2025-1

Tarea 4

LISTA LIGADA

Estructura de Datos
Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

REALIZADO POR:

Enrique Emiliano Cano García

22/08/2024

Clase Nodo

```python
class Nodo:  4 usages  kno4
    def __init__(self, dato = None, siguiente = None):  kno4
        self.dato = dato
        self.siguiente = siguiente

    def set_dato(self, dato):  1 usage (1 dynamic)  kno4
        self.dato = dato

    def get_dato(self):  3 usages (3 dynamic)  kno4
        return self.dato

    def set_siguiente(self, siguiente):  5 usages (3 dynamic)  kno4
        self.siguiente = siguiente

    def get_siguiente(self):  18 usages (16 dynamic)  kno4
        return self.siguiente

    def __str__(self):  kno4
        return f"Node(value={self.dato}, siguiente={self.siguiente})"
```

Clase Lista Ligada

```python
from Tareas.Tarea4.Nodo import Nodo

class ListaLigada:  2 usages  kno4
    def __init__(self):  kno4
        self._cabeza = None
        self._tamanio = 0

    def tamanio(self):  kno4
        return self._tamanio

    def agregar_al_final(self, dato):  6 usages  kno4
        nuevo_nodo = Nodo(dato)
        if self.esta_vacia():
            self._cabeza = nuevo_nodo
        else:
            aux = self._cabeza
            while aux.get_siguiente() is not None:
                aux = aux.get_siguiente()
            aux.set_siguiente(nuevo_nodo)
        self._tamanio += 1

    def esta_vacia(self):  3 usages  kno4
        return self._cabeza is None
```

```python
    def agregar_al_inicio(self, dato):  1 usage  kno4
        nuevo_nodo = Nodo(dato,self._cabeza)
        self._cabeza = nuevo_nodo
        self._tamanio += 1

    def get_tamanio(self):  kno4
        aux = self._cabeza
        contador = 0
        while aux.get_siguiente() is not None:
            contador += 1
            aux = aux.get_siguiente()
        return contador

    def agregar_despues_de(self, ref, nuevo_dato):  kno4
        # Buscar el nodo con el dato `nodo_anterior_dato`
        aux = self._cabeza
        while aux is not None:
            if aux.get_dato() == ref:
                break
            aux = aux.get_siguiente()

        if aux is None:
            print(f"Nodo con dato '{ref}' no encontrado.")
        else:
            # Crear el nuevo nodo y ajustar los punteros
            nuevo_nodo = Nodo(nuevo_dato)
            nuevo_nodo.set_siguiente(aux.get_siguiente())
            aux.set_siguiente(nuevo_nodo)
            self._tamanio += 1

    def buscar(self, dato):  kno4
        aux = self._cabeza
        posicion = 0
        while aux is not None:
            if aux.get_dato() == dato:
                return posicion
            aux = aux.get_siguiente()
            posicion += 1
        # Devuelve -1 si el dato no se encuentra en la lista
        return -1

    def transversal(self):  5 usages  kno4
        aux = self._cabeza
        while aux is not None:
            print(aux.get_dato(), "-->", end=" ")
```

```
        while aux is not None:
            print(aux.get_dato(), "-->", end=" ")
            aux = aux.get_siguiente()
        print("\n")

    def eliminar(self, posicion):  2 usages  kno4
        if posicion < 0 or posicion >= self._tamanio:
            print("Posición no válida")
            return

        if posicion == 0:
            self._cabeza = self._cabeza.get_siguiente()
        else:
            aux = self._cabeza
            for _ in range(posicion - 1):
                aux = aux.get_siguiente()
            aux.set_siguiente(aux.get_siguiente().get_siguiente())
        self._tamanio -= 1

    def actualizar(self, posicion, nuevo_dato):  1 usage  kno4
        if posicion < 0 or posicion >= self._tamanio:
            print("Posición no válida")
            return False
        aux = self._cabeza
        for _ in range(posicion):
            aux = aux.get_siguiente()
        aux.set_dato(nuevo_dato)
        return True

    def eliminar_primero(self, dato):  kno4
        if self.esta_vacia():
            print("La lista está vacía, no se puede eliminar el primer nodo.")
            return

        self._cabeza = self._cabeza.get_siguiente()
        self._tamanio -= 1

    def eliminar_ultimo(self):  kno4
        if self.esta_vacia():
            print("La lista está vacía, no se puede eliminar el último nodo.")
            return

        #cuando solo haya un nodo y sea la cabeza
        if self._cabeza.get_siguiente() is None:
            self._cabeza = None
        else:
            #Recorre la lista hasta encontrar el penúltimo nodo
            aux = self._cabeza
            while aux.get_siguiente().get_siguiente() is not None:
                aux = aux.get_siguiente()
            aux.set_siguiente(None)
        self._tamanio -= 1
```

Clase SmarthPone

```python
class SmartPhone:  9 usages  kno4
    def __init__(self, marca, modelo, precio):  kno4
        self.marca = marca
        self.modelo = modelo
        self.precio = precio
    def __str__(self):  kno4
        return f"{self.marca} {self.modelo} -${self.precio}"
```

Main

```python
from Tareas.Tarea4.ListaLigada import ListaLigada
from Tareas.Tarea4.SmartPhone import SmartPhone
def main():  1 usage  kno4
    lista_smartphones = ListaLigada()

    lista_smartphones.agregar_al_final(SmartPhone("Apple", "iPhone 14", 999))
    lista_smartphones.agregar_al_final(SmartPhone("Samsung", "Galaxy S21", 799))
    lista_smartphones.agregar_al_final(SmartPhone("Google", "Pixel 6", 599))
    lista_smartphones.agregar_al_final(SmartPhone("OnePlus", "9 Pro", 699))
    lista_smartphones.agregar_al_final(SmartPhone("Xiaomi", "Mi 11", 499))

    print("Contenido inicial de la lista:")
    lista_smartphones.transversal()

    lista_smartphones.eliminar(2)

    print("\nContenido después de eliminar el nodo en la posición 2:")
    lista_smartphones.transversal()

    nuevo_smartphone = SmartPhone("Sony", "Xperia 5", 899)
    lista_smartphones.actualizar(1, nuevo_smartphone)

    print("\nContenido después de actualizar el segundo elemento:")
    lista_smartphones.transversal()

    lista_smartphones.agregar_al_inicio(SmartPhone("Motorola", "Edge 20", 699))
    lista_smartphones.agregar_al_final(SmartPhone("Huawei", "P50 Pro", 999))

    print("\nContenido después de agregar al inicio y al final:")
    lista_smartphones.transversal()

    lista_smartphones.eliminar(0)

    print("\nContenido después de eliminar el primer nodo:")
    lista_smartphones.transversal()


if __name__ == '__main__':
    main()
```

Ejecución

```
Contenido inicial de la lista:
Apple iPhone 14 -$999 --> Samsung Galaxy S21 -$799 --> Google Pixel ó -$599 --> OnePlus 9 Pro -$699 --> Xiaomi Mi 11 -$499 -->


Contenido después de eliminar el nodo en la posición 2:
Apple iPhone 14 -$999 --> Samsung Galaxy S21 -$799 --> OnePlus 9 Pro -$699 --> Xiaomi Mi 11 -$499 -->


Contenido después de actualizar el segundo elemento:
Apple iPhone 14 -$999 --> Sony Xperia 5 -$899 --> OnePlus 9 Pro -$699 --> Xiaomi Mi 11 -$499 -->


Contenido después de agregar al inicio y al final:
Motorola Edge 20 -$699 --> Apple iPhone 14 -$999 --> Sony Xperia 5 -$899 --> OnePlus 9 Pro -$699 --> Xiaomi Mi 11 -$499 --> Huawei P50 Pro -$999 -->


Contenido después de eliminar el primer nodo:
Apple iPhone 14 -$999 --> Sony Xperia 5 -$899 --> OnePlus 9 Pro -$699 --> Xiaomi Mi 11 -$499 --> Huawei P50 Pro -$999 -->


Process finished with exit code 0
```