**Enero - mayo del 2020. Examen Final 2.5 hs.**

**Estructuras de Datos**

Nombre \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Clave Única \_\_\_\_\_\_\_\_\_\_\_\_\_

**INSTRUCCIONES PARA EMPEZAR**

Copie el proyecto de Netbeans “*ExFinEM20*”, que viene dentro del material del curso “**ExFinInicio**”.

Este proyecto “*ExFinEM20*” es un esqueleto con las clases “*Proble1*”, y “*Proble2*, donde usted programara los problemas más abajo enunciados.

Además, en el proyecto, también se cuenta con los paquetes de ADTs: *Queues* y *LinkedStructures*.

Inmediatamente cambie el nombre del proyecto “*ExFinEM20*”, cambiando a un nuevo nombre de proyecto que contenga su clave única de alumno seguido de sus tres o cuatro iniciales, p.e, “*199047MAJ2doEx*”

**En los métodos a programar, en los problemas, no se olvide de verificar los parámetros pasados y agregar ejemplos para mostrar su CORRECTA FUNCIONALIDAD. Según corresponda, debe mostrar ejemplos con parámetros adecuados y no adecuados, y contraejemplos; también deberá proteger los métodos de posibles excepciones. Los problemas no deben de presentar ERRORES DE COMPILACIÓN ni ERRORES DE EJECUCIÓN.**

**Dentro de los problemas no se olvide de mandar imprimir su nombre y clave única.**

**Todas estas actividades se califican.**

**No se le permite modificar ninguno de los módulos que se le entregaron, exceptuando la adición de los métodos solicitados (y cualquier otro método en los que éstos se apoyen). Tampoco se le permite agregar otros módulos al proyecto de NetBeans, exceptuando los que se le solicitan explícitamente.** Sus respuestas deben tomar en cuenta los principios de diseño del software de calidad: generalidad, modularidad, reusabilidad, robustez, legibilidad, eficiencia, etc. También deben seguir los principios de la programación estructurada y de la programación orientada a objetos vistos en la materia de Algoritmos y Programas y en esta materia.

Ya hay varias pruebas pre-programadas en el métodos ***main(…)*** de la clases *Proble1* y *Proble2*, que permiten evaluar el funcionamiento correcto de los métodos que se le solicitan en estas problemas.

Al final no se le olvide leer las INSTRUCCIONES PARA ENTREGAR / SUBIR EL EXAMEN.

SIGUIENTE PÁGINA >>

**Problema 1**. **[4.0 puntos en total]**. Clase *Proble1*, paquete de default.

Queremos poder eliminar el segundo elemento de una *cola*, contándolos desde el inicio (*front*) de la *cola*. También se necesita poder imprimir todos los elementos de la *cola*. En cada uno de los dos métodos (incisos **a** y **b**) que se solicitan a continuación se debe apoyar en el uso de una *cola auxiliar* (no un arreglo, ni una pila, ni un conjunto, ni ninguna otra cosa), para implementar la funcionalidad descrita.

**a) [2 puntos]:**

Escriba el método ***eliminaSegundo(…)***, agregándolo a la clase **Proble1**. Dicho método debe ser destructivo en caso de que la *cola* de entrada tenga dos o más elementos (pues en ese caso debe dejar en la *cola* todos los demás elementos, en el mismo orden relativo en el que estaban antes, exceptuando el segundo). Por otra parte, el método debe ser no destructivo en caso de que la *cola* tenga menos de dos elementos (es decir, en estas situaciones debe dejar la *cola* en su estado original, aunque quizá se haya alterado su contenido provisionalmente durante la ejecución del método). El método debe regresar el *elemento eliminado*, o *null* en caso de no haber eliminado ningún elemento. Considere que, si la cola es *null* o *vacia*, igualmente no podrá eliminar ningún elemento, regresando *null*.

La firma del método es:

**public static <T> T eliminaSegundo(CircularArrayQueue<T> cola) {…}**

**b) [2 puntos]:**

Escriba también el método recursivo ***imprimeCola(…)***, agregándolo a la clase **Proble1**, que regrese una cadena de caracteres, que concatene, en una sola línea de texto, y separados por espacios (' '), los *toString()* de todos los elementos almacenados en una *cola* genérica, de tal forma que al imprimir dicha línea de texto aparezca hasta la izquierda el elemento que estaba al inicio de la cola y que aparezca hasta la derecha el elemento que estaba al final de la cola. El método ***imprimeCola(…)*** debe ser no destructivo. Al ser este método recursivo, no se permiten ciclos iterativos: *for*, *while*, *do-while*, etc.

La firma del método es:

**public static <T> String imprimeCola(ArrayQueue<T> cola) {…}**

Algunos ejemplos del *main(…)*:

Clave y nombre del alumno.

Cola 1 inicial: { hola 3 3.1416 -17 }

Primera prueba...se eliminó: 3

Cola 1 resultante: { hola 3.1416 -17 }

Cola 2 inicial: { }

Segunda prueba...se eliminó: null

Cola 2 resultante: { }

Cola 3 inicial: { A Z }

Tercera prueba...se eliminó: Z

Cola 3 resultante: { A }

SIGUIENTE PÁGINA >>

**Problema 2**. **[6.0 puntos en total]** Clase *Proble2*, paquete de default.

Haciendo uso de las clases de los paquetes *Queues* y *LinkedStructures* que se les ha proporcionado, se desea contar con código que permita imprimir, procesar y sumar los elementos de estructuras *LinkedStruct<T>* tal como se detalla en los siguientes incisos.

**a) [1.25 punto]:** Agregue un método ***toString()*** a la clase *LinkedStruct<T>* para que las instrucciones de impresión de estructuras *LinkedStruct* que aparecen en el método *main(…)* de la clase *Proble2*, impriman el contenido completo de los datos almacenados en la estructura enlazada que se les envía como argumento. El método debe ser no destructivo.

**b) [1.25 punto]:** Agregue a la clase *Proble2* un método ***queueSize(…)*** que, dada una *cola* como argumento, regrese un número entero que indique cuántos datos contiene dicha cola. Si la *cola* esta *null* o *vacia*, regrese un cero. El método debe ser no destructivo.

La firma del método debe ser:

**public static <T> int queueSize(CircularArrayQueue<T> cola) {…}**

**c) [1.75 puntos]:** Agregue a la clase *Proble2* un método ***intercambiaColasEnLS(…)*** que, dada una estructura *LinkedStruct* de *colas* (vamos a suponer que la estructura *LinkedStruct* contiene N *colas*, con N par o impar), modifique la estructura *LinkedStruct* tal que termine intercambiando de posición las colas de la siguiente manera: la cola de la posición 1 con la cola de la posición 2, la cola de la posición 3 con la de la posición 4, y así sucesivamente, claro que puede pasar que N sea impar, entonces la cola de la última posición quedara intacta.

Puede suponer que el parámetro de entrada inicialmente almacena en sus nodos únicamente *colas* (y que el método que se solicita aquí no tiene que verificarlo). El método después de ejecutarlo deja la estructura *LinkStruct* con las mismas colas, aunque en distinta posición.

La firma del método debe ser:

**public static void intercambiaColasEnLS(LinkedStruct<Object> ls) {…}**

Por ejemplo:

e1: {LS:3: { }, { 0.5 7.0 0.0 -3.14 }, { -17 3 18 }, }

después: {LS:3: { 0.5 7.0 0.0 -3.14 }, { }, { -17 3 18 }, }

**d) [1.75 puntos]:** Agregue un método ***suma()*** a la clase *LinkedStruct<T>* que regrese la suma de los valores numéricos almacenados en una estructura *LinkedStruct* (únicamente se va a considerar que un valor es numérico si es un objeto de tipo *Integer* o *Double*). Si ninguno de los valores de una estructura *LinkedStruct* dada es numérica, el método debe lanzar una excepción de un tipo no-predefinido, *ADTsException*, la clase que describe dicho tipo de excepción. El método debe ser no destructivo.

La firma del método debe ser:

**public double suma() {…}**

Por ejemplo:

e3: {LS:4: -0.1, 3.1, 7, -3, }

después: e3.suma: 3.0999

ANTERIOR PÁGINA >>

SIGUIENTE PÁGINA >>

**INSTRUCCIONES PARA ENTREGAR EL EXAMEN**

Habiendo salido de Netbeans, comprima su proyecto de trabajo, “*199047MAJF2doEx*”, con ZIP, resultando el archivo “*199047MAJF2doEx.zip*”.

Para subir su examen a Comunidad, entre a la sección de nuestro curso, seleccione “**Trabajos y Exámenes**” y agréguelo al enlace “**ExFinFinal**”.

Como un elemento de respaldo de su examen, no vaya a tocar posteriormente en la computadora, ni el proyecto de trabajo ni el archivo ZIP, para que no se modifique la última fecha de modificación.

<< ANTERIOR PÁGINA

ANTERIOR PÁGINA >>