Estructuras de Datos
Examen Final, Parte Escrita, Versión A
Agosto-diciembre de 2017
Andrés Gómez de Silva Garza

Escoja la opción que representa la mejor respuesta para cada una de las siguientes preguntas. En todas ellas suponga que las interfaces genéricas StackADT<T> y QueueADT<T>, así como las clases genéricas ArrayStack<T> y ArrayQueue<T>, han sido definidas exactamente como lo hicimos en el salón de clases durante este semestre. [2 puntos en total]

1. ¿Qué se imprimiría en la pantalla si se ejecuta la siguiente secuencia de instrucciones?

```
ArrayStack<Integer> a=new ArrayStack<Integer>();
ArrayStack<Integer> b=new ArrayStack<Integer>();
ArrayStack<Integer> c=new ArrayStack<Integer>();
ArrayStack<Integer> d;
ArrayQueue<ArrayStack<Integer>> z=new ArrayQueue<ArrayStack<Integer>>();
a.push(new Integer(7));  a.push(new Integer(6));  a.push(new Integer(5));
b.push(new Integer(-1));  b.push(new Integer(-2));  b.push(new Integer(-3));
z.enqueue(a);  z.enqueue(b);  z.enqueue(c);  d=z.dequeue();  d.pop();
System.out.println(d.pop());
```

a) 7 b) 6 c) 5 d) -1 e) -3

2. ¿Qué se imprimiría en la pantalla si se ejecuta la siguiente secuencia de instrucciones?

```
ArrayQueue<Integer> a=new ArrayQueue<Integer>();
a.enqueue(new Integer(7));  a.enqueue(new Integer(6));
a.enqueue(new Integer(5));  a.enqueue(a.first());  a.enqueue(a.first());
a.dequeue();  a.dequeue();  a.dequeue();  a.dequeue();
System.out.println(a.dequeue());
```

a) 7 b) 6 c) 5 d) -1 e) -3

3. ¿Qué se imprimiría en la pantalla si se ejecuta la siguiente secuencia de instrucciones?

```
ArrayStack<Integer> a=new ArrayStack<Integer>();
ArrayQueue<Integer> z=new ArrayQueue<Integer>();
a.push(new Integer(7));  a.push(new Integer(6));  a.push(new Integer(5));
z.enqueue(new Integer(-1));  z.enqueue(new Integer(-2));
z.enqueue(new Integer(-3));  z.enqueue(a.pop());  z.enqueue(a.pop());
z.enqueue(a.peek());  a.push(z.dequeue());
System.out.println(a.peek());
```

a) 7 b) 6 c) 5 d) -1 e) -3

4. ¿Qué se imprimiría en la pantalla si se ejecuta la siguiente secuencia de instrucciones?

```
ArrayQueue<Integer> a=new ArrayQueue<Integer>();  ArrayQueue<Integer> b=a;
a.enqueue(new Integer(7));  a.enqueue(new Integer(6));
a.enqueue(new Integer(5));  b.enqueue(new Integer(-1));
b.enqueue(new Integer(-2));  b.enqueue(new Integer(-3));
b.dequeue();  b.dequeue();
System.out.println(b.first());
```

a) 7 b) 6 c) 5 d) -1 e) -3

Estructuras de Datos
Examen Final, Parte Programada
Agosto-diciembre de 2017
Andrés Gómez de Silva Garza

Ésta es la parte programada del examen. Lea por completo, y cuidadosamente, todo lo que se especifica y se solicita en esta hoja. Como siempre, al diseñar su solución debe tomar en cuenta los principios del software de calidad: robustez, eficiencia, legibilidad, modularidad, reutilización, generalidad, etc. También es necesario tomar en cuenta los aspectos/temas que se han visto en el curso acerca de la programación orientada a objetos y las estructuras de datos. No se permite utilizar las instrucciones *break* o *continue* en su programa, pues se deben seguir los principios de la programación estructurada.

La entrega de la parte programada debe ser a través de WeTransfer (http://www.wetransfer.com) a más tardar a las 10:45 de hoy. La dirección electrónica de su profesor es: agomez@itam.mx. Se debe entregar un solo proyecto de NetBeans con todo el código completo (es decir, el código solicitado junto con la solución parcial que se le entregó al principio del examen, formando un sistema completo). Se debe comprimir la carpeta raíz del proyecto de NetBeans y el archivo comprimido resultante es el que se debe enviar a través de WeTransfer. Al hacerlo asegúrese de escribir, en el cuadro de "mensaje", su grupo de la materia y su nombre, ya que muchas veces no se puede inferir el mismo simplemente por la dirección electrónica de la persona que hizo el envío.

Durante la resolución del examen no puede consultar apuntes/libros ni acceder a ningún archivo o sitio de Internet, con la excepción de que inicialmente tendrá que descargar de ComunidadITAM un proyecto de NetBeans parcial que se le va a entregar por ese medio (ver abajo) y al final tendrá que entrar a WeTransfer para enviar su solución.

Problema de programación 1 [5 puntos]. La situación es la siguiente:

Se tiene una clase llamada ListaDesordenada<T> cuyos datos se almacenan usando nodos enlazados. Esta clase se le proporcionará a través de ComunidadITAM, dentro de un proyecto de NetBeans parcial que contiene también otras clases que son necesarias para resolver este examen. Lo que se desea es agregarle un método *intercambiaPorPares* a la clase ListaDesordenada<T> que consiga que se reordenen los datos que contiene la lista de tal forma que los datos que antes aparecían en posiciones impares de la secuencia de nodos terminen apareciendo en posiciones pares, y viceversa. Otra forma de entenderlo es que se desea que, dentro de cada par de valores consecutivos de la secuencia, los valores se intercambien de posición. Más específicamente, si un dato originalmente aparecía en una posición *i* impar, debe terminar apareciendo en la posición *i+1* después del intercambio, y si un dato originalmente aparecía en una posición *i* par, debe terminar apareciendo en la posición *i-1* después del intercambio. Si originalmente había menos de dos datos, no debe haber ninguna modificación a la secuencia de nodos enlazados. Si originalmente había una cantidad impar de nodos enlazados, el último dato de la secuencia debe permanecer en la misma posición que tenía originalmente. Para fines de este problema vamos a suponer que el primer dato que aparece en la secuencia de nodos enlazados ocupa la posición 1 (es decir, está en una posición impar). Como ejemplo, si la secuencia de nodos enlazados contenía originalmente a los siguientes datos alfabéticos (en el orden mostrado):

Datos: (inicio) L E F E N A E T (fin)

La secuencia debe terminar siendo la siguiente después del intercambio:

Datos: (inicio) E L E F A N T E (fin)

Otro ejemplo es que si la secuencia original era:

Datos: (inicio) A (fin)

La secuencia debe terminar siendo:

Datos: (inicio) A (fin)

Otro ejemplo es que si la secuencia original era:

Datos: (inicio) A B C (fin)

La secuencia debe terminar siendo:

Datos: (inicio) B A C (fin)

Problema de programación 2 [3 puntos]. La situación es la siguiente:

Se desea agregarle a la misma clase ListaDesordenada<T> del problema anterior un método *eliminaNesimoDesdeElFinal* que, dado un número entero *n*, elimine de la secuencia de nodos enlazados el que está a *n* posiciones del final de la secuencia. Si el valor de *n* es tal que, debido a la cantidad de nodos/datos que tiene la secuencia, no se puede realizar la eliminación especificada, entonces no se debe realizar ninguna modificación a la secuencia. Como ejemplo, si originalmente teníamos la siguiente secuencia de datos alfabéticos:

Datos: (inicio) A B C D (fin)

Si usamos el método con un valor de 0 para *n*, la secuencia que resulte después de la eliminación debería ser:

Datos: (inicio) A B C (fin)

Si por otra parte se hubiera usado el método con un valor de 2 para *n*, la secuencia resultante después de la eliminación debería ser:

Datos: (inicio) A C D (fin)

Si por otra parte se hubiera usado el método con un valor de 3 para *n*, la secuencia resultante después de la eliminación debería ser:

Datos: (inicio) B C D (fin)

Si por otra parte se hubiera usado el método con un valor de 50 para *n*, la secuencia resultante después de la eliminación debería seguir siendo:

Datos: (inicio) A B C D (fin)

Si se decide implementar un método *size*, *length*, o equivalente en la clase ListaDesordenada<T> como método de apoyo para implementar la funcionalidad de *eliminaNesimoDesdeElFinal*, se debe programar este método de apoyo recursivamente.

Revise cuidadosamente qué clases e interfaces existen en el proyecto de NetBeans que se le repartió y qué atributos y métodos tienen éstas. También revise cuidadosamente las pruebas que se hacen en el método *main* de la clase Main, para entender qué situaciones tratan de evaluar dichas pruebas y qué resultados deberían arrojar. Algunas de las pruebas involucran los ejemplos presentados arriba, pero *main* incluye pruebas adicionales. Todas deben poderse ejecutar exitosamente. Las únicas modificaciones que puede realizar en el proyecto de NetBeans que se le entregó son para agregar los métodos solicitados/descritos arriba.