Programación III

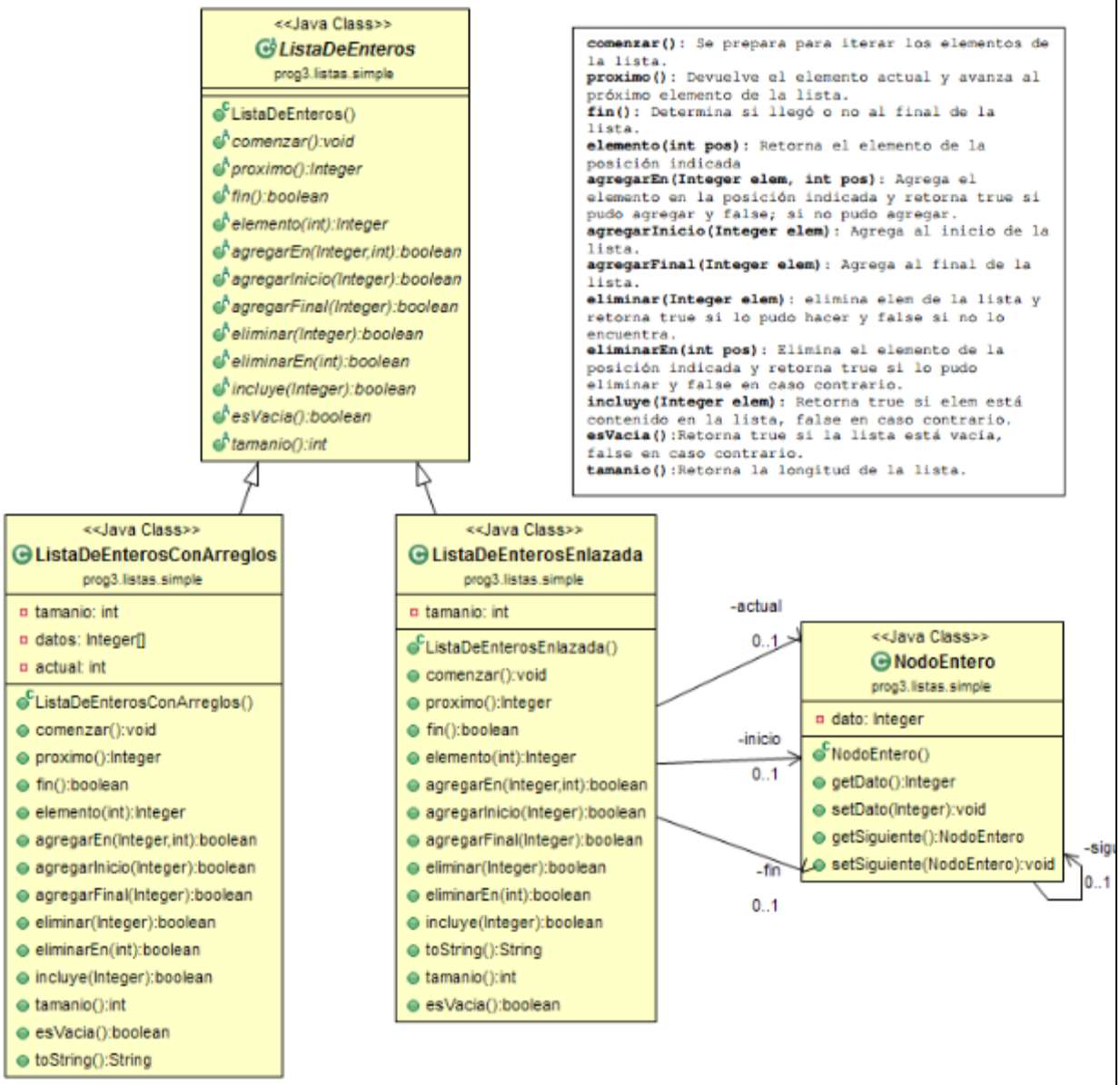
TEMA 3: Listas de Enteros

Práctica nº 3 - A

Tema: Abstracción, Encapsulamiento, Herencia, Tipos Genéricos. Listas.

Importante

- Se recomienda trabajar a partir de esta práctica en un mismo proyecto (podría llamarlo **Programacion3**).
- En esta práctica necesitará la implementación de la cátedra de **ListaDeEnteros**. En particular se trabajará con la implementación de una subclase, la **ListaDeEnterosEnlazada**. Agregue estas clases es su proyecto **Programacion3.**



1. **Acerca de la implementación propuesta, responda:**
    a. ¿Podría ponerle comportamiento a algún método de la superclase **ListaDeEnteros**?
    b. ¿Indique 2 motivos por los cuales la clase ListaDeEnteros se define como abstracta? Note que una subclase implementa la lista usando un arreglo de tamaño fijo y la otra usando nodos enlazados.
    c. ¿Cuál es el motivo por el cual las subclases no compilan? Haga lo necesario para que las dos subclases compilen.
    d. Escriba una clase llamada **ListaDeEnterosEnlazadaTestBasico** en el paquete prog3.lista.simple.test que reciba en su método main una secuencia de números, los agregue a un objeto de tipo ListaDeEnterosEnlazada y luego imprima los elementos de dicha lista.

2. **Método "ordenar" de ListaDeEnterosEnlazada.** Implemente un método llamado ordenar que devuelva una nueva lista ordenada usando el clásico método de la burbuja (consiste en seleccionar el menor elemento de la lista y colocarlo al final de la lista resultado). Como precondición la lista original contiene valores mayores a 0 y todos son diferentes. Tenga en cuenta que la lista original NO debe modificarse.
    a. La firma del método deberá ser la siguiente:

```
public ListaDeEnterosEnlazada ordenar();
```

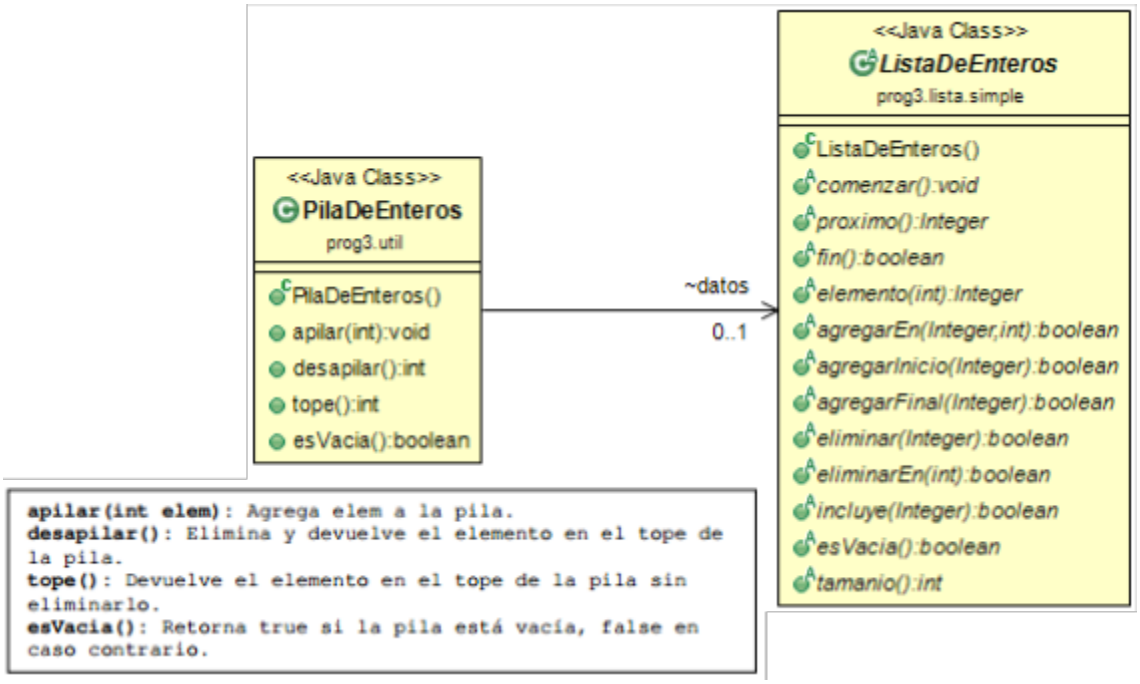
    b. Escriba una clase llamada TestOrdenamiento, que permita verificar el correcto funcionamiento de *ordenar*.
    c. Indique ¿cuántos elementos recorrió para generar la nueva lista resultante?

3. **Método "combinarOrdenado" de ListaDeEnterosEnlazada.** Implemente un método llamado combinarOrdenado que reciba 1 lista de elementos ordenada y devuelve una lista también ordenada conteniendo los elementos de las 2 listas. Como precondición, la lista que recibe el mensaje "combinarOrdenado" también deberá estar ordenada
    a. La firma del método deberá ser la siguiente:

```
public ListaDeEnterosEnlazada combinarOrdenado(ListaDeEnterosEnlazada listaParam);
```

    b. Escriba una clase llamada TestCombinarOrdenado, que permita verificar el correcto funcionamiento de *combinarOrdenado*.
    c. Indique ¿cuantos elementos recorrió para generar la nueva lista resultante?

4. **Pila de enteros.** Considerando el siguiente diagrama:



a. Implemente la clase **PilaDeEnteros**
b. Cree una clase llamada **TestPila**. En su método main, escriba el siguiente código e indique cuál es el resultado. **JUSTIFIQUE.**

```
PilaDeEnteros p1, p2;
int valor2=0;
p1=new PilaDeEnteros();
p1.apilar(1);
p1.apilar(2);
p2=p1;
valor2 = p2.desapilar();
System.out.println("El valor del tope de la pila p1 es: " +
p1.desapilar());
```

5. JUnit (OPCIONAL)

- Descargue del sitio https://github.com/junit-team/junit/releases el achivo .jar (librería recomendada version 4.7) correspondiente a JUnit ó descarguelo de la página de la cátedra.
- Incluya dicha librería en su proyecto (cree una carpeta lib de modo que la librería quede dentro de su proyecto)
- Ejecute la clase ListaDeEnterosEnlazadaJUnitTest y verifique que los Test se ejecutan exitosamente.