Assignment 4 — List Manipulator Impl

Nicholas White

ListManipulatorImpl.java
- size()
- sameSame()
- sublist()
- feed()
- superFeed()
- diff()
- delDiff()

```
public class ListManipulatorImpl<E extends Comparable<E>>
                                implements ListManipulatorIF<E> {
int counter;

@Override
   public int size(SingleLinkedListIF l) {
      counter = 0;
      Iterator<E> iter = l.iterator();
      while (iter.hasNext()){
         counter++;
         iter.next();
      }
      return counter;
   }

   @Override
   public boolean sameSame(SingleLinkedListIF l1, SingleLinkedListIF l2) {
      Iterator<E> iter1 = l1.iterator();
      Iterator<E> iter2 = l2.iterator();
      counter = 0;
      while (iter1.hasNext() && iter2.hasNext()){
         if (iter1.next() == iter2.next()){
            counter++;
         }
      }
      return (counter == size(l1) && counter == size(l2));
   }
```

```
@Override
public boolean sublist(SingleLinkedListIF<E> l1, SingleLinkedListIF<E> l2) {
    boolean sublist = false;
    try{
        Iterator<E> iter1 = l1.iterator();
        Iterator<E> iter2 = l2.iterator();



        if (l1.isEmpty()){
            sublist = true;
        }

        E temp = iter1.next();

        if (l2.find(temp) == -1){
            sublist = false;
        }
        if (l2.find(temp) != -1){
            while (iter1.hasNext()){
                if (iter2.next() == temp){
                    counter++;
                }
                temp = iter1.next();
            }
        }
        if (iter2.next() == temp){
```

```
                counter++;
            }
        if (counter == size(l1)){
            sublist = true;
        }
    }catch(NoSuchElementException e){};
    return sublist;
}

@Override
public void feed(SingleLinkedListIF l1, SingleLinkedListIF l2) throws
NoSuchElementException {
    try{
        if (l1.isEmpty()){
            throw new NoSuchElementException("List 1 is empty");
        }
        l2.insertFirst(l1.removeFirst());
    }catch(NoSuchElementException e){};
}

@Override
public void superFeed(SingleLinkedListIF l1, SingleLinkedListIF l2, int n) throws
NoSuchElementException {
    try{
        SingleLinkedListIF<E> temp = new SingleLinkedListImpl();
        if (l1.isEmpty()){
            throw new NoSuchElementException("List 1 is empty");
        }
```

```
        for (int i=0;i<n;i++){
            temp.insertFirst((E)l1.removeFirst());
        }
        for (int i=0;i<n;i++){
            l2.insertFirst((E)temp.removeFirst());
        }
    }catch(NoSuchElementException e){};
}

@Override
public SingleLinkedListIF diff(SingleLinkedListIF l1, SingleLinkedListIF l2) {
    SingleLinkedListIF<E> diff = new SingleLinkedListImpl();
    try{
        Iterator<E> iter1 = l1.iterator();
        Iterator<E> iter2 = l2.iterator();
        E temp = iter1.next();

        for (int i=0;i<size(l2);i++){

            for (int j=0;j<size(l1);j++){
                while (iter1.hasNext()){
                    if (l2.find(temp) == -1){
                        diff.insertFirst(temp);
                    }
                    temp = iter1.next();
                }
```

```
            }
        }
        if (l2.find(temp) == -1){
                diff.insertFirst(temp);
            }
        diff.display();
    } catch(NoSuchElementException e){};
    return (SingleLinkedListIF) diff;
}

@Override
public int delDiff(SingleLinkedListIF l1, SingleLinkedListIF l2) {
    counter = 0;
    try{
        Iterator<E> iter1 = l1.iterator();
        Iterator<E> iter2 = l2.iterator();
        E temp = iter1.next();

        for (int i=0;i<size(l2);i++){
            for (int j=0;j<size(l1);j++){
                while (iter1.hasNext()){
                    while (l2.find(temp) != -1){
                        l2.delete(temp);
                        counter++;
                    }
                    temp = iter1.next();
                }
```

```
            }
        }
        if (l2.find(temp) != -1){
            l2.delete(temp);
            counter++;
        }

    } catch(NoSuchElementException e){};
        return counter;
    }

}
```