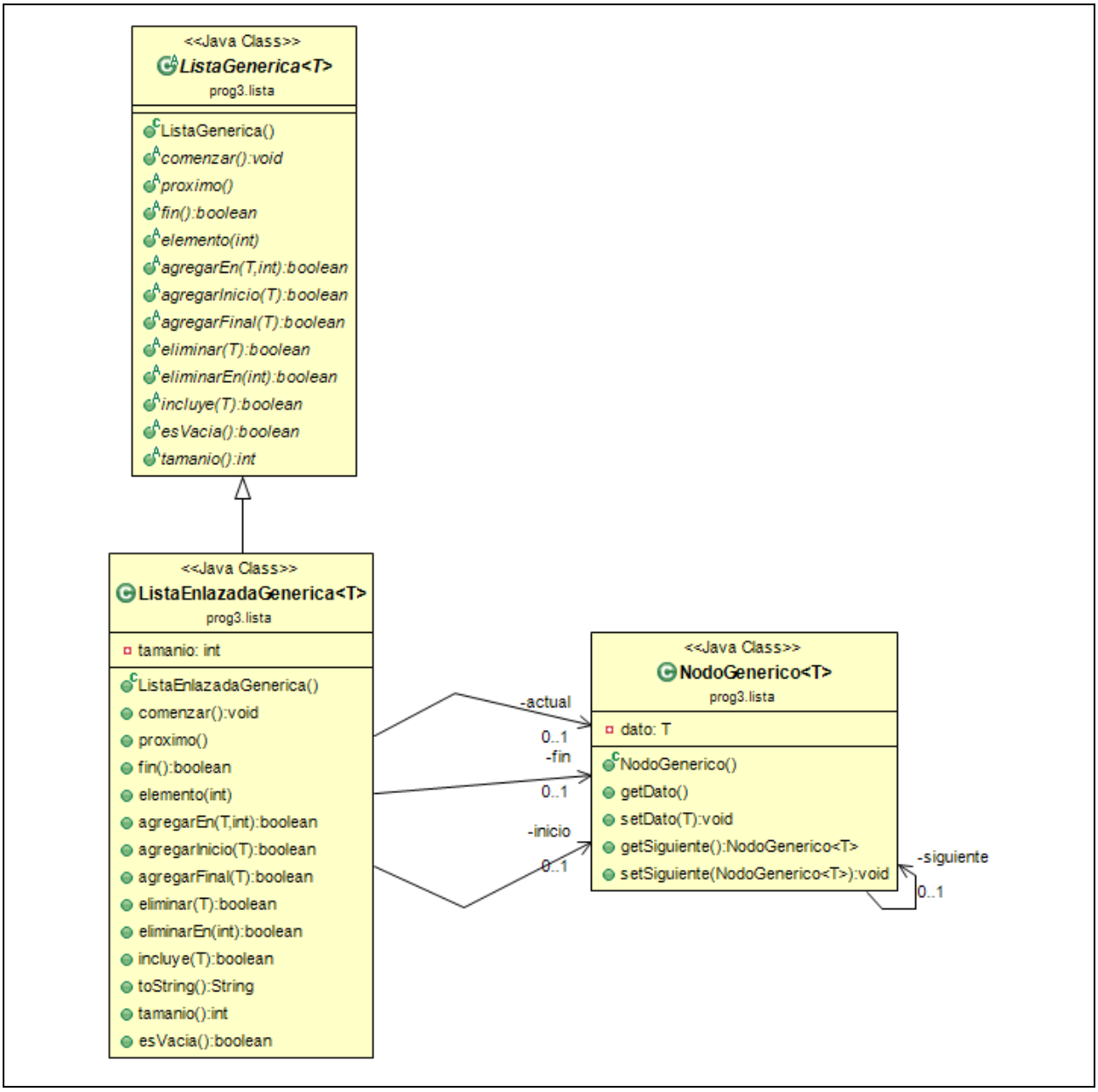
Programación III
TEMA 3: Listas Genéricas
Práctica nº 3 - B

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

1. **Lista Genérica.** Continúe trabajando en su proyecto Programacion3 e implemente la clase ListaGenérica en el paquete prog3.lista. Recuerde que el tipo de dato que almacena es genérico (recién al momento de instanciación se indica el tipo de dato con el que se va a trabajar. Debe respetar los métodos indicados en el diagrama. Tenga en cuenta que la operatoria de la lista genérica es similar a la lista de enteros; **NO** trabaja con un tipo de datos específico sino con un tipo genérico (**T**).



2. **Método "invertir" de ListaEnlazadaGenerica.** Implemente un método de instancia en la clase ListaEnlazadaGenerica que devuelva una nueva lista pero con los elementos invertidos.
   a. La firma del método deberá ser la siguiente:

   ```
   public ListaGenericaEnlazada<T> invertir();
   ```

   b. Indique ¿cuantos elementos deberá recorrer para generar la nueva lista resultante?

3. Analice el código del método **agregarFinal()** y **agregarEn(T elem, int pos)** de la clase ListaEnlazadaGenérica provista por la cátedra e indique
   a. ¿cuántos elementos deberá recorrer para insertar un elemento en la lista con el método **agregarFinal(T elem) ?**
   b. ¿cuántos elementos deberá recorrer -en el peor de los casos- para insertar un elemento en la lista con el método **agregarEn(T elem, int pos) ?**

4. **Pila Genérica.**
   a. A partir de la clase PilaDeEnteros, cree una nueva implementaciòn de pila pero que admita cualquier tipo de elemento, es decir, implemente la clase PilaGenerica. Puede implementar su clase PilaGenerica en el paquete prog3.util.

   b. Escriba una clase que sirva de **Test** para su implementación, donde pruebe su pila con elementos de tipo "**Character**".
      i. Agregue a la pila los caracteres 'a', 'b', 'c', 'd', 'e'
      ii. Saque de la pila 4 elementos del tope
      iii. Imprima en pantalla el elemento en el tope de la pila (debería imprimir 'a').

5. **String de caracteres.** Considere que un string de caracteres {[]} está balanceado si cada símbolo de apertura se corresponde con un símbolo de cierre de manera "armoniosa". Por ejemplo, "{( ) [( )] }" está balanceado, pero "([ )]" no lo está.
   a. Proponga una solución a la verificación del balanceo de un String e indique qué estructura de datos de las implementadas en esta práctica elegiría.
   b. Cree una clase llamada **TestBalanceo** en el paquete **prog3.complementos** e implemente un método cuyo objetivo es determinar si un String dado está balanceado o no. Para implementarlo, deberá recorrer el String caracter por caracter y **usar la estructura de datos elegida** en el punto a.

   Nota: puede usar el método charAt(..) de la clase String para recuperar cada caracter del String original.

6. **CALCULADORA CON NOTACION POLACA INVERSA (RPN).** La notación polaca inversa no es más que otra forma de escribir las expresiones mátemáticas que normalmente usariamos con notación infija. Por ejemplo, la expresión 5* (6+2) -12/4 se convertiría así a RPN:

5* (6+2) -12/4 -> 5*[6,2,+]-[12,4,/] -> [5,6,2,+,*]-[12,4,/] -> **5,6,2,+,*,12,4,/,-**

a. Escriba una clase llamada TestRPN en el paquete **prog3.complementos** e implemente un método que dado un String que representa una expresión en RPN la evalúe, es decir, que resuelva la ecuación y devuelva el valor resultante.

Para los ayudantes...
5,6,2,+,*,12,4,/,-

Símbolo	**examinado Pila**
5	**5**
6	**5,6**
2	**5,6,2**
+	**5,8**
*****	**40**
12	**40,12**
4	**40,12,4**
/	**40,3**
-	**37**

Esto que sigue es sòlo extra...para revisar si es necesario....

~~**El "borrado perezoso" en una lista consiste en lo siguiente: para borrar un elemento sencillamente lo marcamos como borrado (usando un campo booleano). El número de elementos borrados y no borrados se guarda como parte del tipo lista. Si hay tantos elementos borrados como no borrados atravesamos la lista entera, borrando realmente los nodos marcados como borrados. Discutir las ventajas y desventajas del "borrado perezoso". Escribir la implementación del resto de las operaciones de las listas usando "borrado perezoso".**~~

El "problema de Josefo" es el siguiente juego: se sientan n personas, numeradas de 1 a n, formando un círculo. Se pasa una patata caliente empezando en la persona 1. Después de pasar m veces la patata, se elimina la persona que tiene la patata, el círculo se estrecha y el juego continúa, tomando la patata la persona sentada después de la que ha sido eliminada. La persona que se mantiene hasta el final gana. Es bastante habitual considerar que m es un parámetro inicial del juego, aunque se puede usar un generador de números aleatorios para cambiar m tras cada eliminación. Diseña un programa que simule el problema de Josefo.

7. **Lista Circular.**
   - **a.** Cree un paquete llamado **info.ayed.lista.circular**
   - **b.** Copie las clases `ListaGenericaEnlazada` y `NodoGenerico` en el nuevo paquete.
   - **c.** Usando el eclipse, renombre la clase `ListaGenericaEnlazada` por `ListaGenericaEnlazada` `ListaCircularGenericaEnlazada`.
   - **d.** Realice las modificaciones necesarias para que la nueva lista se comporte como una lista circular.
   - **e.** Escriba una clase que sirva de Test para su implementación.

8. **Lista Doblemente enlazada.**
   - **a.** Cree un paquete llamado **info.ayed.lista.doble**
   - **b.** Copie las clases `ListaGenericaEnlazada` y `NodoGenerico` en el nuevo paquete.
   - **c.** Usando el eclipse, renombre la clase `ListaGenericaEnlazada` por `ListaGenericaDoblementeEnlazada`.
   - **d.** Realice las modificaciones necesarias para que la nueva lista se comporte como una lista doblemente enlazada.
   - **e.** Escriba una clase que sirva de Test para su implementación.