UNIVERSIDAD TECNOLÓGICA DE PANAMÁCENTRO REGIONAL DE CHIRIQUÍ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

CARRERA:
Gestión y Desarrollo de Software

ACTIVIDAD No. 5

LABORATORIO No. 5

“Listas Enlazadas”

ASIGNATURA: Estructura de Datos I

DOCENTE:
Profa. Nunehar Mondul

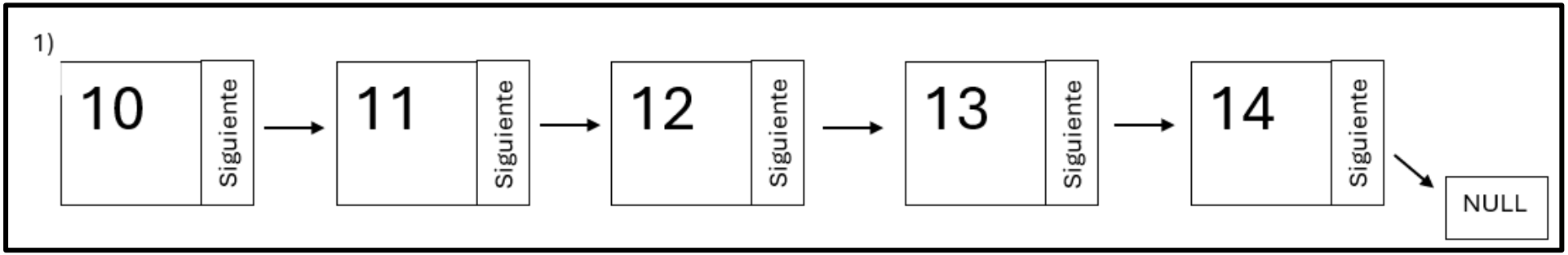
ESTUDIANTE:
Jorge Javier Jiménez Ruiz
4-826-874

II SEMESTRE 2024

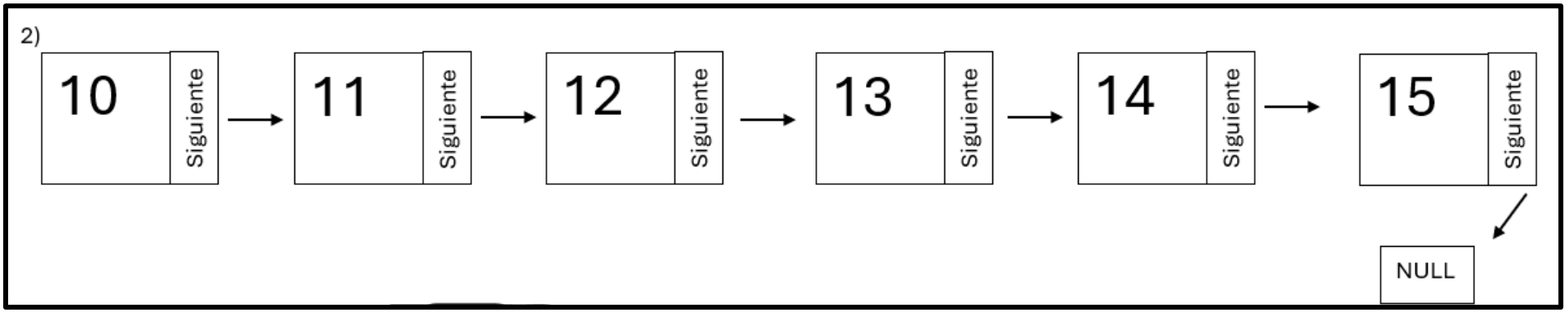
FECHA:
25/09/2024

Desarrollo

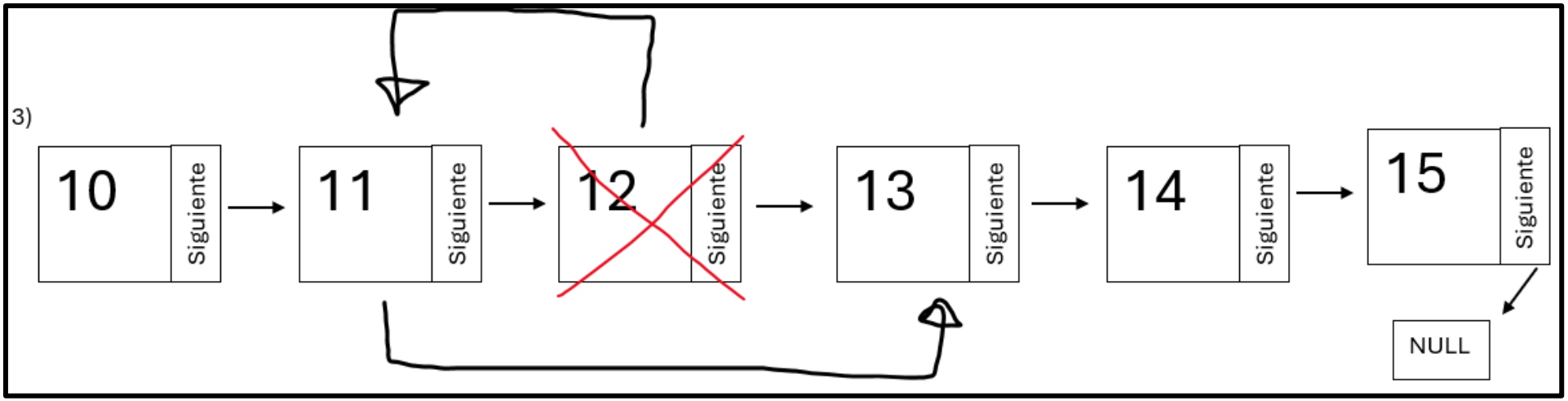
1)



2)



3)



4)
Memoria: A diferencia de los arrays, que requieren que los elementos estén almacenados de manera continua en memoria, las listas enlazadas pueden almacenar elementos dispersos, asignando y liberando memoria de manera dinámica. Esto evita la necesidad de reservar más memoria de la necesaria, como ocurre con los arrays que deben reservar un tamaño fijo desde el principio.

Velocidad: Las listas enlazadas permiten insertar o eliminar elementos en cualquier posición de la lista con solo ajustar los punteros, lo cual es más eficiente que en los arrays, donde insertar o eliminar un elemento puede requerir mover todos los elementos siguientes. En una lista enlazada, este proceso solo requiere cambiar los enlaces entre nodos

5) ***"Mencionar donde se verifica si la lista tiene los elementos o no."***
La verificación de si la lista contiene o no un elemento específico ocurre en el método "**borrar(int elem)**" en el código que se dio en clase.

Codigo en Texto (Nuevo):

```
public class Main {
   public static void main(String args[]) {

     listaSimple n1 = new listaSimple();
     n1.insertarPrimero(2);
     n1.insertarPrimero(1);
     n1.insertarFinal(3);
     n1.insertarFinal(4);

     System.out.println("Listamos desde main:");
     n1.listar();

     n1.insertarFinal(20);
     System.out.println("Listamos despues de agregar el nodo 20:");
     n1.listar();

     System.out.println("Borramos un Elemento (2):");
     n1.borrar(2);
     System.out.println("Borramos un Elemento (5):");
     n1.borrar(5);
     System.out.println("Volvemos a listar:");
     n1.listar();
```

```
    }
}

class Nodo {
    private int elemento;
    private Nodo siguiente;

    public Nodo(int elem, Nodo sig) {
        elemento = elem;
        siguiente = sig;
    }

    public int getElemento() {
        return elemento;
    }

    public Nodo getSig() {
        return siguiente;
    }

    public void setElemento(int elem) {
        elemento = elem;
    }

    public void setSig(Nodo sig) {
        siguiente = sig;
    }
}

class listaSimple {
    private Nodo primero;
    private int numElem;

    public listaSimple() {
        primero = null;
        numElem = 0;
    }
```

```
public void insertarPrimero(int elemento) {
    Nodo nuevo = new Nodo(elemento, primero);
    primero = nuevo;
    numElem++;
}

public void insertarFinal(int elemento) {
    Nodo nuevo = new Nodo(elemento, null);
    if (primero == null) {
        primero = nuevo;
    } else {
        Nodo actual = primero;
        while (actual.getSig() != null) {
            actual = actual.getSig();
        }
        actual.setSig(nuevo);
    }
    numElem++;
    System.out.println("Nodo insertado al final: " + elemento);
}

public void borrar(int elem) {
    if (primero == null) {
        System.out.println("Lista vacía");
    } else if (primero.getElemento() == elem) {
        primero = primero.getSig();
        numElem--;
        System.out.println("Elemento " + elem + " borrado.");
    } else {
        Nodo actual = primero;
        while (actual.getSig() != null && actual.getSig().getElemento() != elem) {
            actual = actual.getSig();
        }
        if (actual.getSig() == null) {
            System.out.println("Elemento " + elem + " no está en la lista.");
        } else {
```

```
                actual.setSig(actual.getSig().getSig());
                numElem--;
                System.out.println("Elemento " + elem + " borrado.");
            }
        }
    }

    public void listar() {
        if (primero == null) {
            System.out.println("La lista está vacía.");
        } else {
            Nodo actual = primero;
            while (actual != null) {
                System.out.println(actual.getElemento());
                actual = actual.getSig();
            }
        }
    }
}
```

[Codigo en OnlineGBD:]

```java
public class Main {
    public static void main(String args[]) {

      listaSimple n1 = new listaSimple();
      n1.insertarPrimero(2);
      n1.insertarPrimero(1);
      n1.insertarFinal(3);
      n1.insertarFinal(4);

      System.out.println("Listamos desde main:");
      n1.listar();

      n1.insertarFinal(20);
      System.out.println("Listamos despues de agregar el nodo 20:");
      n1.listar();

      System.out.println("Borramos un Elemento (2):");
      n1.borrar(2);
      System.out.println("Borramos un Elemento (5):");
      n1.borrar(5);
      System.out.println("Volvemos a listar:");
      n1.listar();

    }
}

class Nodo {
    private int elemento;
    private Nodo siguiente;

    public Nodo(int elem, Nodo sig) {
        elemento = elem;
        siguiente = sig;
    }

    public int getElemento() {
        return elemento;
    }

    public Nodo getSig() {
```

```
41          return siguiente;
42      }
43
44      public void setElemento(int elem) {
45          elemento = elem;
46      }
47
48      public void setSig(Nodo sig) {
49          siguiente = sig;
50      }
51  }
52
53  class listaSimple {
54      private Nodo primero;
55      private int numElem;
56
57      public listaSimple() {
58          primero = null;
59          numElem = 0;
60      }
61
62      public void insertarPrimero(int elemento) {
63          Nodo nuevo = new Nodo(elemento, primero);
64          primero = nuevo;
65          numElem++;
66      }
67
68      public void insertarFinal(int elemento) {
69          Nodo nuevo = new Nodo(elemento, null);
70          if (primero == null) {
71              primero = nuevo;
72          } else {
73              Nodo actual = primero;
74              while (actual.getSig() != null) {
75                  actual = actual.getSig();
76              }
77              actual.setSig(nuevo);
78          }
79          numElem++;
80          System.out.println("Nodo insertado al final: " + elemento);
```

```
81      }
82
83      public void borrar(int elem) {
84          if (primero == null) {
85              System.out.println("Lista vacía");
86          } else if (primero.getElemento() == elem) {
87              primero = primero.getSig();
88              numElem--;
89              System.out.println("Elemento " + elem + " borrado.");
90          } else {
91              Nodo actual = primero;
92              while (actual.getSig() != null && actual.getSig().getElemento() != elem) {
93                  actual = actual.getSig();
94              }
95              if (actual.getSig() == null) {
96                  System.out.println("Elemento " + elem + " no está en la lista.");
97              } else {
98                  actual.setSig(actual.getSig().getSig());
99                  numElem--;
100                 System.out.println("Elemento " + elem + " borrado.");
101             }
102         }
103     }
104
105     public void listar() {
106         if (primero == null) {
107             System.out.println("La lista está vacía.");
108         } else {
109             Nodo actual = primero;
110             while (actual != null) {
111                 System.out.println(actual.getElemento());
112                 actual = actual.getSig();
113             }
114         }
115     }
116 }
```

Resultados del Codigo:

```
Nodo insertado al final: 3
Nodo insertado al final: 4
Listamos desde main:
1
2
3
4
Nodo insertado al final: 20
Listamos despues de agregar el nodo 20:
1
2
3
4
20
Borramos un Elemento (2):
Elemento 2 borrado.
Borramos un Elemento (5):
Elemento 5 no está en la lista.
Volvemos a listar:
1
3
4
20


...Program finished with exit code 0
Press ENTER to exit console.
```