INSTRUCCIONES

- Trabajo en grupos de 2 personas máximo.
- Enviar la solución al tecDigital EVALUACIONES / TAREAS. Revise que el trabajo enviado sea el correcto y a este destino. El proyecto se debe enviar como un archivo comprimido .zip de nombre TAREA2_SUS_NOMBRES.zip.
- Fecha de entrega: 6 de setiembre 11:30pm
- Se coordinará día y hora para revisar la evaluación junto con los estudiantes quienes siendo los autores mostrarán el dominio de la solución implementada desde el punto de vista técnico (uso de conceptos de programación y del lenguaje) así como de la funcionalidad (lo que hace la solución). La revisión puede constar de las siguientes actividades:
    - Revisar esta solución particular
    - Revisar conceptos incluidos en la evaluación
    - Aplicar otras actividades con una complejidad igual o menor a la evaluación.

La tarea va a tomar la práctica que se ha venido trabajando en el proyecto sistemaAcademico sobre definición de clases y creación de instancias (objetos). Vamos a incrementar la funcionalidad del proyecto agregando una matriz de una dimensión (vector) de tamaño dinámico. Cada elemento del vector es una instancia de la clase Estudiante. Para ello realizar cada uno de los siguientes pasos:

1- Modificar proyecto de práctica anterior: sistemaAcademico

2- Usar paquete: universidad.estudiantes

3- Definir otra clase: ListaEstudiante

3.1- Atributos (5P): lista de objetos tipo Estudiante (clase creada en práctica anterior) – Usar un vector dinámico (List)

```
private List<Estudiante> listaEstudiante; // atributo
```

3.2- Métodos:

constructor (5P): crea instancia de la lista (ArrayList)

```
public ListaEstudiante(){
    listaEstudiante = new ArrayList<>(); // crear instancia de lista y dejarla en variable
}
```

agregarAListaEstudiante (20P): recibe por parámetro objeto Estudiante para agregarlo a la lista, el atributo identificación en los estudiantes es único, el estudiante se agrega sino existe su identificación en la lista y se retorna true, de lo contrario retorne false. Use el método buscarEstudiante para determinar si el estudiante existe en la lista.

buscarEstudiante (10P): recibe por parámetro identificación del objeto para buscarlo en la lista según su identificación, retorna el índice si existe, de lo contrario retorna -1

eliminarDeListaEstudiante (15P): recibe por parámetro objeto para eliminarlo de la lista. Si pudo borrarse retorna true de lo contrario retorna false.

desplegarListaEstudiante (15P): despliega objetos de la lista usando método toString de Estudiante. Para recorrer la lista use for-each.

4- En la clase main (30P)

4.1- Crear una variable de referencia para objetos de la clase ListaEstudiante

```
12      // crea variable de referencia para objetos ListaEstudiante
13      ListaEstudiante listaEstudiante;
```

4.2 Crear objeto para la lista (new)

4.3- Crear una variable de referencia para objetos de la clase Estudiante (Estudiante estudiante)

```
18      // crea variable de referencia para objetos Estudiante
19      Estudiante estudiante;
```

4.4- Crear los 3 objetos indicados en práctica anterior (new). Cada vez que se crea un objeto se deja en la variable de referencia del punto 4.3 y se agrega a la lista de objetos

4.5- Probar que no se agreguen a la lista identificaciones repetidas (agregar un objeto que ya está en la lista)

4.6- Desplegar la información de cada estudiante en la lista

4.7- Eliminar de la lista algún objeto, enviar al método la identificación a borrar

4.8- Probar que no se traten de eliminar de la lista identificaciones que no existen (eliminar objeto que no está en la lista)

4.9- Desplegar nuevamente la información para verificar la eliminación