Answer to inlab on swapping doubly linked list from a DUMMY first node to no DUMMY nodes.

I did the add and print in class. So I decided to start with the addInOrder method:

addInOrder method:

The only thing you had to handle in the new addInOrder method was whether the list was null and whether the one you are adding was next to the first. So I added the following code to the beginning of the addInOrder method:

```
/*********
 First check to see if the list is null, if it is add the new data to start the list and then
return. Do nothing else in this method
 ********/
if(first == null)
{
        data.setNext(data);
        data.setPrevious(data);
        first = data;
        return;
}

    /********************
        This code is to check to see if the data we are adding is less than the first so this data will
        become the new first
***********************/
    if(data.getData().compareTo(first.getData()) <=0)
    {
        data.setNext(first);
        data.setPrevious(first.getPrevious());
```

```
            first.getPrevious().setNext(data);

            first.setPrevious(data);

            first = data;

        }


/****************************/

//The rest of the code in this method didn't change.
```

The remove Method:

The three local reference variables stay the same.

```
            Node<E> toRemove = first.getNext();

            Node<E> before;

            Node<E> after;

/********************************

        The first change is checking to see if the node that is being referenced by first and handle moving
first and deleting the node.

******************************/

        if(data.getData().compareTo(first.getData()) == 0)

        {

          //removing first

          first.getPrevious().setNext(first.getNext());

          first.getNext().setPrevious(first.getPrevious());

          first = first.getNext();

        }

/***********************

            The rest of the remove method stays the same.

**********************************/
```

Print Backwards:

I added the null check and the last print statement at the bottom.

Just those two things.

```
public void printBackwards()
  {
    if(first == null)
      return;
    Node<E> temp = first.getPrevious();
    while (temp != first)
    {
      System.out.println(temp.getData());
      temp = temp.getPrevious();
    }
    System.out.println(temp.getData());


  }
```