Programación II

Trabajo Práctico Obligatorio - 2025

Aclaraciones:

- El Trabajo Práctico Obligatorio (TPO) será desarrollado en forma grupal.
- Para aprobar el presente TPO debe ser aprobada la defensa individual del mismo. En esta se solicitará la explicación de alguno/s de los ejercicios planteados en el mismo.
- El uso de colecciones de Java no está permitido. Las únicas estructuras que pueden utilizarse son arreglos, listas enlazadas definidas en clase, objetos definidos en el mismo ejercicio y cualquiera de los ocho TDA vistos.
- Entrega: 19 de Mayo
- Defensa: se publicará en el archivo compartido de los grupos
- Se recomienda leer con detalle el enunciado antes de comenzar, y definir la estrategia antes de codificar.

Se solicita para cada ejercicio:

a) Describir la estrategia utilizada para el/los método/s.
b) Escribir la interfaz de cada clase.
c) Escribir el código en lenguaje Java de el/los método/s.
d) Estimar la complejidad, salvo en el caso de haber utilizado recursividad.

1. Se define un nuevo TDA denominado ConjuntoEspecialTDA basado en ConjuntoTDA, con la particularidad de que se permite más de una acepción de cada elemento agregado. Tal cual como en ConjuntoTDA, no existe orden alguno. Codificar la clase que implementa ConjuntoEspecialTDA, con representación dinamica.

2. Se define un nuevo TDA denominado DiccionarioSimpleModTDA basado en DiccionarioSimpleTDA, con la particularidad de registrar la cantidad de veces que el valor se ve modificado. Codificar la clase que implementa DiccionarioSimpleModTDA, con representación dinámica. La interfaz del mismo es la siguiente:

```
public interface DiccionarioSimpleModTDA {
    public void inicializarDiccionario(); //inicializa el TDA
    public void agregar(int clave, int valor); //agrega el par clave-valor al TDA,
      //conjuntamente con la cantidad de veces que dicho valor se vio
      //modificado. Si la clave se agrega por primera vez, el factor de
      //modificaciones será 0. Si la clave se agrega por segunda vez,
      //modificándose el valor, el factor de
      //modificaciones será 1. Y así sucesivamente.
    public void eliminar(int clave); //elimina la clave del TDA, con su valor y
    //factor de modificaciones
    public int recuperar(int clave); //devuelve el valor asociado a la clave, que
    //se supone existente
    public int recuperarMod(int clave); //devuelve la cantidad de
    //modificaciones que sufrió el valor relacionado a dicha clave, que se
    //supone existente
    public ConjuntoTDA claves(); //devuelve el conjunto de claves
    }
```

3. Se define un método que reciba una PilaTDA y devuelva un int (número entero) con la cantidad de elementos pares de la pila. Codificar la clase que implementa PilaTDA, con representación dinámica

4. Se define un método que reciba una PilaTDA y devuelva un ConjuntoTDA con los elementos repetidos de la pila.

5. Se define un método que reciba una PilaTDA y devuelva un DiccionarioSimpleTDA, en el cual se guardarán los elementos de la pila como claves, y la cantidad de apariciones de dicho elemento en la pila, como valores.

6. Se define un método que reciba un DiccionarioMultipleTDA (implementación con estructura de datos dinámica) y devuelva una ColaTDA con todos los valores del diccionario, sin ninguna repetición.

Para los ejercicios anteriores, se requiere hacer pruebas con las siguientes entradas.

1) Entrada para el conjunto: {18,2,7,4,18,22,2,7}
2) Entrada para el DiccionarioSimpleModTDA: (7,3), (16,4), (3,4), (16,2), (3,6), (5,3). Explicar como ingresa el factor de modificación.
3) Entrada para la Pila: <3,4,6,2,45,7,8>
4) Entrada para la Pila: <3,3,6,2,6,7,8,4>
5) Entrada para la Pila: <4,3,6,6,6,7,8,4>
6) Entrada DiccMultiple: <(4, <4,5,6>), (6, <7,5,8,9>), (9, <4,8,7,5)>, (8,<7,4,5,9,0,4>)>.