Data Structures
DoubleLinkedList project

STAGE 1: **The Files** (Download CANVAS)

1) "MyDoubleLinkedList.java": **All the methods that you need to implement for this project must be added to this file.**
2) "TestMyDoubleLinkedList.java": The driver program that you will use to test the new "MyDOubleLinkedList.java"

STAGE 2: **The Methods (70 points – each method 5 points)**

Implement all the methods that are marked as "Left as Exercise". The Driver program only works when you have done all the methods. I have added the template of all the methods with a placeholder value as the return types of the methods. For example, return 0 if the method returns an int value.

List of the methods left as exercise are as follows:

```
/** Create a double linked ist from an array of
    objects */
  @SuppressWarnings("unchecked")
  public MyDoubleLinkedList(E[] objects)

/**
* Add a new element at the specified index in this list
* The index of the head element is 0
*/
public void add(int index, E e)

/** Add a new element at the specified index in this
list in ascending order */
  public void addInOrder(E e)

/** Check to see if this list contains element e */
  public boolean contains(E e)
```

Data Structures
DoubleLinkedList project

```
/** Remove the element at the specified position in this list.
* Return the element that was removed from the list.
*/
public E remove(int index)
```

```
/** Remove the first occurrence of the element e
  *  from this list. Return true if the element is
  * removed. */
  public boolean removeElement(E e)
```

```
/** Return the length of this list using recursion */
  public int getLength()
```

```
/** Print the list in reverse order */
public void PrintReverse()
```

```
/** Print this list using recursion */
  public void printList()
```

```
/** Return the element at the specified index */
  public E get(int index)
```

Data Structures
DoubleLinkedList project

```
/** Return the index of the head matching element in
   *  this list. Return -1 if no match. */
  public int indexOf(E e)
```

```
/** Return the index of the last matching element in
   *  this list. Return -1 if no match. */
  public int lastIndexOf(E e)
```

```
/** Replace the element at the specified position
   *  in this list with the specified element.
   *  throw exception if index out of bound and
   *  return null */
  public E set(int index, E e)
```

```
/** Split the original list in half. The original
  * list will continue to reference the
  * front half of the original list and the method
  * returns a reference to a new list that stores the
  * back half of the original list. If the number of
  * elements is odd, the extra element should remain
  * with the front half of the list. */
  public MyDoubleLinkedList<E> split()
```

Data Structures
DoubleLinkedList project

```
/** Check to see of two given lists are equal */
   public boolean equals(Object o)
    {

    // Hint: first cast o to MyDoubleLinkedList
    }
```

The expected output of TestMyDoubleLinkedList.java

```
esting addInOrder()
list1: [George, Jane, Jean, Peter, Tom]

Testing overloaded constructor
name1; [Tom, George, Peter, Jean, Jane]
list2: [Tom, George, Peter, Jean, Jane]

Testing equlas()
list2 equals temp: PASSED
list1 !equals temp: PASSED

Testing list1.set(1,"John")
list1: [George, Jane, Jean, Peter, Tom]
list1: [George, John, Jean, Peter, Tom]

Testing list1.set(10,"John")
list1: [George, John, Jean, Peter, Tom]
Index 10 out of bound.
Indedx out of bound: PASSESD

Testing printList() & printReverse()
Tom George Peter Jean Jane
Jane Jean Peter George Tom

Testing indexOf("Peter")
First index of Peter is 1: PASSED
```

Data Structures
DoubleLinkedList project

```
Testing lastIndexOf("Peter")
Last index of Peter is 4 PASSED

Testing contains()
list1 contains Bahram is false
list3 contains Peter is true

Testing getLength()
Length of list1 is 5 PASSED
Length of list3 is 7 PASSED

Testing removeElement("Peter")
list2: [Tom, George, Peter, Jean, Jane]
list2: [Tom, George, Jean, Jane]

Testing remove(1)
list3: [Tom, Peter, Jane, Adam, Peter, Mary, David]
list3: [Tom, Jane, Adam, Peter, Mary, David]

Testing get(2) and get(10)
[George, John, Jean, Peter, Tom]
Jean
null

Testing split()
[George, John, Jean, Peter, Tom]
[George, John, Jean]
[Peter, Tom]
```

SUBMISSION:

1) Create a folder called “Project3”
2) Copy and Paste “MyDoubleLinkedList.java” and “TestMyDoubleLinked.java” into the “project3” folder.
3) Compress the “Project3” folder
4) Submit the compressed folder (Canvas)