



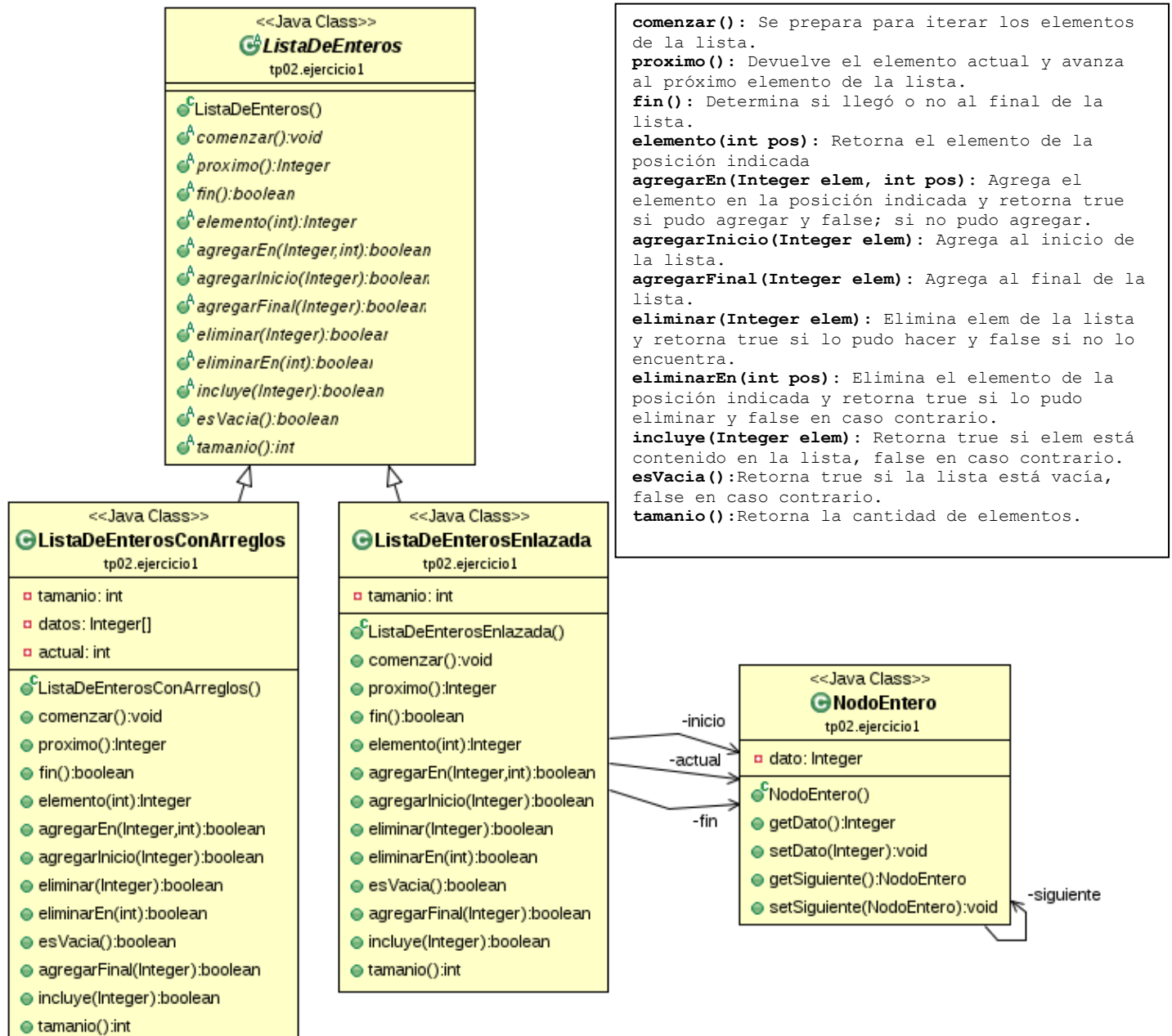
Práctica 2

Abstracción y encapsulamiento

Herencia. Polimorfismo. Tipos Genéricos

Importante: Descargue el material disponible en el sitio de la cátedra. Se recomienda trabajar a partir de esta práctica en un mismo proyecto (**AyED**) y crear un paquete por cada ejercicio.

1. Considere la siguiente especificación de operaciones de una lista de enteros:





- 1.1.** Importe en Eclipse el archivo **ListasDeEnteros.zip** dado por la cátedra usando la opción Import > Existing Projects into Workspace, y luego click en "Select archive file" y seleccione el archivo .zip descargado. Para poder usar las listas de enteros y sus operaciones, en cada una de las declaraciones de clases se debe agregar **import tp02.ejercicio1.*;**
- 1.2.** Escriba una clase llamada **TestListaDeEnterosConArreglos** que reciba en su método **main** una secuencia de números, los agregue a un objeto de tipo **ListaDeEnterosConArreglos** y luego imprima los elementos de dicha lista.
- 1.3.** Escriba una clase llamada **TestListaDeEnterosEnlazada** 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.
- 1.4.** ¿Qué diferencia encuentra entre las implementaciones de los puntos anteriores?
- 1.5.** Escriba un método recursivo que imprima los elementos de una lista en sentido inverso. La lista la recibe por parámetro.
- 1.6.** Si se aplica la siguiente función de forma recursiva a partir de un número n positivo se obtiene un sucesoión que termina en 1:

$$f(n) = \begin{cases} \frac{n}{2}, & \text{si } n \text{ es par} \\ 3n + 1, & \text{si } n \text{ es impar} \end{cases}$$

Por ejemplo para $n = 6$, se obtiene la siguiente sucesoión:

1. $f(6) = 6/2 = 3$
2. $f(3) = 3*3 + 1 = 10$
3. $f(10) = 10/2 = 5$
4.

Es decir la sucesoión 6, 3, 10, 5, 16, 8, 4, 2, 1. Para cualquier n con el que se arranque siempre se llegará al 1.

a) Escriba un programa recursivo que, a partir de un número n , devuelva una lista con cada miembro de la sucesoión.

```
public class Ejercicio1_6 {  
  
    public ListaDeEnterosEnlazada calcularSucesion (int n) {  
  
        //código  
  
    }  
  
}
```

Sugerencia: Primero modele el problema sin tener que devolver una lista.



UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos
Cursada 2019

b) Escriba un método **main** que pruebe el método implementado en a) y recorra la lista resultado e imprima cada uno de los elementos.

```
public class Ejercicio1_6 {  
  
    ...  
  
    public static void main (String[] args) {  
  
        Ejercicio1_6 f = new Ejercicio1_6();  
  
        ListaDeEnterosEnlazada l = f. calcularSucesion(4);  
  
        //código que recorre e imprime los valores de l  
  
    }  
  
}
```

1.7. Analice las implementaciones de la clase **ListaDeEnteros** y sus subclases.

a) ¿Podría darle comportamiento a algún método de la superclase **ListaDeEnteros**? ¿Por qué la clase se define como abstracta? Note que una subclase implementa la lista usando un arreglo de tamaño fijo y la otra usando nodos enlazados.

b) Considerando los enlaces entre nodos, ¿qué diferencias existen entre agregar un nodo al principio de la lista, agregar un nodo en el medio y agregar un nodo al final?

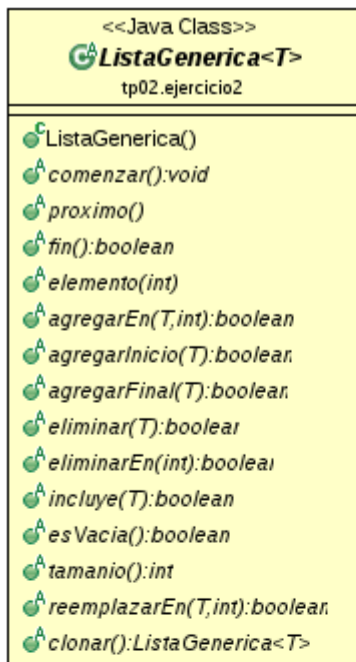
c) Una lista implementada con arreglos, ¿tiene su primer elemento en el índice del vector: 0, 1 o depende de la implementación?

2. Tipos Genéricos

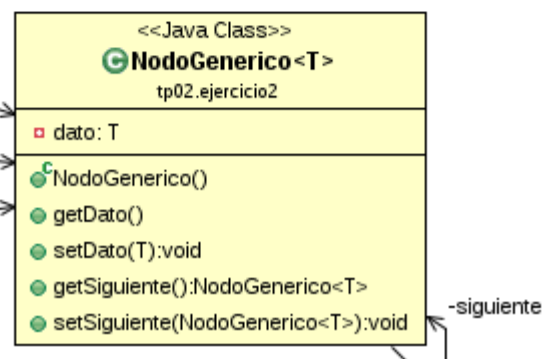
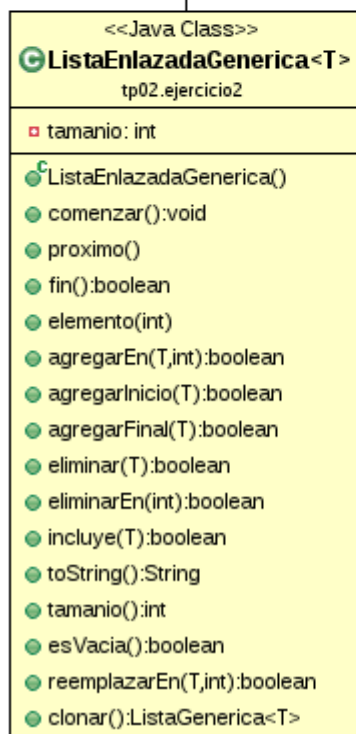
Considere la siguiente especificación de operaciones de listas genéricas:



UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos
Cursada 2019



comenzar(): Se prepara para iterar los elementos de la lista.
proximo(): Devuelve el elemento actual y avanza al próximo elemento de la lista.
fin(): Determina si llegó o no al final de la lista.
elemento(int pos): Retorna el elemento de la posición indicada
agregarEn(T elem, int pos): Agrega el elemento en la posición indicada y retorna true si pudo agregar y false; si no pudo agregar.
agregarInicio(T elem): Agrega al inicio de la lista.
agregarFinal(T elem): Agrega al final de la lista.
eliminar(T elem): Elimina elem de la lista y retorna true si lo pudo hacer y false si no lo encuentra.
eliminarEn(int pos): Elimina el elemento de la posición indicada y retorna true si lo pudo eliminar y false en caso contrario.
incluye(T elem): Retorna true si elem está contenido en la lista, false en caso contrario.
esVacia(): Retorna true si la lista está vacía, false en caso contrario.
tamano(): Retorna la longitud de la lista.
reemplazarEn(int pos, T elem): Reemplazar el valor de la posición indicada y retorna true si lo pudo reemplazar.
clonar(): Crea una copia de la lista genérica y la retorna.

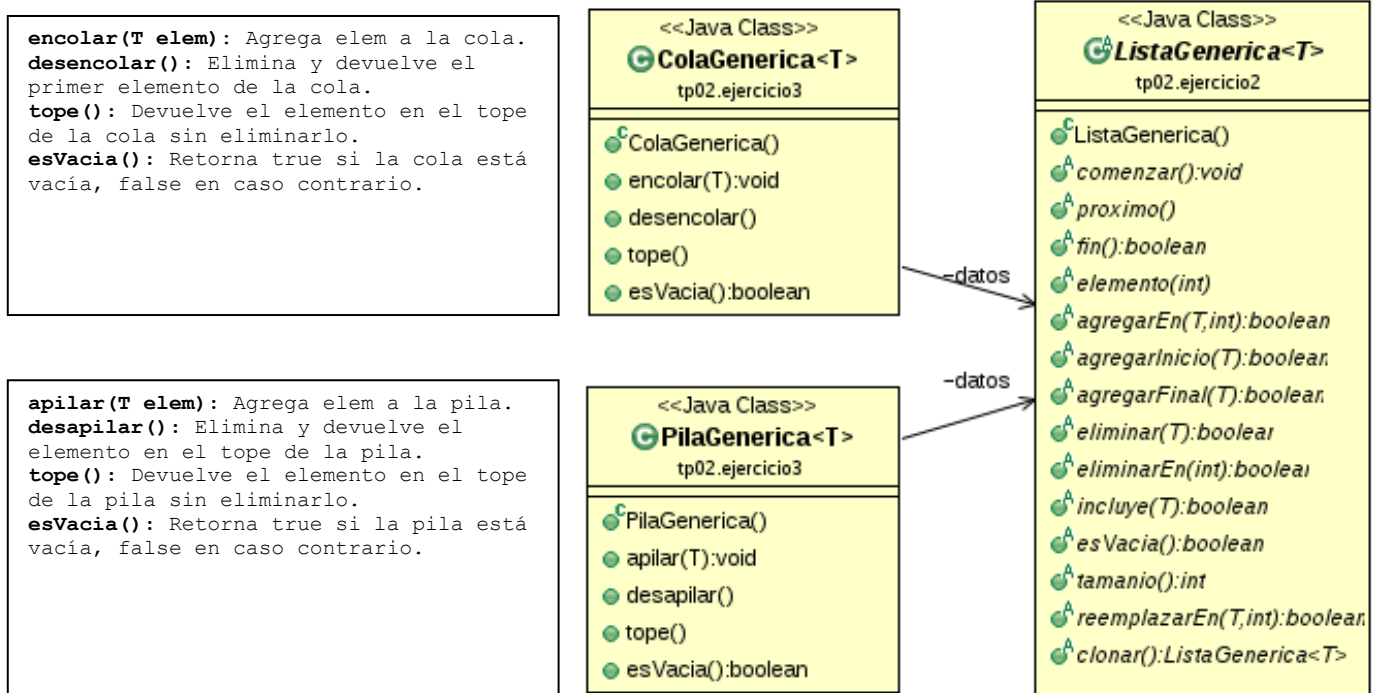




UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos
Cursada 2019

- 2.1. ¿Podría resolver los ejercicios del punto 1 utilizando listas genéricas?
- 2.2. Importe el archivo **ListasGenericas.zip** dado por la cátedra en Eclipse usando la opción Import > Existing Projects into Workspace, y luego click en "Select archive file" y seleccione el archivo .zip descargado. Para poder usar las listas genéricas y sus operaciones, en cada una de las declaraciones de clases se debe agregar **import tp02.ejercicio2.*;**
- 2.3. Escriba una clase llamada **TestListaEnlazadaGenerica** que cree 4 objetos de tipo **Estudiante** (implementado en el TP01B) y los agregue a un objeto de tipo **ListaEnlazadaGenerica** usando los diferentes métodos de la lista y luego, imprima los elementos de dicha lista usando el método **tusDatos()**.
- 2.4. Analice las implementaciones de la clase **ListaGenerica<T>** y sus subclasses, luego responda:
 - a) ¿Qué diferencia observa entre las implementaciones de **ListaEnlazadaGenerica** y **ListaDeEnterosEnlazada**?
 - b) ¿Cómo se define el nodo genérico? ¿Cómo se crea una instancia del mismo?
 - c) ¿Qué devuelve el método **elemento()** de la lista?
 - d) ¿Cómo agregaría un método nuevo? Implemente un nuevo método de la lista que se llame **agregar(T[]):boolean**. El mismo debe agregar todos los elementos del arreglo que recibe como parámetro y retornar true si todos ellos fueron agregados.

3. Sean las siguientes especificaciones de cola y pila genérica:





UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos
Cursada 2019

a) Implemente en JAVA (pase por máquina) las clases **ColaGenerica** y **PilaGenerica** de acuerdo a la especificación dada en el diagrama de clases. Defina estas clases adentro del paquete **tp02.ejercicio3**.

- 4.** Considere un *string* de caracteres S, el cual comprende únicamente los caracteres: (,),[,],{,}. Decimos que S está balanceado si tiene alguna de las siguientes formas:

S = "" S es el *string* de longitud cero.

S = "(T)"

S = "[T]"

S = "{T}"

S = "TU"

Donde ambos T y U son *strings* balanceados. Por ejemplo, "{() [()] }" está balanceado, pero "([)]" no lo está.

a) Indique que estructura de datos utilizará para resolver este problema y como la utilizará.

b) Implemente una clase llamada **tp02.ejercicio4.TestBalanceo** (pase por máquina), cuyo objetivo es determinar si un String dado está balanceado. El String a verificar es un parámetro de entrada (no es un dato predefinido).