Primer examen integrador - Programación II
1C 2025

Profesor Monzón, Nicolás Alberto

18 de abril de 2025

Requisitos para aprobar el integrador

A. Por actos de deshonestidad académica será sancionado.

B. Para poder aprobar este examen ud. deberá estar administrativamente en condiciones de poder rendir el examen. En caso de no estarlo y rendir el examen, este no va a ser corregido y quedará anulado.

C. Deberá comprimir la carpeta `src` de su proyecto junto a los `txt` utilizados (también se admiten `pdf` y `MD`). El archivo comprimido deberá tener como nombre el número de grupo. Deberá enviarlo por mail a `nimonzon@uade.edu.ar`.

D. Deberá desarrollar un set de prueba para demostrar que funciona su código (no se piden test unitarios, pero si alguna prueba en el `main` del proyecto).

E. Solo podrá utilizar técnicas vistas en este curso. Está prohibido el uso de librerías, estructuras que vienen por defecto en la JRE y el uso de genéricos. No respetar este punto es suficiente para desaprobar el integrador.

F. Deberá ser entregado dentro de las 3 horas y 30 minutos. a partir del horario de inicio.

G. CONDICIONES PARA APROBAR: el examen se mide con una escala logarítmica donde obtener al menos 60 puntos corresponderá a un 4 y se deberá cumplir con todos los puntos anteriores. En caso de no cumplir con alguno de los puntos anteriores, el examen queda desaprobado sin excepción.

1. Ejercicio 1 (50 %)

Crear el TDA `CircularBuffer`, imitando (en lo que se pueda) la implementación de `Queue` desarrollada en clase. Será una estructura lineal destructiva.

La característica principal de la estructura es que cuando un elemento se *consume* pasa a estar al final de la estructura.

Se espera:

- Definir un conjunto de métodos que permita resolver la mayor cantidad de algoritmos.
- La estructura será mutable.
- Agregar a la definición de los métodos cuáles son las precondiciones, postcondiciones y estrategia de implementación.
- Indicar además en un archivo `txt` cuáles son los invariantes que consideran para la estructura.

Tengan en cuenta todo lo que se les ocurra. Por ejemplo, si consumir un elemento no cambia el tamaño de la estructura, para recorrerla desde fuera tiene que ser posible con el resto de los métodos definidos.

2. Ejercicio 2 (10 %)

Modificar la implementación estática de `Stack`, para que en lugar de usar un arreglo nativo, se utilice el TDA `CircularBuffer`.

3. Ejercicio 3 (10 %)

Desarrolle un método `map` que permita convertir una instancia de `CircularBuffer` a una instancia de `Queue`, y otro método `map` que permita convertir una instancia de `Queue` a una `CircularBuffer`.

4. Ejercicio 4 (10 %)

Desarrolle una función que permita invertir el contenido de la estructura.

5. Ejercicio 5 (10 %)

Crear el TDA `Pair` que cumpla las propiedades usuales de una Tupla de 2 componentes. La diferencia con una dupla es que el par es mutable. Dar una implementación, invariantes, precondiciones, postcondiciones y definir una clase utilitaria con una función que proyecte la i-ésima componente (en este caso $i = 0$ o $i = 1$).

6. Ejercicio 6 (10 %)

Modificar la estructura anterior para que utilice genéricos. *Buscar Java generics.*