Ejercicios tercera actividad del portafolio.

Asignatura: ESTRUCTURAS DE DATOS Y ALGORITMOS 15GIIN

Resolución de ejercicios teórico-prácticos sobre Listas, Pilas y Colas

Calificación: 10 puntos (10% de la nota final)

Instrucciones:

- **Deberá resolver los ejercicios 1 y 2**
- Deberá enviar al profesor un archivo comprimido (.zip, .rar) con la siguiente información:
  - el nombre cumpla con la especificación siguiente: ACT3_NOMBRE_PRIMERAPELLIDO_SEGUNDOAPELLIDO. Por ejemplo: ACT3_OSCAR_MEZA_HOUTTEMAN.zip
  - Los códigos de los programas en JAVA de cada ejercicio en un archivo de texto (o .java), de manera que el profesor pueda luego ejecutarlos. Coloque todas las clases e interfaces en un solo archivo por cada ejercicio. Recuerde la importancia de documentar bien el código en el ejercicio 2.

Fecha de entrega del enunciado por parte del profesor: 29 de Noviembre 2022
Fecha de entrega de la solución por parte del alumno: 20 de Diciembre 2022
Recuerde que de entregar después del 20 de Diciembre, se corregirá la actividad sobre 8 y no sobre 10 puntos

Se incluye el archivo con el enunciado de la actividad y el archivo ListaEnlazada.java

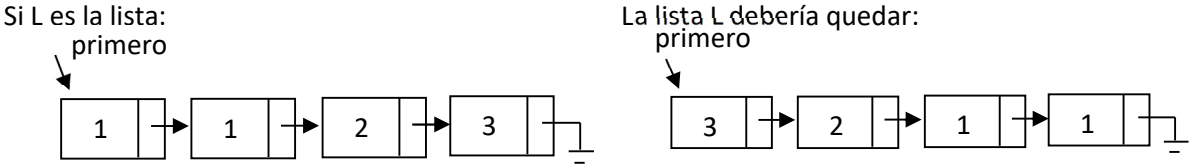
Preguntas:

1. Implementar un método **invertir()** iterativo que invierta una lista (genérica) enlazada de objetos. Deberá implementar el método como uno de la clase **ListaEnlazada<AnyType>**. El método invertirá los elementos de la lista a la que se le pase el mensaje; por ejemplo: lista.invertir(), el objeto lista contendrá después de llamar a invertir() a la lista original pero invertida. **No se permite crear nuevos objetos tipo Nodo en invertir()**. El algoritmo debe ser O(N) donde N es el número de elementos en la lista.

En la clase **ListaEnlazada**, cada nodo sólo hace referencia al siguiente. Al final del ejercicio se presenta la clase **ListaEnlazada** que utilizará (**también incluyo un archivo .java con la clase ListaEnlazada**), puede agregar métodos propios a la clase ListaEnlazada que haga falta, sólo de ser necesario; como agregar, eliminar, etc.. Se tiene referencia solamente al primer elemento de la lista, en la estructura de datos que representa la lista.

Para probar el programa, el método **main** permitirá leer una lista de enteros por pantalla (en una misma línea) y luego imprimir la lista invertida. El método main formará parte de los métodos de la clase **ListaEnlazada<AnyType>**. Deberá enviar al profesor un archivo de texto con el programa **ListaEnlazada.java** que se proporciona con la actividad modificado.

Ejemplo:



Código:

```
class ListaEnlazada<AnyType> {

        // Nodo Inicial
        private Nodo<AnyType>  primero;
        private int tam;  // número de elementos de la lista

        // Constructor de la clase
        public ListaEnlazada () {
                this.primero = null;
                this.tam = 0;
```

```
}

private static class Nodo<AnyType> {
private AnyType valor;
private Nodo<AnyType>  siguiente;
public Nodo ( AnyType valor ) {
        this.valor = valor;
        this.siguiente = null;
}
public void setSiguiente (Nodo<AnyType>  sig) {
        this.siguiente = sig;
}
public void setValor (Nodo<AnyType>  valor) {
        this.valor = valor;
}

Public AnyType getValor () {
        return this.valor;
}
public Nodo<AnyType>  getSiguiente () {
        return this.siguiente;
}
```

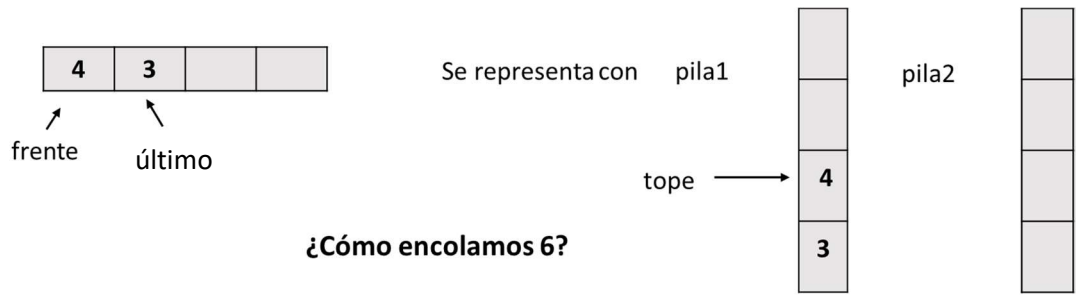
Otros métodos.... entre ellos el método **void invertir()**

```
}
```

(3 puntos)

2. Hacer un programa en JAVA que implemente el tipo **ColaConDosPilas<E>** genérico utilizando en su representación (como estructura de datos) dos PILAS. Por tanto, deberá implementar la interfaz PILA<E> con las operaciones dadas en clase (puede utilizar la implementación dada en clases o utilizar ArrayList<E> para representar la pila y llamar a la implementación PilaArrayList<E>), la cual usará en los métodos de la clase **ColaConDosPilas**. No deberá incluir ninguna estructura de datos adicional para representar una cola, sólo las dos pilas. La clase **ColaConDosPilas** implementará la interfaz Cola genérica vista en clases

Una posible forma de representar una cola con dos pilas (**pila1** y **pila2**) es tener siempre todos los elementos de la cola en la **pila1** y la **pila2** vacía (sólo la usamos para hacer las operaciones de cola), el tope de la **pila1** sería el frente de la cola, y el elemento más al fondo de la **pila1** sería el último elemento de la cola.



Por ejemplo para encolar un elemento **e** en la cola lo que hacemos es desempilar todos los elementos de la **pila1** y empilarlos uno a uno en la **pila2**, luego empilamos el elemento **e** en la **pila1** y finalmente desempilamos los elementos de la **pila2** y los empilamos uno a uno en la **pila1.**

La interfaz Cola sería:

```
public interface Cola<AnyType>
{
   void        encolar( AnyType x );
   void       desencolar( );
   AnyType  frente( );        // devolver el frente de la cola
   boolean   estaVacia( );
   void         convertirVacia( );
   int           numElem( );
   String       toString();  // para poder imprimir la cola comenzando con el frente de la cola
}
```

Y en su programa crearía una Cola de la forma:

```
Cola<String> cola = new ColaConDosPilas<String>();  // puede ser cola de Integer también
// encolar un elemento
Cola.encolar("Hola");
Etc….
```

Para probar su programa, lea por cónsola en una línea los elementos de la cola (pueden ser Integer o String) realice cada una de las operaciones de cola e imprima después de cada operación el valor (estado) de la cola. Cree un método en la clase **ColaConDosPilas**, con el nombre **toString**, que implemente toString() de la interfaz, y devuelva un String con todos los elementos en la cola (el primero en el String es el frente de la cola y el último sería el último de la cola, separados por un espacio en blanco), para poder imprimirla. Dentro de los comentarios de cada método del tipo **ColaConDosPilas** deberá indicar el orden del algoritmo (programa) en función del tamaño de la cola.

(7 puntos)