Listas doblemente enlazadas

Oscar Velandia Salgado

Estructura de datos

Uniciencia

Problemas propuestos

Plantear una clase para administrar una lista genérica doblemente encadenada implementando los siguientes métodos:

a) Insertar un nodo al principio de la lista.

```
public void insertarAlPrincipio(int x) {
        Nodo nuevo = new Nodo();
        nuevo.info = x;
        nuevo.sig = raiz;
        if (raiz != null) {
            raiz.ant = nuevo;
        }
        raiz = nuevo;
    }
```

b) Insertar un nodo al final de la lista.

```
public void insertarAlFinal(int x) {
        Nodo nuevo = new Nodo();
        nuevo.info = x;
        if (raiz == null) {
            raiz = nuevo;
        } else {
            Nodo reco = raiz;
            while (reco.sig != null) {
                reco = reco.sig;
            }
            reco.sig = nuevo;
            nuevo.ant = reco;
        }
    }
```

c) Insertar un nodo en la segunda posición. Si la lista está vacía no se inserta el nodo.

```
public void insertarEnSegundaPosicion(int x) {
        if (raiz != null && raiz.sig != null) {
            Nodo nuevo = new Nodo();
            nuevo.info = x;
            nuevo.sig = raiz.sig;
            nuevo.ant = raiz;
            raiz.sig.ant = nuevo;
            raiz.sig = nuevo;
        }
    }
```

d) Insertar un nodo en la ante última posición.

```
public void insertarEnAnteUltimaPosicion(int x) {
        if (raiz != null) {
            Nodo nuevo = new Nodo();
            nuevo.info = x;
            Nodo reco = raiz;
            while (reco.sig != null) {
                reco = reco.sig;
            }
            if (reco.ant != null) {
                nuevo.sig = reco;
                nuevo.ant = reco.ant;
                reco.ant.sig = nuevo;
                reco.ant = nuevo;
            } else {
                raiz.sig = nuevo;
                nuevo.ant = raiz;
            }
        }
```

```
    }
```

e) Borrar el primer nodo.

```
public void borrarPrimerNodo() {
        if (raiz != null) {
            raiz = raiz.sig;
            if (raiz != null) {
                raiz.ant = null;
            }
        }
    }
```

f) Borrar el segundo nodo.

```
    public void borrarSegundoNodo() {
        if (raiz != null && raiz.sig != null) {
            raiz.sig = raiz.sig.sig;
            if (raiz.sig != null) {
                raiz.sig.ant = raiz;
            }
        }
    }
```

g) Borrar el último nodo.

```
public void borrarUltimoNodo() {
        if (raiz != null) {
            Nodo reco = raiz;
            while (reco.sig != null) {
                reco = reco.sig;
            }
            if (reco.ant != null) {
                reco.ant.sig = null;
            } else {
                raiz = null;
```

```
            }
        }
    }
```

h) Borrar el nodo con información mayor.

```
public void borrarNodoMayor() {
        if (raiz != null) {
            Nodo reco = raiz;
            Nodo mayor = raiz;

            // Encontrar el nodo con información mas alto
            while (reco != null
            {
                if (reco.info > mayor.info) {
                    mayor = reco;
                }
                reco = reco.sig;
            }
            // Borrar el nodo con información del mas alto
            if (mayor.ant == null) {    // Si el nodo mayor es el primero
                raiz = mayor.sig;
                if (raiz != null) {
                    raiz.ant = null;
                }
            } else if (mayor.sig == null) {    // Si el nodo mayor seria el último
                mayor.ant.sig = null;
            } else { // Si el nodo mayor estária en el centro
                mayor.ant.sig = mayor.sig;
                mayor.sig.ant = mayor.ant;
            }
        }
```

```
}

// Método que imprime los elementos de la lista
public void imprimir() {
    Nodo reco = raiz;
    while (reco != null) {
        System.out.print(reco.info + "-");
        reco = reco.sig;
    }
    System.out.println();
}

public static void main(String[] ar) {
    ListaGenerica lg = new ListaGenerica();
    lg.insertarAlPrincipio(10);
    lg.insertarAlPrincipio(20);
    lg.insertarAlFinal(30);
    lg.insertarEnSegundaPosicion(15);
    lg.insertarEnAnteUltimaPosicion(25);
    lg.imprimir();
    lg.borrarPrimerNodo();
    lg.imprimir();
    lg.borrarSegundoNodo();
    lg.imprimir();
    lg.borrarUltimoNodo();
    lg.imprimir();
    lg.borrarNodoMayor();
    lg.imprimir();
}
```