
# Multimap Dictionary

PROJECT #8

## *Structured Implementation:*

## *Node:*

```
public class Node {
   public int key;
   public String value;
   public Node nextLink;

   //Link constructor
   public Node(int d1, String d2) {
      key = d1;
      value = d2;
   }
   //Print Link data
   public void printLink() {
      System.out.print(this.value+ " , ");}}
```

## *Linked List :*

```
public class LinkedList {
  private Node first;

  public LinkedList() {
      this.first = null;
   }

 public boolean isEmpty() {
   return first == null;
 }
 }
```

# *Function Insert :*

```
public void insert(int key, String value) {
      Node link = new Node(key,value);
      // first != null
      if(!isEmpty())
      {
        Node last = this.first;
        Node Current = this.first;

        while( key >= Current.key)
        {
           if(Current.nextLink != null)
           {
```

```
      last=Current;
      Current=Current.nextLink;
    }
    else
    {
       break;
    }
}
if(Current.key <= link.key)
    {
      //Next node Will Be Link
      Current.nextLink=link;
    }
else
    {
       // current (key) > Link (key)
       //check last == first
       if(last.key == this.first.key && last.value.equals(this.first.value))
       {
          // If last == first
          // Link.key < first.key
          //check if fisrt key < key or first.key > Key
          if(key < last.key)
          {
            Current=this.first;
            this.first=link;
            this.first.nextLink=Current;
          }
          // Link.key > first.eky
```

```
            else
            {
               this.first.nextLink=link;
               link.nextLink=Current;
            }
         }
         else
         {
            // If last != first
            //check if fisrt key < key or first.key > Key
            last.nextLink=link;
            link.nextLink=Current;
         }
         }
      }
   // first = null LinkedList => SIZE = 0
   else
   {
      first = link;
   }
}
```

## *Function Delete :*

```
public void delete(int key, String value) {
        if(deleteNode(key,value))
        {
           System.out.println("Data deleted Successfully");
        }
        else
        {
           System.out.println("Data Not Found");
        }
        printList();
    }



   private boolean deleteNode(int key, String value) {
       Node temp = first;
       Node Last = this.first;
       int count = 0;
       if(first == null){
        return false;
       }
       while(temp != null)
       {
           if (temp.key == key &&  value.equals(temp.value) )
       {
          if(temp.key == first.key && temp.value.equals(first.value))
          {
```

```
      if (temp.nextLink == null)
      {
      this.first = null;
      }
      else
      {
         this.first = temp.nextLink;
      }
   }
   else
   {
       if (temp.nextLink == null)
      {
       Last.nextLink = null;
      }
      else
      {
         Last.nextLink = temp.nextLink;
      }
   }
   count ++;
}
    Last = temp;
    temp = temp.nextLink;

 }
if(count ==0)
{ return false; }
else { return true; }}
```

## *Function Modify :*

```
public void modify(int key, String Oldvalue , String Newvalue) {
        if(modifyNode(key,Oldvalue,Newvalue))
        {
           System.out.println("Data Updated Successfully");
        }
        else
        {
           System.out.println("Data Not Found to Modify");
        }
        printList();
    }



    private boolean modifyNode(int key, String Oldvalue ,String Newvalue) {
        Node temp = first;
        Node Last = this.first;
        int count = 0;
        if(first == null){
         return false;
        }
        while(temp != null)
        {
            if (temp.key == key && temp.value.equals(Oldvalue) )
        {
           if(temp.key == first.key && temp.value.equals(first.value))
           {
```

```
        this.first.value = Newvalue;
        }
        else
        {
        temp.value = Newvalue;
        }
        count++;

    }
        Last = temp;
        temp = temp.nextLink;

     }
    if(count ==0)
    {
        return false;
    }
    else
    {
        return true;
    }

}
```

## *Print Function :*

```
public void printByKey(int key) {
        System.out.println("");
        if(!isEmpty())
        {
       Node currentLink = first;
       System.out.println("Node: ");
       while(currentLink != null) {
          if(currentLink.key == key)
          {
           currentLink.printLink();
          }
          currentLink = currentLink.nextLink;
       }
       System.out.println("");
   }
    }
```

```
//Prints list data
public void printList() {
        if(!isEmpty())
    {
   Node currentLink = first;
   System.out.print("List: ");
   System.out.println("");
   int key = currentLink.key;
   System.out.print("{ "+Integer.toString(currentLink.key)+" : ");
   while(currentLink != null) {
      if(currentLink.key == key)
      {

      }
      else
      {
         System.out.print("}");
         key = currentLink.key;
         System.out.println("");
         System.out.print("{ "+Integer.toString(currentLink.key)+" : ");

      }
      currentLink.printLink();
      currentLink = currentLink.nextLink;
   }
   System.out.print("}");

}
```

# *Main Code :*

```
public static void main(String[] args) {
        LinkedList L = new LinkedList();
        Scanner sc = new Scanner(System.in);
        Scanner sca= new Scanner(System.in);
        System.out.println("Hello This Multimap Dictionary Project ");
        char y = 'y';
        while(y == 'Y' || y == 'y')
        {
        System.out.println("Please Choose Operation : ");
        System.out.println("1 - Add pairs to the collection. ");
        System.out.println("2 - Remove pairs from the collection");
        System.out.println("3 - Modify the values of existing pairs");
        System.out.println("4 - Print values associated with a particular key.");
        System.out.println("5 - Print all linked list keys and  values .");
        System.out.println("");
        System.out.println("Enter Number Of Operation :");
        int choose = sc.nextInt();
        System.out.println("");

        int Key ;
        String Value ;

        switch(choose)
          {
          case 1:
             System.out.println("Please Enter Key : ");
```

```
    Key = sc.nextInt();
    System.out.println("");
    System.out.println("Please Enter Value : ");
    Value = sc.next();
    L.insert(Key, Value);
    L.printList();
    System.out.println("");
    break;
case 2:
    System.out.println("Please Enter Key : ");
    Key = sc.nextInt();
    System.out.println("");
    System.out.println("Please Enter Value : ");
    Value = sc.next();
    L.delete(Key, Value);
    System.out.println("");
    break;
case 3:
    System.out.println("Please Enter Key : ");
    Key = sc.nextInt();
    System.out.println("");
    System.out.println("Please Enter Value : ");
    Value = sc.next();
    System.out.println("");
    System.out.println("Enter New Value : ");
    String newValue = sc.next();
    L.modify(Key, Value, newValue);
    System.out.println("");
    break;
```

```
    case 4:
        System.out.println("Please Enter Key : ");
        Key = sc.nextInt();
        System.out.println("");
        L.printByKey(Key);
        System.out.println("");
        break;
    case 5:
        L.printList();
        System.out.println("");
        break;
    default:
        System.out.println("Sorry you Entered Not Valid Value");
        System.out.println("");
    }
  System.out.println("Do you Want Containue ? (Y/N)");
  y= sc.next().charAt(0);
}   }
```