CS 1083

Assignment #2

Author: Yulong Wang

Id: 3713596

1. Source Code

```java
import java.lang.reflect.Array;
import java.util.Scanner;

public class WishList {
    /**
     The items on the customer's wish list, sorted by sku.
    */
    private Item[] list;

    /**
     Constructs a new WishList given a sorted array of Items.
     @param listIn The list of items.
    */
    public WishList (Item[] listIn) {
        list = listIn;
    }

    /**
     Constructs a new WishList by reading the number of items and then
     the sorted list of item information using a Scanner; input format
     consists of a line with the number of items, followed by a line for
     each item containing values separated by commas
     @param scin The Scanner reading input.
    */
    public WishList (Scanner scin) {
        int count = scin.nextInt();
        scin.nextLine(); //read newline following the first int
        list = new Item[count];
        for(int i=0; i < count; i++){
            String s = scin.nextLine();
            Scanner scline = new Scanner(s);
            scline.useDelimiter(",");
            long sku = scline.nextLong();
            String name = scline.next();
            int priority = scline.nextInt();
            list[i] = new Item(name, sku, priority);
        }
    }

    /**
     This method return the largest index of an element in the array
     where the target is smaller or equals to.
     @param target The target number you want to search.
     @param array The array you are searching.
     @return The index of the element.
     */
    private int binarySearch(long target, Item[] array){
        if(array.length < 1){
            return -1;
        }
        int left = 0;
        int right = array.length - 1 ;
        while(left<=right) {
            int mid = (left + right) / 2;
            if (array[mid].getSKU() > target) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return right;
    }

    /**
     Returns the number of items that appear in only one of the two
     wish lists (this one and the other one that is passed in as a
     parameter).
     @param other The other wish list.
     @return The number of items that appear in only one of the two lists.
    */
    public int findUnique (WishList other){
        if(this.list.length < 1 || other.list.length < 1){
            return this.list.length + other.list.length;
        }
        int count = 0;
        Item[] smaller;
        Item[] larger;
//        find which of the array has a smaller head.
        if(this.list[0].getSKU() < other.list[0].getSKU()) {
            smaller = this.list;
            larger = other.list;
        }else{
            smaller = other.list;
            larger = this.list;
        }
//        find the index where the head of the larger array is in the smaller array.
        int smallerPointer = binarySearch(larger[0].getSKU(),smaller);
        int largerPointer = 0;
//        if the smaller array does not intercept or intercept at the end with larger
array, return the sum of the length.
        if(smallerPointer == smaller.length-1){
            if(smaller[smallerPointer].getSKU() == larger[0].getSKU()){
                return this.list.length + other.list.length -2;
            }else{
                return this.list.length + other.list.length;
            }

//        else, the two array has interception
        }else{
//            add the number of items that is smaller than the interception domain.
```

```
            count += smallerPointer;
//            while none of the pointer reaches the end of their array
            while (smallerPointer < smaller.length && largerPointer < larger.length){
                if (smaller[smallerPointer].getSKU() ==
larger[largerPointer].getSKU()){
                    smallerPointer += 1;
                    largerPointer += 1;
                }else if(smaller[smallerPointer].getSKU() <
larger[largerPointer].getSKU()){
                    smallerPointer += 1;
                    count += 1;
                }else{
                    largerPointer += 1;
                    count += 1;
                }
            }
//            add the number of items that is larger than the interception domain.
            count += (smaller.length - smallerPointer) + (larger.length -
largerPointer);
        }
        return count;
    }

    /**
     Merges this wish list with another one (passed in as a parameter),
     producing a new sorted wish list.
     @param other The wish list to be merged with this wish list.
     @return The merged wish list.
    */
    public WishList merge (WishList other){
        Item[] res = new Item[this.list.length + other.list.length];
//        check boundary situation
        if(this.list.length<1 || other.list.length <1){
            for(int i = 0; i < this.list.length;i++){
                res[i] = this.list[i];
            }
            for(int i = 0; i < other.list.length;i++){
                res[i] = other.list[i];
            }
            this.list = res;
            return this;
        }
//        Here, the order does not matter.
        Item[] smaller = this.list;
        Item[] larger = other.list;
        int count = 0;
        int smallerPointer = 0;
        int largerPointer = 0;
//        while none of the pointer reaches the end of their array
        while (smallerPointer < smaller.length && largerPointer < larger.length){
            if (smaller[smallerPointer].getSKU() == larger[largerPointer].getSKU()){
                res[count] = smaller[smallerPointer];
                count+=1;
                smallerPointer += 1;
                res[count] = larger[largerPointer];
                count+=1;
                largerPointer += 1;
            }else if(smaller[smallerPointer].getSKU() <
larger[largerPointer].getSKU()){
                res[count] = smaller[smallerPointer];
                smallerPointer += 1;
                count += 1;
            }else{
                res[count] = larger[largerPointer];
                largerPointer += 1;
                count += 1;
            }
        }
//        add items that is larger than the interception domain.
        for(int i = smallerPointer; i < smaller.length;i++){
            res[count] = smaller[i];
            count +=1;
        }
        for(int i = largerPointer; i < larger.length;i++){
            res[count] = larger[i];
            count +=1;
        }
        this.list = res;
        return this;
    }

    /**
    Updates the wish list by adding the item passed in as a parameter to
    the wish list in the correct order if the item is not already in the list.
    @param newItem The item to be added to this wish list.
    @return If item was added successfully or not.
    */
    public boolean addItem (Item newItem){
//        check boundary condition
        if(this.list.length == 0){
            this.list = new Item[] {newItem};
            return true;
        }
        Item[] res = new Item[this.list.length+1];
        int i = 0;
        int j = 0;
//        if the newItem is the smallest in the original list.
        if (newItem.getSKU() < this.list[0].getSKU()){
            res[0] = newItem;
            j += 1;
        }
        while(i < this.list.length) {
            if (i + 1 < this.list.length
                    && this.list[i].getSKU() < newItem.getSKU()
```

```
                     && this.list[i + 1].getSKU() > newItem.getSKU()) {
                 res[j] = this.list[i];
                 res[j + 1] = newItem;
                 res[j + 2] = this.list[i + 1];
                 i += 2;
                 j += 3;
             } else {
                 res[j] = this.list[i];
                 i += 1;
                 j += 1;
             }
         }
//        if the new item is the largest in the original list
         if (newItem.getSKU() > this.list[this.list.length-1].getSKU()){
             res[res.length-1] = newItem;
         }
//        if the new item already exists
         if(res[res.length-1] == null){
             return false;
         }else{
             this.list = res;
             return true;
         }
     }

     public String toString(){
         String s = "";
         for(int i=0; i < list.length; i++){
             s += list[i].getSKU() + "\t" + list[i].getName() + "\t"
                 + list[i].getPriority() + "\n";
         }
         return s;
     }

}
```

2. Input and Associated Output for All 5 Test Cases

Note: The explaination of the coverage of each test case is include here.

a. TestCase1.dat:

This test case covers:

1. Adding item to the middle of a wishlist
2. Find unique item of two wishlists with repeated item between them.
3. Merge two wishlist with repeated item between them.

This test case test the normal condition that the algorithm should work.

```
4
11039926010,Digital Kitchen Scale,3
11798411010,KitchenAid Stand Mixer,1
24179114710,Autumn Plaid Tablecloth,2
96796133410,Tan Cotton Blanket,2
4
11781701910,Cast Iron Round Griddle,2
11798009510,Espresso Machine,1
11798112010,NutriBullet Blender,1
11798411010,KitchenAid Stand Mixer,2
```

Output:

```
(base) yulongwang@YulongdeMacBook-Pro src % java TestWishList TestCase1.dat
List 1:
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

Item added to List 1:
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

List 2:
11781701910     Cast Iron Round Griddle 2
11798009510     Espresso Machine        1
11798112010     NutriBullet Blender     1
11798411010     KitchenAid Stand Mixer  2

There are 7 items found in one wish list but not the other
Merged wish lists:
11039926010     Digital Kitchen Scale   3
11781701910     Cast Iron Round Griddle 2
11798009510     Espresso Machine        1
11798112010     NutriBullet Blender     1
11798411010     KitchenAid Stand Mixer  1
11798411010     KitchenAid Stand Mixer  2
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2
```

b. TestCase2.dat:

This test case covers:

1. Adding item to a empty list.
2. Find unique item when one of the wishlist is empty.
3. Merge two wishlist when one of the wishlist is empty.

This test case test the boundary condition where one of the list is empty.

```
0
0
```

Output:

```
(base) yulongwang@YulongdeMacBook-Pro src % java TestWishList TestCase2.dat
List 1:

Item added to List 1:
11881701910     Rice Cooker & Steamer   2

List 2:

There are 1 items found in one wish list but not the other
Merged wish lists:
11881701910     Rice Cooker & Steamer   2
```

c. TestCase3.dat:

This test case covers:

1. Find unique item when the two wishlist is the same.
2. Merge two wishlist when the two wishlist is the same.

This test case test the boundary condition where two list are the same.

```
4
11039926010,Digital Kitchen Scale,3
11798411010,KitchenAid Stand Mixer,1
24179114710,Autumn Plaid Tablecloth,2
96796133410,Tan Cotton Blanket,2
5
11039926010,Digital Kitchen Scale,3
11798411010,KitchenAid Stand Mixer,1
11881701910,Rice Cooker & Steamer,2
24179114710,Autumn Plaid Tablecloth,2
96796133410,Tan Cotton Blanket,2
```

Output:

```
(base) yulongwang@YulongdeMacBook-Pro src % java TestWishList TestCase3.dat
List 1:
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

Item added to List 1:
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
```

```
96796133410     Tan Cotton Blanket      2

List 2:
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

There are 0 items found in one wish list but not the other
Merged wish lists:
11039926010     Digital Kitchen Scale   3
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2
96796133410     Tan Cotton Blanket      2
```

d. TestCase4.dat:

This test case covers:

1. Add item to the list where already exist
2. Find unique item when the two wishlist does not have repeated items.
3. Merge two wishlist when the two wishlist does not have repeated items.

This test case test the boundary condition where two list are totally different.

```
5
11039926010,Digital Kitchen Scale,3
11798411010,KitchenAid Stand Mixer,1
11881701910,Rice Cooker & Steamer,2
24179114710,Autumn Plaid Tablecloth,2
96796133410,Tan Cotton Blanket,2
3
11781701910,Cast Iron Round Griddle,2
11798009510,Espresso Machine,1
11798112010,NutriBullet Blender,1
```

Output:

```
(base) yulongwang@YulongdeMacBook-Pro src % java TestWishList TestCase4.dat
List 1:
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

Item not added to List 1:
```

```
11039926010     Digital Kitchen Scale   3
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

List 2:
11781701910     Cast Iron Round Griddle 2
11798009510     Espresso Machine        1
11798112010     NutriBullet Blender     1

There are 8 items found in one wish list but not the other
Merged wish lists:
11039926010     Digital Kitchen Scale   3
11781701910     Cast Iron Round Griddle 2
11798009510     Espresso Machine        1
11798112010     NutriBullet Blender     1
11798411010     KitchenAid Stand Mixer  1
11881701910     Rice Cooker & Steamer   2
24179114710     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2
```

e. TestCase5.dat:

This test case covers:

1. Add item to the begining of the wishlist
2. Find unique item when the two wishlist has repeated item on boundary.
3. Merge two wishlist when the two wishlist has repeated item on boundary.

This test case test the boundary condition where two list have overlap item on their boundary.

```
4
24179114701,Digital Kitchen Scale,3
24179114702,KitchenAid Stand Mixer,1
24179114703,Autumn Plaid Tablecloth,2
96796133410,Tan Cotton Blanket,2
2
96796133410,Tan Cotton Blanket,2
96796133411,Ipad,2
```

Output

```
(base) yulongwang@YulongdeMacBook-Pro src % java TestWishList TestCase5.dat
List 1:
24179114701     Digital Kitchen Scale   3
24179114702     KitchenAid Stand Mixer  1
24179114703     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

Item added to List 1:
11881701910     Rice Cooker & Steamer   2
24179114701     Digital Kitchen Scale   3
```

```
24179114702     KitchenAid Stand Mixer  1
24179114703     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2

List 2:
96796133410     Tan Cotton Blanket      2
96796133411     Ipad    2

There are 5 items found in one wish list but not the other
Merged wish lists:
11881701910     Rice Cooker & Steamer   2
24179114701     Digital Kitchen Scale   3
24179114702     KitchenAid Stand Mixer  1
24179114703     Autumn Plaid Tablecloth 2
96796133410     Tan Cotton Blanket      2
96796133410     Tan Cotton Blanket      2
96796133411     Ipad    2
```