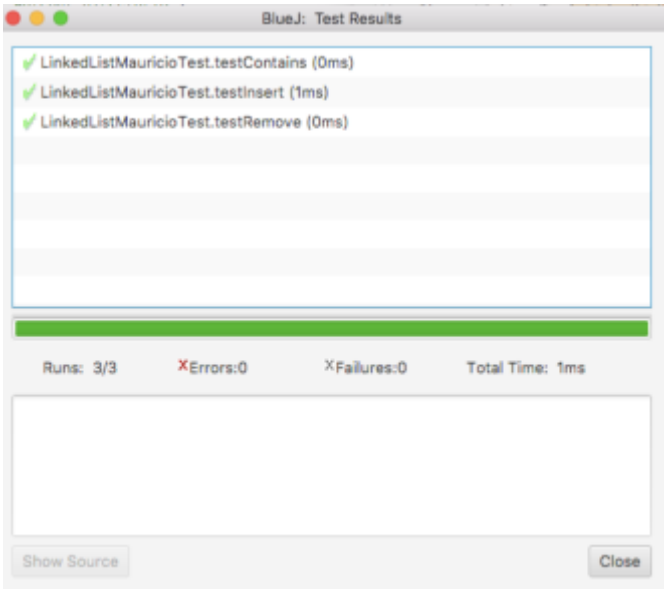
Laboratorio Nro. 4: Implementación de listas enlazadas

Isabela Muriel Roldán
Universidad Eafit
Medellín, Colombia
imurielr@eafit.edu.co

Mateo Flórez Restrepo
Universidad Eafit
Medellín, Colombia
mflorezr@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. **Test JUnit:**



2. **Complejidad:**

Método	Complejidad
pila()	O(n)
moveOnto()	$O(n^3)$
moveOver()	$O(n^3)$
pilaOnto()	$O(n^2)$
pilaOver()	$O(n^2)$

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

Pila():

```
public static void pila(int n,  Stack [] array ){
      for (int i=0; i<n; ++i){                                  //c+c*n
         Stack<Integer> pila = new Stack<Integer>();            //c*n
         pila.push(i);                                          //c*n
         array [i] = pila;                                      //c*n
      }
   }
```

moveOnto():

```
public static void moveOnto(int a, int b, Stack [] array){
      for(int j=0; j < array.length; ++j){                      //c+c*n
         if(array[j].contains(b)){                              //c*n
            if(array[j].contains(a)){                           //c*n
               System.out.println("Comando ilegal");           //c*n
               break;                                           //c*n
            }
            while(((int)array[j].peek())!=(b)){                 //c*n*n
               for(int i =0; i< array.length;++i){              //c*n*n
                  if(((int)array[j].peek())==(i)){              //c*n*n
                     array[i].push(array[j].pop());             //c*n*n
                  }
               }
            }
            for(int k=0; k < array.length; k++){                //c*n*n
               if(array[k].contains(a)){                        //c*n*n
                  while(((int)array[k].peek())!=(a)){           //c*n*n*m
                     for(int i =0; i< array.length;++i){        //c*n*n*n*m
                        if(((int)array[k].peek())==(i)){        //c*n*n*n*m
                           array[i].push(array[k].pop());       //c*n*n*n*m
                        }
                     }
                  }
                  array[j].push(array[k].pop());                //c*n
               }
            }
         }
      }
   }
```

moveOver():

```
public static void moveOver(int a, int b, Stack [] array){
      for (int i=0; i<array.length; ++i){                       //c+c*n
         if (array[i].contains(b)){                             //c*n
            if(array[i].contains(a)){                           //c*n
```

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

```
            System.out.println("Comando ilegal");          //c*n
            break;                                                  //c*n
          }
          for(int j= 0; j< array.length; j++){              //c*n*n
            if(array[j].contains(a)){                               //c*n*n
              while(((int)array[j].peek())!=(a)){                   //c*n*n*m
                for(int k =0; k< array.length;++k){                 //c*n*n*m*n
                  if(((int)array[j].peek())==(k)){          //c*n*n*m*n
                    array[k].push(array[j].pop());                  //c*n*n*m*n

                  }
                }
              }
              array[i].push(array[j].pop());                //c*n*n
            }
          }
        }
      }
    }
```

pilaOnto():

```
public static void pilaOnto(int a, int b, Stack [] array){
    for(int j=0; j < array.length; ++j){                    //c+c*n
      if(array[j].contains(b)){                                     //c*n
        if(array[j].contains(a)){                           //c*n
          System.out.println("Comando ilegal");          //c*n
          break;                                                    //c*n
        }
        while(((int)array[j].peek())!=(b)){                         //c*n*m
          for(int i =0; i< array.length;++i){               //c*n*n*m
            if(((int)array[j].peek())==(i)){                        //c*n*n*m
              array[i].push(array[j].pop());                        //c*n*n*m
            }
          }
        }
        for(int k=0; k < array.length; k++){                        //c*n*n
          if(array[k].contains(a)){                                     //c*n*n
            Stack <Integer> tmp = new Stack <Integer>();  //c*n*n
            while(((int)array[k].peek())!=(a)){                     //c*n*n*m
              tmp.push((int)array[k].pop());                        //c*n*n*m
            }
            array[j].push(array[k].pop());                      //c*n*n
            for(int l=0; l <= tmp.size(); l++){                     //c*n*n
              array[j].push(tmp.pop());                         //c*n*n
            }
          }
        }
      }
```

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

```
    }
  }
```

pilaOver():

```
public static void pilaOver(int a, int b, Stack [] array){
    for(int j=0; j < array.length; ++j){                          //c+c*n
      if(array[j].contains(b)){                                        //c*n
        if(array[j].contains(a)){                                  //c*n
          System.out.println("Comando ilegal");                    //c*n
          break;                                                           //c*n
        }
        for(int k=0; k < array.length; k++){                       //c*n*n
          if(array[k].contains(a)){                                        //c*n*n
            Stack <Integer> tmp = new Stack <Integer>();  //c*n*n
            while(((int)array[k].peek())!=(a)){                    //c*n*n
              tmp.push((int)array[k].pop());                       //c*n*n
            }
            array[j].push(array[k].pop());                         //c*n*m
            for(int l=0; l <= tmp.size(); l++){                            //c*n*n*m
              array[j].push(tmp.pop());                            //c*n*n*m
            }
          }
        }
      }
    }
  }
```

3. Funcionamiento del ejercicio en línea:

El ejercicio en línea que implementamos está basado en la modularización de un programa con 5 funciones y una entrada tipo numérica para manipular un arreglo que contiene n bloques que se trasladan según el comando que se elija.

La técnica que usamos fue hacer un menú con scanner que tiene las cinco funciones que nos piden en el problema: *move a onto b, move a over b, pile a onto b, pile a over b.* Pero antes de poder ejecutar las funciones el usuario debe ingresar el número N de bloques que desea manipular.

Luego de conocer el número de bloques el programa creara un arreglo de pilas que contiene el número n de pilas que va desde 0 <n-1 y dentro de cada pila se tiene el bloque correspondiente al número de pila, es decir, a la pila 0 le corresponde el bloque 0, a la pila 1 el bloque 1, y así sucesivamente. Después de ser creado el arreglo de pilas, se muestran las 5 funciones comentadas anteriormente con las que el usuario podrá controlar las apilaciones de bloques que suceden en el arreglo, por medio del menú numérico.

Cada función tiene su correspondiente método.

- ✓ Move a onto b: El método consiste en hacer que el bloque a este sobre el bloque b, siempre y cuando los demás bloques que estén apilados sobre el bloque a y /o el bloque b regresen a su posición original. El código del método buscara en todas las pilas, recorriendo el arreglo, donde este guardado el bloque a y si tiene bloques sobre él, los devuelve a su correspondiente posición, y luego hará lo mismo con el bloque b hasta que por último apile al bloque a sobre el b.

- ✓ Move a over b: En este caso se debe colocar el bloque a sobre el b siempre y cuando los bloques que estén sobre el bloque a sean retornados a su posición original. Igual que el método anterior,

este método busca en todas las pilas el bloque a y regresa cualquier bloque que haya sobre el a su correspondiente posición, y luego busca la pila en la que este el bloque b para colocar el bloque a en ella.

- ✓ Pile a onto b: Busca la pila en la que se encuentre el bloque a y la mueve toda incluyendo aquellos bloques que se encuentran sobre el bloque a, y la coloca sobre el bloque b, pero primero devolviendo aquellos bloques que están sobre b a su posición inicial. Para esta función el método uso una pila temporal que guardase todos los bloques que estuvieran sobre a, para luego moverlos a b en su orden correspondiente.

- ✓ Pile a over b: Busca la pila en la que se encuentra a y mueve el bloque a junto con los bloques que se encuentren sobre él a la pila en la que se encuentre b; al igual que el método anterior se utilizó una pila temporal para mover los bloques que estaban sobre a junto con a, sobre la pila de b, solo que esta vez aquellos bloques que estaban sobre b conservaron su posición.

- ✓ Quit: Este método termina el programa y sus funciones, e imprime el resultado final de los bloques y pilas después de haber sido manipulados, mostrando las n pilas junto con los bloques que contiene cada una.

  Nota: Cuando a y b se encuentran en la misma pila, al seleccionar uno de los 4 comandos de manipulación, se imprimirá "comando ilegal" que significa que no se ejecutó el comando y no hubo cambios y seguirá ejecutándose normalmente el programa.

4. **Variables n y m:** En el momento de calcular la complejidad de un algoritmo solo se utiliza n cuando hay una sola entrada, pero cuando hay dos entradas, por ejemplo, en dos ciclos anidados en los que se evalúan variables diferentes se utiliza la m para representar la segunda entrada.

4) Simulacro de Parcial

1. *a). lista.size()*
   b). lista.add(auxiliar.pop());

2. *a). auxiliar1.size()>0, auxiliar2.size()>0*
   b). personas.offer(edad);

3. *c) O(n^2)*
   a) El reporte de cambios del informe de laboratorio

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