

## ALGORITMOS Y ESTRUCTURAS DE DATOS

### Examen Evaluativo 3

Nombre: \_\_\_\_\_ Cuenta: \_\_\_\_\_

UNAH

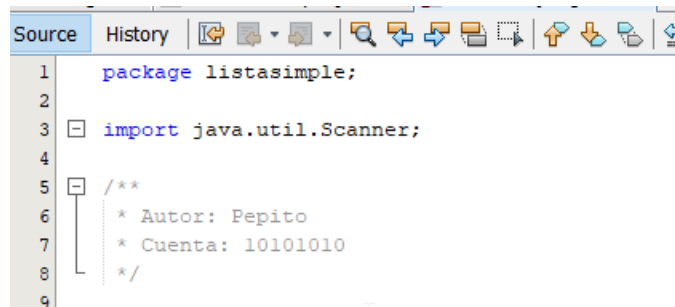
Ing. en Sistemas

## Instrucciones

1. Desarrolle, empleando la herramienta vista en clase, los ejercicios abajo descritos en un proyecto para los ejercicios de LISTAS y otro para el de LISTAS DOBLES. Asegurarse de que exportan el proyecto para simplificar su revisión (podrá castigarse la falta de este aspecto).

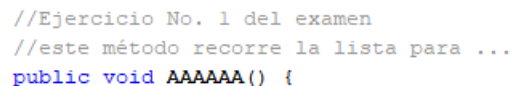
### 2. Documentación

- a. Cada archivo del proyecto deberá estar identificado con su nombre y número de cuenta



```
1 package listasimple;
2
3 import java.util.Scanner;
4
5 /**
6  * Autor: Pepito
7  * Cuenta: 10101010
8  */
9
```

- b. Cada método solicitado (u otros creados que no son parte del TAD), deben de identificar a que número de ejercicio corresponde y además agregar una descripción de lo que hace; indicando por lo menos los argumentos que recibe y la salida del mismo.



```
//Ejercicio No. 1 del examen
//este método recorre la lista para ...
public void AAAAAA() {
```

***\*\*esto es obligatorio y podrá castigarse con puntos la falta de éste aspecto.***

3. Al finalizar, suba al espacio en la plataforma un archivo comprimido con el proyecto; en el caso de que la plataforma fallara por alguna razón, entonces enviarlo al correo electrónico.
4. Este examen es de carácter individual y está sujeto a las regulaciones académicas de la Universidad y de la clase.
5. Usted podrá entregar el examen hasta 30 minutos después de la hora límite indicada, pero esto será un castigo directo a su calificación de hasta 25pts del valor obtenido en el examen.

## Ejercicios Árboles (60%)

---

Tome uno de los proyectos hechos en la temática de árboles (AB o ABB), que esté funcionando, y realice sobre él mismo los siguientes métodos:

1. Se pide implementar la función `esABB(a)` que devuelve un booleano indicando si su argumento `a` (que es cualquier árbol binario de números naturales) es un árbol binario de búsqueda.
2. Escribir un método/función que retorne el número de nodos que tienen dos hijos. Por ejemplo en el árbol de la *figura 1* existen 5 nodos.

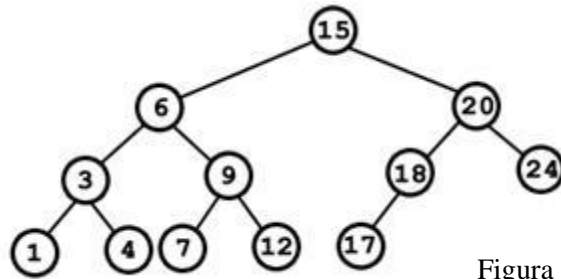


Figura 1

3. Hacer la función `NIVEL_LLENO` que reciba como parámetro un número y verifica si existen todos los nodos que debería haber en ese nivel. En la figura 1 el nivel 3 no está lleno.