que se recibe como parámetro. Si queda un hueco en el arreglo se debe de recorrer los elementos delante de él para quitar el hueco y no olvides actualizar ***numContactos***.

Método **buscarContacto**: Busca todos los contactos que contienen en alguno de los campos el String que se pasa como parámetro y despliega la información de aquellos contactos que cumplan. **Si se pasa como parámetro un String vacío entonces deberá desplegar todos los contactos que hay almacenados. Además la primer línea a imprimir de cada contacto a desplegar será la posición que ocupa en el arreglo. Ejemplo:**

3
Nombre: Juan
Apellidos: Pérez
Categoría: Amigos
Celular: (33)11330011

Clase Principal

Principal
- imprimeMenu( ): void *- pideContacto( ): Contacto* *+ main(String[ ]): void*

Esta es la clase que se manda a llamar para ejecutar el programa.

Método **imprimeMenu**: Imprime las opciones que puede ejecutar el programa: 1) Agregar Contacto, 2)Buscar contacto, 3)Borrar contacto 4)Salir.

Método **pideContacto**: Pregunta al usuario los datos para cada uno de los campos de un contacto nuevo. Crear el nuevo contacto y lo regresa.

Método **main**: Comienza desplegando el menú y esperando a que el usuario seleccione una opción. Una vez que el usuario seleccionó alguna, el programa hace la acción requerida. Al finalizar la opción se regresa al menú para que el usuario seleccione otra opción. Así hasta que el usuario seleccione la opción salir.

Fecha límite de entrega: lunes 02 de octubre a las 8:00 hrs.

Notas

- Sólo uno de los dos integrantes del equipo debe subir el proyecto.
- Los archivos de los códigos fuentes deben contener los nombres y matrículas de los integrantes del equipo así como el grupo al que pertenecen.
- Recomiendo investigar de la clase String el método indexOf(String).