Prac01 - Agenda de Contactos.

version 1.0 - 7/03/2021

Descripción del Problema

Se trata de desarrollar una agenda de contactos mediante un diseño pautado.

1. Contacto.

Material a entregar

- Contacto.java
- ContactoTest.java

Implementar un tipo abstracto de datos Contacto junto con sus pruebas.

```
TAD Contacto

- nombre : String
- tf : String
- email : String

+ toString () : String
  POST: Retorna los datos del Contacto como texto.

+ nombre () : String
  POST: Da el nombre de este Contacto.

+ tf () : String
  POST: Da el tf de este Contacto.

+ email () : String
  POST: Da el email de este Contacto.

+ ponerNombre (nombre : String)
  EFECTO: Modifica este contacto poniendo <nombre> como
          nuevo nombre.

+ ponerTf (nombre : String)
  EFECTO: Modifica este contacto poniendo <tf> como
          nuevo teléfono.

  ponerEmail (email : String)
  EFECTO: Modifica este contacto poniendo <email> como
          nuevo email.

+ equals (c : Contacto) : boolean
  POST: Determina si <c> es igual a este Contacto.

+ clone () : Contacto
  POST: Da una copia de este Contacto.

+ linea () : String
  POST: Convierte este Contacto en una línea de texto
        con los atributos separados por el carácter ';'.

+ static leer () : Contacto
  POST: Crea un Contacto con datos leídos de la entrada
        estándar (input) y lo entrega como resultado.
```

2. Lista de Contactos.

Material necesario

- ColDeContactosTest.java
- ColDeContactosJugar.java

Material a entregar

- ColDeContactos.java

Se trata de implementar el TAD *ColDeContactos*, especificado a continuación, como una colección (acotada) de contactos. Utiliza las pruebas de *ColDeContactosJugar* y *ColDeContactosTest*. Añade todas las pruebas que te vengan bien.

```
TAD ColDeContactos

- elementos : [0+]Contacto
- longitud : int

+ ColDeContactos ()

+ ColDeContactos (maxNumContactos : int)

+ toString () : String
  POST: Retorna la ColDeContactos como texto.

+ size () : int
  POST: Da el número de contactos que hay en la colección.

+ get (pos : int) : Contacto
  PRE: pos IN [0, longitud-1]
  POST: Da el Contacto que está en <pos> en la colección.

+ set (pos : int; elem : Contacto)
  PRE: pos IN [0, longitud-1]
  EFECTO: Añade <elem> a la colección, poniéndolo en la
          posición <pos>, sin abrir hueco, es decir,
          cambiando lo que hubiera por <elem>.

+ add (elem : Contacto)
  EFECTO: Añade el elemento <elem> a la colección, poniéndolo
          al final.

+ add (pos : int, elem : Contacto)
  PRE: pos IN [0, longitud]
  EFECTO: Añade el elemento <elem> a la colección, poniéndolo
          en la posicion <pos>, abriendo hueco.

+ indexOf (elem : Contacto) : int
  POST: Devuelve la posicion donde se encuentra <elem>.
        Si no se encuentra, el resultado es -1.

+ removeElementAt (pos : int)
  PRE: pos IN [0,longitud-1]
  EFECTO: Elimina el elemento que está en la posición <pos>.
          Mueve todos los elementos en [pos+1, size()-1] un
          lugar a la izquierda. El objeto pasa a tener un
          elemento menos.

+ remove (elem : Contacto) : boolean
  EFECTO: Si existe un elemento igual a <elem> en la
          colección, lo elimina. En caso contrario no hace nada.

+ maxSize () : int
  POST: Da la capacidad máxima de la colección.
```

3. Agenda.

Material necesario

- AgendaTest.java

Material a entregar

- Agenda.java

Se trata de implementar el TAD *Agenda*, junto con sus pruebas, especificada a continuación como una colección de operaciones sobre la agenda de contactos. Utiliza las pruebas de *AgendaTest.* Completa las pruebas que falten.

```
TAD Agenda

- ldc : ColDeContactos

+ Agenda ()

+ toString () : String
  POST: Retorna la Agenda como texto.

+ estaVacia () : boolean
  POST: Determina si esta Agenda está vacía.

+ estaLlena () : boolean
  POST: Determina si esta Agenda está llena.

+ anhadirContacto (c : Contacto)
  EFECTO: Añade el Contacto <c> a esta Agenda,
          poniéndolo al final.

+ size () : int
  POST: Retorna el tamaño de la agenda.

+ get (pos : int) : Contacto
  PRE: pos IN [0, size()-1]
  POST: Retorna el elemento de la agenda que está
        en la posición <pos>.

+ set (pos : int; elem : Contacto)
  PRE: pos IN [0, size()-1]
  EFECTO: Modifica el elemento de la colección que
          está en la posición <pos> poniendo <elem>.

+ nombresContactos () : String
  POST: Retorna un listado de los nombres que están
        en esta Agenda, cada nombre en una línea.

+ buscarNombre (nombre : String) : Contacto
  POST: Devuelve el primer Contacto de esta Agenda
        cuyo nombre sea <nombre>. Si no se encuentra,
        devuelve un Contacto nulo.

+ buscarTelefono (tf : String) : Contacto
  POST: Devuelve el primer Contacto de esta Agenda
        cuyo telefono es <tf>. Si no se encuentra,
        devuelve un Contacto nulo.

+ modificarContacto (nombre : String;
                      tf : String)
  EFECTO: Modifica el Contacto de esta Agenda cuyo
          nombre es <nombre> cambiando su tf por <tf>.
          Si no se encuentra, no hace nada.

+ quitar (contacto : Contacto)
  EFECTO: Elimina de esta Agenda el elemento <contacto>. Si no existe no hace nada.
```

4. Agenda Ampliada. (opcional)

Material necesario

- AgendaAmpliadaTest.java

Material a entregar

- AgendaAmpliada.java
- AgendaAmpliadaTest.java

Se trata de implementar la AF *AgendaAmpliada*, junto con sus pruebas, especificada a continuación como una colección de operaciones sobre la agenda de contactos. Todas las operaciones son de clase (*static*). Utilizar las clase *In* y *Out* de la librería *stdlib*. Ambas clases se pueden descargar de la carpeta *stdlib* enlazada desde el *GoogleSite –> materiales*. Utiliza las pruebas de *AgendaAmpliadaTest.java*. Añade las pruebas que te vengan bien.

```
AF AgendaAmpliada

+ leer (archivo : String) : Agenda
  EFECTO: Crea una Agenda leyendo los datos de un fichero de
          texto de nombre <archivo> que tiene los datos
          guardados.
          Cada línea tiene un contacto con el formato:
          <nombre>;<tf>;<email>

+ escribir (agenda : Agenda; archivo : String)
  EFECTO: Guarda la <agenda> en un fichero de texto de nombre
          <archivo> con el formato:
          <nombre>;<tf>;<email>

+ ordenar (agenda : Agenda) // agenda es INOUT
  EFECTO: Ordena la <agenda>.
```

5. AgendaApp. (opcional)

Material necesario

- agenda.pdf

Material a entregar

- AgendaApp.java

Se trata de implementar una aplicación que tenga las funcionalidades descritas en el documento *agenda.pdf* mas las de cargar la agenda a partir de un archivo, guardar la agenda en un archivo y ordenar la agenda. Puedes añadir alguna otra mas a tu criterio. La aplicación presentará un menú de texto con todas las opciones numeradas para que el usuario elija la opción que quiere ejecutar.