UNIVERSIDAD AUTÓNOMA GABRIEL RENE MORENO
FACULTAD DE INGENIERÍA Y CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS 2

CONTENIDO: Tarea de clase Listas Encadenadas

PORCENTAJE TERMINADO : 100%.

GRUPO: 15

Integrantes	DT	HG	HI	EVAL
Ibarra Cuellar Gustavo	1	1	1	100

Fecha de Presentación: Jueves 19 de Septiembre 2024
Fecha Presentada: Jueves 19 de Septiembre 2024

CLASES

```
public boolean vacio(){
 return cantElem == 0;
}
 public String ToString() {
   String s1 = "[";
   Nodo p = prim;
   while (p != null) {
      s1 = s1 + p.elem;
```

```
            if (p.prox != null) {
                s1 = s1 + " <-> ";
            }
            p = p.prox;
        }
        return s1 + "]";
    }
```

L1.eliminarPrim()

```
    public void eliminarPrim() {
    if (!vacio()) {
        prim = prim.prox;
        cantElem--;
        if (prim == null) { // Si después de
eliminar el primer nodo la lista está vacía
            ult = null;
        }
    }
}
```

L1.eliminarUlt()

```
    public void eliminarUlt() {
    if (!vacio()) {
        if (prim == ult) { // Solo hay un
elemento
            prim = ult = null;
        } else {
            Nodo p = prim;
            while (p.prox != ult) {
                p = p.prox;
```

```
            }
            p.prox = null;
            ult = p;
        }
        cantElem--;
    }
}
```

L1.eliminarIesimo()

```
    public void eliminarIesimo(int i) {
    if (i < 0 || i >= cantElem) return; //
índice inválido
    if (i == 0) {
        eliminarPrim();
    } else {
        Nodo p = prim;
        for (int j = 0; j < i - 1; j++) {
            p = p.prox;
        }
        p.prox = p.prox.prox;
        if (p.prox == null) ult = p;
        cantElem--;
    }
}
```

L1.eliminarTodo(x)

```
    public void eliminarTodo(int x) {
    while (prim != null && prim.elem == x) {
// Elimina los primeros si son iguales a x
```

```
        eliminarPrim();
    }
    Nodo p = prim;
    while (p != null && p.prox != null) {
        if (p.prox.elem == x) {
            p.prox = p.prox.prox;
            cantElem--;
            if (p.prox == null) ult = p;
        } else {
            p = p.prox;
        }
    }
}
```

L1.eliminarPrim(n)

```
    public void eliminarPrim(int n) {
    for (int i = 0; i < n && prim != null; i++) {
        eliminarPrim();
    }
}
```

L1.eliminarEntre(a,b)

```
public void eliminarEntre(int a, int b) {
    while (prim != null && prim.elem >= a
&& prim.elem <= b) {
        eliminarPrim();
```

```
    }
    Nodo p = prim;
    while (p != null && p.prox != null) {
        if (p.prox.elem >= a && p.prox.elem
<= b) {
            p.prox = p.prox.prox;
            cantElem--;
            if (p.prox == null) ult = p;
        } else {
            p = p.prox;
        }
    }
}
```

L1.eliminarDuplicados()

```
public void eliminarDuplicados() {
    Nodo p = prim;
    while (p != null) {
        Nodo q = p;
        while (q.prox != null) {
            if (q.prox.elem == p.elem) {
                q.prox = q.prox.prox;
                cantElem--;
                if (q.prox == null) ult = q;
            } else {
                q = q.prox;
            }
        }
        p = p.prox;
```

```
    }
}
```

L1.eliminar(L2)

```
public void eliminar(listaEncadenada L2) {
    Nodo p = L2.prim;
    while (p != null) {
        eliminarTodo(p.elem);
        p = p.prox;
    }
}
```

L1.eliminarDesde(i,j)

```
public void eliminarDesde(int i, int j) {
    for (int k = i; k <= j; k++) {
        eliminarIesimo(i); // Siempre
eliminar el i-ésimo, porque se van
corriendo
    }
}
```

L1.eliminarUnicos()

```
public void eliminarUnicos() {
    Nodo p = prim;
    while (p != null) {
        if (frecuencia(p.elem) == 1) {
            eliminarTodo(p.elem);
        }
```

```
            p = p.prox;
        }
    }
```