UTN – FRC
Ingeniería en Sistemas de Información
Tecnología de Software de Base – Electiva Tercer Nivel
Parcial Único [Turno Tarde - COMPLETO]

[Preview] Implemente una clase *TSBDeQueue* de forma de representar la estructura conocida como *Cola de Doble Entrada* (una estructura lineal que permite insertar por el extremo inicial y por el extremo final y también remover por el extremo inicial y por el extremo final, sin permitir ningún otro tipo de inserción o borrado).

La interface nativa *java.util.Deque* enumera la lista de métodos que debería tener una estructura pensada como un soporte de elementos antes de su procesamiento (en principio en orden FIFO pero no necesariamente). La clase nativa java.util.ArrayDeque implementa la interface Deque y es la versión nativa de Java para el concepto de Cola de Doble Entrada. Su trabajo en este parcial: emular tanto como sea posible la clase *java.util.ArrayDeque*, en modalidad *clean room*.

La implementación solicitada aquí consiste en definir una clase *TSBDeQueue* que emule una cola de doble entrada sobre un ***arreglo adaptativo*** y que implemente *Deque* (sin derivar ni usar *java.util.ArrayDeque*), que implemente los siguientes métodos para abstraer una *Cola de Doble Entrada* (inserciones en ambos extremos, eliminaciones por ambos extremos):

1. Todos los constructores (tal como los sugiere *ArrayDeque*).
2. Todos los que permiten insertar al inicio y final del arreglo de soporte.
3. Todos los que permiten eliminar el primer elemento y el último del arreglo de soporte.
4. Todos los que permiten consultar los elementos de los extremos.
5. Un iterador ascedente y otro descendente (*iterator()* y *descendingIterator()*).
6. Los método equals(), hashCode() y clone() (heredados desde Object).

La clase *TSBDeQueue* debe derivar directamente desde *java.util.AbstractCollection* e implementar las interfaces *Serializable*, *Cloneable*, *Iterable* y *Deque*, con uso del mecanismo generics. Podrá observar que la interface *Deque* no requiere demasiados métodos, pero de todos modos tenga cuidado de no saltearse ninguno importante de la clase *java.util.AbstractCollection* que pudiera necesitar redefinir (como *iterator()*...)

La clase *java.util.AbstractCollection* implementa la interface *java.util.Collection* y provee una implementación básica de los métodos de *Collection*, que el estudiante puede dejar tal como están (siempre y cuando provea una implementación completa del iterador para la clase *TSBArrayDeque*). Esos métodos son los siguientes: *isEmpty()*, *contains(Object o)*, *toArray()*, *toArray(T[] a)*, *remove(Object o)*, *containsAll(Collection<?> c)*, *addAll(Collection<? Extends E> c)*, *removeAll(Collection<?> c)*, *retainAll(Collection<?> c)*, *clear()* y *toString()*.

Para resumir: su clase *TSBDeQueue* debe derivar de *AbstractCollection*, implementar *Deque*, *Serialization* y *Cloneable*, y contener una implementación específica de todos los métodos de la interface *Deque* que **NO** hayan sido a su vez implementados por *AbstractCollection*. La clase que desarrolle será testeada con una JUnit especialmente diseñada por la Cátedra a ese efecto. Esa clase JUnit básica viene incluida dentro del proyecto modelo que acompaña a este enunciado. El proyecto modelo también contiene un esquema de la clase *TSBDeQueue*, sin atributos y con las implementaciones "en blanco" de todos los métodos que se le piden implementar. Trabaje con esa clase, agregue todos los atributos que necesite, cambie todas las implementaciones de todos los métodos para que se ajusten a sus especificaciones según se indica en el javadoc de la clase *ArrayDeque*, y finalmente asegúrese que su clase pase el test especificado por la JUnit que se provee en el mismo modelo.

Cuando haya finalizado, preséntese a completar su parcial en el día 17/10 en el horario especificado para su curso. Ese día se le presentarán consignas adicionales que deberá completar presencialmente. Podrá trabajar con el proyecto que haya desarrollado para este preview, así que asegúrese de llevarlo o tenerlo disponible cuando concurra al horario del parcial.

[Consignas adicionales – Turno Mañana]

1.] Agregue en su clase *TSBDeQueue* un método:

public boolean checkDuplicates(E e)

tal que determine cuántas veces está en la cola el objeto especificado por *e*. Lance una excepción de *NullPointerException* si *e* es *null*.

2.] Diseñe una clase iteradora interna **OddIndexIterator** para su clase *TSBDeQueue*, tal que permita recorrer y retornar *sólo los objetos ubicados en posiciones impares de la cola* (comenzando en la posición 1 e incluida ella).

3.] Agregue en su clase *TSBDeQueue* un método:

public Iterator oddIndexIterator()

tal que cree y retorne un iterador de la clase *oddIndexIterator* pedida en el punto anterior.

4.] Programe una clase *Main* que contenga un método *main()* para hacer pruebas elementales con los métodos que acaba de crear. En ese *main()*, debe crear una instancia de su clase *TSBDeQueue*, luego generar *n* valores *int* en forma aleatoria y seleccionar para cada uno (también en forma aleatoria) en qué extremo debe insertar cada valor. Cuando la cola esté creada y completa con esos *n* valores, cree un iterador con el método *oddIndexIterator()*, recorra esa cola mediante ese iterador, y elimine usando el método *remove()* del iterador el *tercer elemento que el iterador le retorne*. Muestre la conversión a String de la cola cuando finalice.