Geeks México

BLOG DE PROGRAMACIÓN EN ESPAÑOL SOBRE JAVA, FRAMEWORKS, BASES DE DATOS, CÓMPUTO EN LA NUBE, ETC. EN ESPAÑOL Y EN INGLÉS.

Listas doblemente

ligadas en Java

HACE 1 SEMANA DEJA UN COMENTARIO

En el post anterior Listas ligadas en Java paso a paso y en Español (https://geeks-mexico.com/2017/11/20/listas-ligadas-en-java-paso-a-paso-y-en-espanol/) aprendimos como utilizar listas simplemente ligadas, en este explicaremos como crear listas doblemente ligadas en Java.

A diferencia de las listas simplemente ligadas las listas doblemente ligadas contienen una referencia al nodo siguiente y una al anterior, esto debe ser considerado al momento de agregar los valores al principio y al final de ella.

Creando la clase Nodo

```
1  public class Node {
2      private Integer value;
3      private Node nextElement;
4      private Node previousElement;
5
6      public Node(Integer value) {
7          this.value = value;
8      }
9
10     public Integer getValue() {
11         return value;
12     }
13
14     public void setValue(Integer value) {
15         this.value = value;
16     }
17
18     public Node getNextElement() {
19         return nextElement;
20     }
21
22     public void setNextElement(Node nextE
23         this.nextElement = nextElement;
24     }
25
26     public Node getPreviousElement() {
27         return previousElement;
```

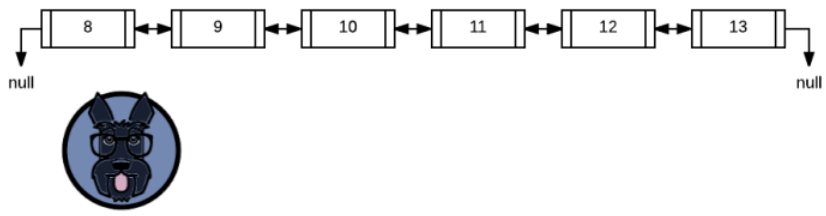
```
28      }
29
30      public void setPreviousElement(Node pr
31          this.previousElement = previousEle
32      }
33
34      @Override
35      public String toString() {
36          return "Node [value=" + value + ",
37                  : null) + ", previousEleme
38      }
39
40  }
```

Como se puede observar la clase nodo contiene 3 atributos:

- Integer value : Contiene el valor a almacenar en la lista

- Node nextElement: Contiene una referencia al nodo siguiente
- Node previousElement: Contiene una referencia al nodo anterior

Con estos atributos la lista doblemente ligada se verá como la siguiente imagen:



Creando la lista doblemente ligada

Una vez que se ha creado el nodo, el siguiente paso es crear la lista doblemente ligada.

```
1   public class DoublyLinkedList {
2       private Node tail;
3       private Node head;
4
5       public void addLast(Integer value) {
6           Node node = new Node(value);
```

```
7           if (tail == null && head == null)
8               tail = node;
9               head = node;
10          } else {
11              tail.setNextElement(node);
12              node.setPreviousElement(tail);
13              tail = node;
14          }
15      }
16
17      public void addFirst(Integer value) {
18          Node node = new Node(value);
19          if (tail == null && head == null)
20              tail = node;
21              head = node;
22          } else {
23              node.setNextElement(head);
24              head.setPreviousElement(node);
25              head = node;
26          }
27      }
28
29      public void print() {
30          for (Node i = head; i != null; i =
31              System.out.printf("\t %s ", i.
32          }
33          System.out.println();
34      }
35  }
```

El código anterior muestra como realizar las siguientes operaciones en una lista doblemente ligada:

- void addLast(Integer value) : Agrega un nodo al final de la lista
- void addFirst(Integer value) : Agrega un nodo al principio de la lista
- void print() : Imprime los elementos en la lista

Es importante mencionar que cada elemento de la lista debe mantener una referencia tanto al nodo siguiente como al nodo anterior.

Probando la lista

El último paso será probar la lista doblemente ligada, veamos el código:

```
1  public class TestDoublyLinkedList {
2      public static void main(String[] args)
3          DoublyLinkedList list = new Doubly
4          list.addLast(10);
5          list.addLast(11);
6          list.addLast(12);
7          list.addLast(13);
8          list.addLast(14);
9          list.addFirst(9);
10         list.addFirst(8);
11         list.print();
12     }
13 }
```

Salida:

```
1  Node [value=8, nextElement=9, previousEleme
```

La salida muestra por cada nodo su valor, su nodo anterior y su nodo siguiente.

Si te gusta el contenido compártelo y no olvides seguirnos en nuestras redes sociales https://twitter.com/geeks_mx (https://twitter.com/geeks_mx) y https://www.facebook.com/geeksJavaMexico/ (https://www.facebook.com/geeksJavaMexico/).

Autor: Alejandro Agapito Bautista

Twitter: @raidentrance

Contacto:raidentrance@gmail.com