Laboratorio Nro. 4: Implementación de Listas Enlazadas

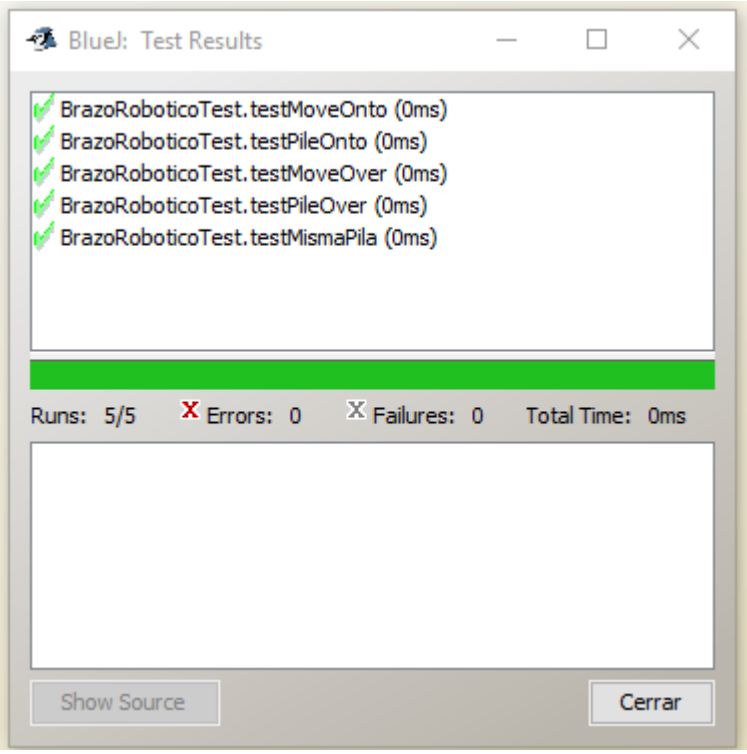
María Paula Chaparro Muñoz
Universidad Eafit
Medellín, Colombia
mpchaparrm@eafit.edu.co

Juan José Parra Díaz
Universidad Eafit
Medellín, Colombia
jjparrad@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. Pruebas con JUnit

Pruebas exitosas



Prueba move onto

```
@Test
public void testMoveOnto() {
    BrazoRobotico brazo = new BrazoRobotico(3);
    String[] com = {"move","1","onto","0"};
    brazo.leerComandos(com);
    String[] com2 = {"move","2","onto","0"};
    brazo.leerComandos(com2);
    assertEquals((long)brazo.stacks[0].pop(),(long)2);
    assertEquals((long)brazo.stacks[0].pop(),(long)0);
    assertEquals((long)brazo.stacks[1].pop(),(long)1);
}
```

Prueba move over

```
@Test
public void testMoveOver() {
    BrazoRobotico brazo = new BrazoRobotico(3);
    String[] com = {"move","1","over","0"};
    brazo.leerComandos(com);
    String[] com2 = {"move","2","over","0"};
    brazo.leerComandos(com2);
    assertEquals((long)brazo.stacks[0].pop(),(long)2);
    assertEquals((long)brazo.stacks[0].pop(),(long)1);
    assertEquals((long)brazo.stacks[0].pop(),(long)0);
}
```

Prueba pile onto

```
@Test
public void testPileOnto() {
    BrazoRobotico brazo = new BrazoRobotico(3);
    String[] com = {"pile","1","onto","0"};
    brazo.leerComandos(com);
    String[] com2 = {"pile","0","onto","2"};
    brazo.leerComandos(com2);
    assertEquals((long)brazo.stacks[2].pop(),(long)1);
    assertEquals((long)brazo.stacks[2].pop(),(long)0);
    assertEquals((long)brazo.stacks[2].pop(),(long)2);
}
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Prueba pile over

```
@Test
public void testPileOver() {
    BrazoRobotico brazo = new BrazoRobotico(4);
    String[] com = {"pile","1","over","0"};
    brazo.leerComandos(com);
    String[] com2 = {"pile","3","over","2"};
    brazo.leerComandos(com2);
    String[] com3 = {"pile","2","over","0"};
    brazo.leerComandos(com3);
    assertEquals((long)brazo.stacks[0].pop(),(long)3);
    assertEquals((long)brazo.stacks[0].pop(),(long)2);
    assertEquals((long)brazo.stacks[0].pop(),(long)1);
    assertEquals((long)brazo.stacks[0].pop(),(long)0);
}
```

Prueba movimiento en misma pila

```
@Test
public void testMismaPila() {
    BrazoRobotico brazo = new BrazoRobotico(2);
    String[] com = {"pile","1","over","1"};
    brazo.leerComandos(com);
    assertEquals((long)brazo.stacks[1].pop(),(long)1);
}
```

2. El ejercicio 2.1 lee un número y a partir de este crea un arreglo de ese número de posiciones. Luego en cada posición del arreglo crea una pila, a la cual le inserta el número que le corresponde, representando los bloques.

Tras esto, se lee nuevamente pero esta vez es un comando, el cual se divide en 4 partes, que son examinadas y de ahí se determinan: el bloque que se desea mover, el bloque al que se desea mover y el tipo de movimiento.

Luego, un método busca en las pilas los bloques deseados y guarda el número de pila para que más tarde sean utilizados.

Nuevamente se utiliza el comando que se ha dividido en 4 partes, y se leen las partes que corresponden a las instrucciones de movimiento, las cuales llaman al método de movimiento apropiado y le envían como parámetro los bloques que se desean mover, y las pilas en las que se encuentran.

Cada método de movimiento funciona de una manera diferente:

MoveOnto, devuelve elementos de la pila en la que se encuentra el bloque ‘a’ hasta que encuentra dicho elemento y hace lo mismo con la pila en la que esté el bloque ‘b’. Luego mueve el elemento ‘a’ sobre el elemento ‘b’.

MoveOver devuelve elementos de la pila en la que se encuentra el bloque ‘a’ hasta que encuentra dicho elemento y lo transporta sobre la pila del bloque ‘b’.

PileOnto crea una pila secundaria en la que inserta todos los elementos de la pila de ‘a’ hasta que encuentra aquel bloque. Luego devuelve todos los elementos que estén encima de ‘b’ y agrega cada elemento de la pila secundaria a la pila del bloque ‘b’.

PileOver crea una pila secundaria en la que inserta todos los elementos de la pila de ‘a’ hasta que encuentra aquel bloque. Tras esto, envía todos los elementos de la pila secundaria sobre la pila del bloque ‘b’.

Se repite este proceso hasta que, finalmente, se encuentra el comando “quit”, el cual finaliza la ejecución del programa.

3.

```
public class BrazoRobotico
{
    Stack<Integer>[] stacks;
    public BrazoRobotico(int n){                                    O(n)
        stacks = new Stack[n];                                      C1
        for(int i = 0; i < stacks.length; i++){                     C2*n
            Stack<Integer> stack = new Stack<Integer>();            C3*n
            stacks[i] = stack;                                      C4*n
            stacks[i].push(i);                                      C5*n
        }
    }
C’ + C*n
C*n
O(n)
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
    public void leerComandos(String[] comandos){                    O(n*m)
        //Leer comando
        int a = Integer.parseInt(comandos[1]);                      C1
        int b = Integer.parseInt(comandos[3]);                      C2
        String comandoUno = comandos[0];                            C3
        String comandoDos = comandos[2];                            C4
        //Decidir pilas
        int sa = decidirPila(a);                                    n*m
        int sb = decidirPila(b);                                    n*m
        if(sa == -1 || sb == -1){                                   C5
            System.out.println("No existen");                       C6
            return;                                                 C7
        } else if(sa == sb){                                        C8
            System.out.println("Están en la misma pila");           C9
            return;                                                 C10
        }
        Stack<Integer> sA = stacks[sa];                             C11
        Stack<Integer> sB = stacks[sb];                             C12
        if(comandoUno.equals("move")){                              C13
            if(comandoDos.equals("onto")){                          C14
                moveOnto(a, b, sA, sB);                             a+b
            } else if(comandoDos.equals("over")) {                  C16
                moveOver(a, b, sA, sB);                             a
            }
        } else if(comandoUno.equals("pile")){                       C18
            if (comandoDos.equals("onto")) {                        C19
                pileOnto(a, b, sA, sB);                             a+b
            } else if(comandoDos.equals("over")) {                  C21
                pileOver(a, b, sA, sB);                             a
            }
        } else {                                                    C23
            System.out.println("Comando inválido");                 C24
        }
    }
```

C' + C*(n*m) + C*(a+b) + c*a

C*(n*m) + C*(a+b)

C*(n*m)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

O(n*m)

```
public int decidirPila(int n){                                  O(n*m)
        int pila = -1;                                          C1
        for(int i = 0; i < stacks.length; i++){                 C2*n
            Stack<Integer> stack = stacks[i];                   C3*n
            if(stack.contains(n)){                              C4*n*m
                return i;                                       C5*m
            }
        }
        return pila;                                            C6
    }
```

C' + C*(n) + C*(n*m)
C*(n) + C*(n*m)
C*(n*m)
O(n*m)

```
    public void moveOnto(int a, int b, Stack<Integer> sa, Stack<Integer> sb){   O(a+b)
        while(sa.peek() != a){                                  C1*a
            devolver(sa.pop());                                 C2*a
        }
        while(sb.peek() != b){                                  C3*b
            devolver(sb.pop());                                 C4*b
        }
        sb.push(sa.pop());                                      C5
    }
```

C + C*(a) + C*(b)
C*(a+b)
O(a+b)

```
    public void moveOver(int a, int b, Stack<Integer> sa, Stack<Integer> sb){   O(a)
        while(sa.peek() != a){                                  C1*a
            devolver(sa.pop());                                 C2*a
        }
        sb.push(sa.pop());                                      C3
    }
```

C + C*a
C*a
O(a)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
    public void pileOnto(int a, int b, Stack<Integer> sa, Stack<Integer> sb){      O(a+b)
        Stack<Integer> losAmigosDeA = new Stack<Integer>();                        C1
        int n = sa.pop();                                                          C2
        while(n != a){                                                             C3*a
            losAmigosDeA.push(n);                                                  C4*a
            n = sa.pop();                                                          C5*a
        }
        losAmigosDeA.push(n);                                                      C6
        while(sb.peek() != b){                                                     C7*b
            devolver(sb.pop());                                                    C8*b
        }
        while(!(losAmigosDeA.isEmpty())){                                          C9*a
            sb.push(losAmigosDeA.pop());                                           C10*a
        }
    }
```

C' + C*(a) + C*(b)
C*(a+b)
O(a+b)

```
    public void pileOver(int a, int b, Stack<Integer> sa, Stack<Integer> sb){      O(a)
        Stack<Integer> losAmigosDeA = new Stack<Integer>();                        C1
        int n = sa.pop();                                                          C2
        while(n != a){                                                             C3*a
            losAmigosDeA.push(n);                                                  C4*a
            n = sa.pop();                                                          C5*a
        }
        losAmigosDeA.push(n);                                                      C6
        while(!(losAmigosDeA.isEmpty())){                                          C7*a
            sb.push(losAmigosDeA.pop());                                           C8*a
        }
    }
```

C' + C*a
C*a
O(a)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
    public void devolver(int n){                                    O(1)
        stacks[n].push(n);                                          C1
    }
C
O(1)

    public String toString(){                                       O(n*m)
        String s = "";                                              C1
        for(int i = 0; i < stacks.length; i++){                     C2*n
            Stack<Integer> secundario = new Stack<>();              C3*n
            s += i + ":";                                           C4*n
            while(!(stacks[i].isEmpty())){                          C5*n*m
                int a = stacks[i].pop();                            C6*n*m
                secundario.push(a);                                 C7*n*m
            }
            while(!(secundario.isEmpty())){                         C8*n*m
                int a = secundario.pop();                           C9*n*m
                s += " " + a;                                       C10*n*m
                stacks[i].push(a);                                  C11*n*m
            }
            s += "\n";                                              C12*n
        }
        return s;                                                   C13
    }
C' + C*(n) + C*(n*m)
C*(n) + C*(n*m)
C*(n*m)
O(n*m)

    public static void main(String [] args){
        System.out.println("Ingresa número de bloques");            C1
        Scanner in = new Scanner(System.in);                        C2
        int n = in.nextInt();                                       C3
        BrazoRobotico brazo = new BrazoRobotico(n);                 (n)
        System.out.println(brazo);                                  (n*m)
        in = new Scanner(System.in);                                C6
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

```
        String s = in.nextLine();                                   C7
        while(!(s.equals("quit"))){                                 C8*i
            String[] comandos = s.split(" ");                       C9*i
            try{
                brazo.leerComandos(comandos);                       (n*m)*i
            }catch(Exception e){
                System.out.println("Comando inválido");              C11*i
            }
            System.out.println(brazo);                              (n*m)
            s = in.nextLine();                                      C13*i
        }
    }
}
C' + (n*m) + (n*m*i)
O(n*m*i)
```

Complejidad en el peor de los casos:
O(n*m*i)

4. Análisis de variables en el cálculo de la complejidad
   - **a)** n: representa la cantidad de bloques iniciales.
   - **b)** m: es la altura de la pila más alta.
   - **c)** a: la cantidad de bloques sobre el bloque 'a' (incluyendo).
   - **d)** a: la cantidad de bloques sobre el bloque 'b' (incluyendo).
   - **e)** i: el número de comandos que se le dan al brazo robótico.

4) Simulacro de Parcial

1.
   - **a)** lista.size()
   - **b)** lista.add(auxiliar.pop());
2.
   - **a)** auxiliar1.size(), auxiliar2.size()
   - **b)** personas.offer(edad)

3. c