UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

ALUMNO:

PAZ MALDONADO CARLOS SAÚL

NOMBRE DEL PROFESOR:

HERNANDEZ CABRERA JESUS

NOMBRE DE LA MATERIA:

ESTRUCTURA DE DATOS

FECHA DE ENTREGA:

19 de septiembre del 2024

TAREA NO. 7

1. INSTRUCCIONES DE TAREA

Debe emplear la clase ColaADT.java desarrollada en clase o equivalente en el lenguaje seleccionado.

Debe diseñar e implementar la clase Paciente.

Debe contar con una clase con un main() en donde:

1. Se agreguen 3 pacientes a la cola.
2. Mostrar el contenido de la cola.
3. Mostrar el paciente que sigue (sin sacarlo de la cola)
4. Atender al siguiente.
5. Mostrar el contenido de la cola.
6. Meter 2 nuevos pacientes.
7. Atender al siguiente.
8. Mostrar el contenido de la cola

2. NOTAS:

Para la realización de la clase ColaADT se implemento otra estructura de datos hecha anteriormente, ListaDoblementeLigada, por lo que también presento el código.

3. CAPTURA DE EJECUCIÓN

```
"C:\Program Files\Java\jdk-22.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2\lib\idea_rt.jar=50622:C:\Program Files\Je


Agregando 3 pacientes a la cola
Mostrando contenido de la cola
[Alexis Martinez Prado 19] <--> [Alejandra Heredia Nava 24] <--> [Luis Angel Cervantes Moreno 29] <-->  Tamaño: 3

El paciente que sigue es (Sin sacar de la cola):
Alexis Martinez Prado 19
Comprobando que no se saco de la cola
[Alexis Martinez Prado 19] <--> [Alejandra Heredia Nava 24] <--> [Luis Angel Cervantes Moreno 29] <-->  Tamaño: 3

Atendiendo al siguiente: Alexis Martinez Prado 19

Mostrando el contenido de la cola nuevamente:
[Alejandra Heredia Nava 24] <--> [Luis Angel Cervantes Moreno 29] <-->  Tamaño: 2

Agregando dos pacientes nuevos a la cola:
[Alejandra Heredia Nava 24] <--> [Luis Angel Cervantes Moreno 29] <--> [Leslie Lopez Hernandez 14] <--> [Ana Nieves Clieto 20] <-->  Tamaño: 4

Process finished with exit code 0
```

4. CÓDIGO

a. Clase MAIN

```
package unam.fesaragon.estructuradatos;

import unam.fesaragon.estructuradatos.ADT.ColaADT;
import unam.fesaragon.estructuradatos.clinica.Paciente;

public class Main {
    public static void main(String[] args) {
        ColaADT<Paciente> colaClinica = new ColaADT<>();
        imp("\n\nAgregando 3 pacientes a la cola");
        colaClinica.encolar(new Paciente("Alexis", "Martinez Prado",19,"Hombre"));
        colaClinica.encolar(new Paciente("Alejandra", "Heredia Nava",24,"Mujer"));
        colaClinica.encolar(new Paciente("Luis Angel", "Cervantes
Moreno",29,"Hombre"));
        imp("Mostrando contenido de la cola");
        imp(colaClinica.toString());
        imp("\nEl paciente que sigue es (Sin sacar de la cola): ");
        imp(colaClinica.siguiente().toString());
        imp("Comprobando que no se saco de la cola");
        imp(colaClinica.toString());
        imp("\nAtendiendo al siguiente: " + colaClinica.desEncolar());
        imp("\nMostrando el contenido de la cola nuevamente: ");
        imp(colaClinica.toString());
        imp("\nAgregando dos pacientes nuevos a la cola: ");
        colaClinica.encolar(new Paciente("Leslie", "Lopez Hernandez", 14, "Mujer"));
        colaClinica.encolar(new Paciente("Ana", "Nieves Clieto", 20, "Mujer"));
        imp(colaClinica.toString());

    }
    private static void imp(String str){
        System.out.println(str);
    }
}
```

b. Clase PACIENTE

```
package unam.fesaragon.estructuradatos.clinica;

public class Paciente {
    //Datos de paciente
    private int edad;
    private String nombre;
    private String apellido;
    private String sexo;
    private int peso;
    private int talla;
    //Datos de consulta
    private int noDeConsulta;
    private String fechaDeConsulta;
    private String tratamiento;

    public Paciente(String nombre, String apellido, int edad, String sexo) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
        this.sexo = sexo;
    }

    public void setPeso(int peso) {
        this.peso = peso;
    }

    public void setTalla(int talla) {
        this.talla = talla;
    }

    public void setNoDeConsulta(int noDeConsulta) {
        this.noDeConsulta = noDeConsulta;
    }

    public void setFechaDeConsulta(String fechaDeConsulta) {
        this.fechaDeConsulta = fechaDeConsulta;
    }
```

```
    public void setTratamiento(String tratamiento) {
        this.tratamiento = tratamiento;
    }

    public int getEdad() {
        return edad;
    }

    public String getNombre() {
        return nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public String getSexo() {
        return sexo;
    }

    public int getPeso() {
        return peso;
    }

    public int getTalla() {
        return talla;
    }

    public int getNoDeConsulta() {
        return noDeConsulta;
    }

    public String getFechaDeConsulta() {
        return fechaDeConsulta;
    }

    public String getTratamiento() {
        return tratamiento;
    }

    @Override
    public String toString() {
        return this.getNombre() + " " + this.getApellido() + " " + this.getEdad();
    }
}
```

5. **Clase ColaADT**

```
package unam.fesaragon.estructuradatos.ADT;

public class ColaADT<T> {
    private ListaDoblementeLigada<T> data;

    public ColaADT() {
        this.data = new ListaDoblementeLigada<>();
    }

    public boolean estaVacia() {
        return this.data.esta_vacia();
    }

    public int longitud() {
        return this.data.get_tamanio();
    }

    public T frente() {
        return this.data.obtener(1);
    }

    public void encolar(T valor) { //enqueue
        this.data.agregar_al_final(valor);
    }

    public T desEncolar() {
        //Obtener el primer elemento
        T dato = this.data.obtener(1);
        //Despues eliminarlo
```

```
        this.data.eliminar_el_primero();
        return dato;
    }

    public T siguiente() {
        return this.data.obtener(1);
    }

    @Override
    public String toString() {
        return this.data.toString();
    }
}
```

6. Clase ListaDoblementeLigada

```
package unam.fesaragon.estructuradatos.ADT;

public class ListaDoblementeLigada<T> {
    private NodoDoble<T> head;
    private NodoDoble<T> tail;
    private int tamanio;

    // Constructor
    public ListaDoblementeLigada() {
        this.head = null;
        this.tail = null;
        this.tamanio = 0;
    }

    public ListaDoblementeLigada(NodoDoble<T> head, NodoDoble<T> tail) {
        this.head = head;
        this.tail = tail;
    }

    // Comprobar si está vacía
    public boolean esta_vacia() {
        return this.tamanio == 0;
    }

    // Agregar al inicio de la lista
    public void agregar_al_inicio(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (esta_vacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            nuevo.setSiguiente(this.head);
            this.head.setAnterior(nuevo);
            this.head = nuevo;
        }
        tamanio++;
    }

    // Agregar al final de la lista
    public void agregar_al_final(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (esta_vacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            this.tail.setSiguiente(nuevo);
            nuevo.setAnterior(this.tail);
            this.tail = nuevo;
        }
        tamanio++;
    }

    // Agregar después de un nodo de referencia
    public void agregar_después_de(T referencia, T valor) {
        if (esta_vacia()) {
            System.out.println("La lista esta vacia y por lo tanto no existe la
referencia");
            return;
        }
```

```
        NodoDoble<T> aux = this.head;
        while (aux != null) {
            if (aux.getDato().equals(referencia)) {
                break;
            }
            aux = aux.getSiguiente();
        }

        if (aux == null) {
            System.out.println("El nodo de referencia no existe");
            return;
        }

        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        nuevo.setSiguiente(aux.getSiguiente());
        nuevo.setAnterior(aux);
        //Nodo de referencia, no es la ultima en la lista.
        if (aux.getSiguiente() != null) {
            aux.getSiguiente().setAnterior(nuevo);
        } else {
            this.tail = nuevo;
        }
        aux.setSiguiente(nuevo);
        this.tamanio++;
    }

    // Obtener el elemento en una posición específica
    public T obtener(int posicion) {
        //Condiciones
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return null;
        }
        if (posicion > this.tamanio) {
            System.out.println("Posición fuera de rango");
            return null;
        }

        NodoDoble<T> aux = this.head;
        for (int i = 1; i < posicion; i++) {
            aux = aux.getSiguiente();
        }

        return aux.getDato();
    }

    // Eliminar el primer elemento
    public void eliminar_el_primero() {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return;
        }
        if (this.head == this.tail) { // Solo un elemento
            this.head = null;
            this.tail = null;
        } else {
            this.head = this.head.getSiguiente();
            this.head.setAnterior(null);
        }
        tamanio--;
    }

    // Eliminar el último elemento
    public void eliminar_el_final() {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return;
        }

        if (this.head == this.tail) { // Solo un elemento
            this.head = null;
            this.tail = null;
        } else {
            this.tail = this.tail.getAnterior();
            this.tail.setSiguiente(null);
        }
        tamanio--;
    }

    // Eliminar un elemento en una posición específica
```

```java
    public void eliminar(int posicion) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }
        if (posicion > this.tamanio) {
            System.out.println("Posición fuera de rango");
            return;
        }

        if (posicion == 1) {
            eliminar_el_primero();
            return;
        }
        if (posicion == this.tamanio) {
            eliminar_el_final();
            return;
        }

        NodoDoble<T> aux = this.head;
        int i = 1;
        while (i < posicion) {
            aux = aux.getSiguiente();
            i++;
        }

        aux.getAnterior().setSiguiente(aux.getSiguiente());
        aux.getSiguiente().setAnterior(aux.getAnterior());
        this.tamanio--;
    }

    // Buscar un elemento y retornar su posición
    public int buscar(T valor) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return 0;
        }

        NodoDoble<T> aux = this.head;
        int posicion = 1;
        while (aux != null) {
            if (aux.getDato().equals(valor)) {
                return posicion;
            }
            aux = aux.getSiguiente();
            posicion++;
        }
        System.out.println("No se encontro el elemento");
        return 0;
    }

    // Actualizar un elemento
    public void actualizar(T a_buscar, T valor) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }

        NodoDoble<T> aux = this.head;
        while (aux != null) {
            if (aux.getDato().equals(a_buscar)) {
                aux.setDato(valor);
                return;
            }
            aux = aux.getSiguiente();
        }
        System.out.println("El elemento valor o elemento no esta en la lista");
    }

    // Recorrido transversal en una dirección específica
    // Verdadero es valor por defecto y con el falso es de derecha a izquierda
    public void transversal(boolean porDefecto) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }

        if (porDefecto) {
            NodoDoble<T> aux = this.head;
            while (aux != null) {
```

```
                System.out.print("[" + aux.getDato() + "] <--> ");
                aux = aux.getSiguiente();
            }
        } else {
            NodoDoble<T> aux = this.tail;
            while (aux != null) {
                System.out.print("[" + aux.getDato() + "] <--> ");
                aux = aux.getAnterior();
            }
        }
        System.out.println("NULL");
    }
    //Sobrecargue el metodo para poner por default el recorrido de izquierda a derecha
    public void transversal() {
        transversal(true);
    }

    //GETTERS Y SETTERS
    // Obtener el tamaño de la lista
    public int get_tamanio() {
        return this.tamanio;
    }

    //ToString
    @Override
    public String toString() {
        StringBuilder estado = new StringBuilder();
        NodoDoble<T> aux = this.head;
        while (aux != null) {
            estado.append("[").append(aux.getDato()).append("] <--> ");
            aux = aux.getSiguiente();
        }
        return estado +" Tamaño: "+ this.tamanio;
    }
}
```

7. Clase NODODOBLE

```
package unam.fesaragon.estructuradatos.ADT;

public class NodoDoble<T> {
    private T dato;
    private NodoDoble<T> anterior;
    private NodoDoble<T> siguiente;

    // Constructores
    public NodoDoble() {
    }

    public NodoDoble(T dato) {
        this.dato = dato;
        this.anterior = null;
        this.siguiente = null;
    }

    public NodoDoble(T dato, NodoDoble<T> anterior, NodoDoble<T> siguiente) {
        this.dato = dato;
        this.anterior = anterior;
        this.siguiente = siguiente;
    }

    // Getters y Setters
    public T getDato() {
        return dato;
    }

    public void setDato(T dato) {
        this.dato = dato;
    }

    public NodoDoble<T> getAnterior() {
        return anterior;
    }
```

```java
    public void setAnterior(NodoDoble<T> anterior) {
        this.anterior = anterior;
    }

    public NodoDoble<T> getSiguiente() {
        return siguiente;
    }

    public void setSiguiente(NodoDoble<T> siguiente) {
        this.siguiente = siguiente;
    }
}
```