07.06 Virtual Lecture Notes

Array Insertion

An insertion algorithm for arrays is always a little tricky, due to the fact that an array has a fixed length. Ideally, we would like to insert an item into a certain location of an array and then just shift the remaining elements down the array. The problem arises with the last element. If the array is full, then the last element will be lost.

Let start with five items in our array in the TestInventory5 class.

```
InventoryItem[] inventory = new InventoryItem[5];

// create inventory
inventory[0] = new InventoryItem("Towel", 200);
inventory[1] = new InventoryItem("Cleaning Cart", 30);
inventory[2] = new InventoryItem("Toiletry Sets", 100);
inventory[3] = new InventoryItem("Coffee Set", 300);
inventory[4] = new InventoryItem("Pillows", 50);
```

If we want to insert Relaxation Kits at index 2, where will Pillows go? Since the array is full, the Pillows item will be lost. The array cannot expand.

First, we start by calling the method to add the Relaxation Kits with a quantity of 1000 into position 2.

```
insertItem1(inventory, 2, "Relaxation Kit", 1000);
```

So in order to insert Relaxation Kits into our inventory, we could write a method like this:

```
public static void insertItem1(InventoryItem[] itemList, int
location,
                                String addN, int addS)
```

```
{
    //move items down in the array - last item is lost
    for(int index = itemList.length - 1; index > location;
index--)
        itemList[index] = itemList[index-1];

    itemList[location] = new InventoryItem(addN, addS);
}
```

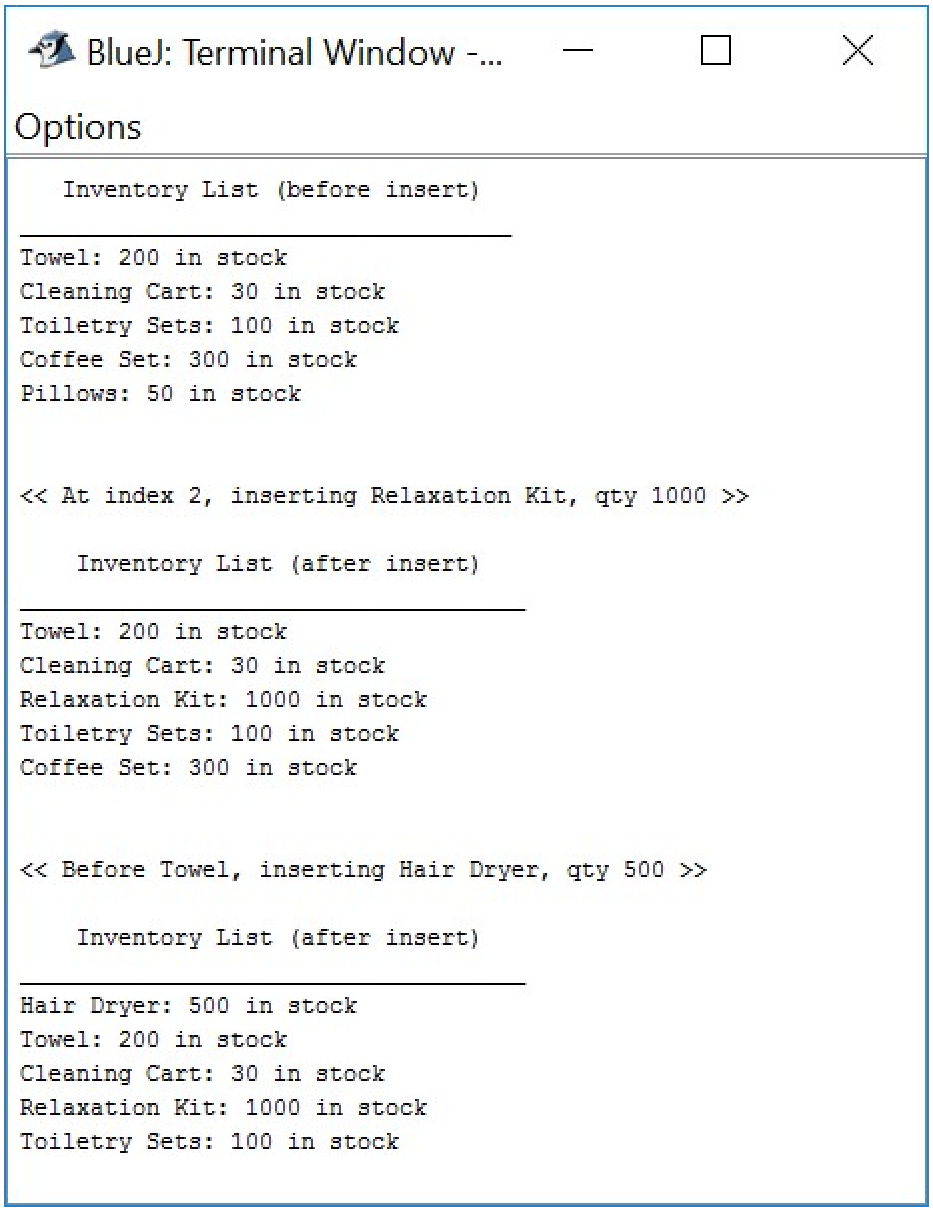
Look closely at the for loop. Notice that `index` starts at the last position of the array and moves the items down the array, stopping right before the location where we want to add our new item. Then the new inventory item is added to the requested location.

You can also choose where to insert, based upon the name of an inventory item. The following method does that:

```
public static void insertItem2(InventoryItem[] itemList, String
find,
                               String addN, int addS)
{
    int location = 0;

    // find location of item you want to insert before
    for(int index = 0; index < itemList.length; index++)
        if(itemList[index].getName().equals(find))
            location = index;

    //move items down in the array - last item is lost
    for(int index = itemList.length - 1; index > location;
index--)
        itemList[index] = itemList[index-1];

    itemList[location] = new InventoryItem(addN, addS);
}
```

In this method, we first have to find the location of the item in front of which we want to insert. Then we insert in a similar manner as the previous insert method.

If we test it by first printing an inventory list before changing and then after changing, we get the following output:



```
   Inventory List (before insert)
___________________________________
Towel: 200 in stock
Cleaning Cart: 30 in stock
Toiletry Sets: 100 in stock
Coffee Set: 300 in stock
Pillows: 50 in stock


<< At index 2, inserting Relaxation Kit, qty 1000 >>

    Inventory List (after insert)
____________________________________
Towel: 200 in stock
Cleaning Cart: 30 in stock
Relaxation Kit: 1000 in stock
Toiletry Sets: 100 in stock
Coffee Set: 300 in stock


<< Before Towel, inserting Hair Dryer, qty 500 >>

    Inventory List (after insert)
____________________________________
Hair Dryer: 500 in stock
Towel: 200 in stock
Cleaning Cart: 30 in stock
Relaxation Kit: 1000 in stock
Toiletry Sets: 100 in stock
```

Be sure to run the program and try changing the calls for inserting items.

ArrayList Insertion

Now, how about an `ArrayList`? When we are using an `ArrayList`, we do not have to drop the last item. We can insert the Relaxation Kits and keep all the items we had before the insert. The size of the `ArrayList` will increase by one with each added item.

The first method would be modified like this:

```
public static void insertItem1(List<InventoryItem> itemList, int
location,
                                String addN, int addS)
{
    // insert item into ArrayList
    list.add(location, new InventoryItem(addN, addS));
}
```

Since we are using an `ArrayList`, only the add method is needed. So insertion at a specific index is extremely easy.

The second way of inserting is also simplified when using an `ArrayList`.

```
public static void insertItem2(List<InventoryItem> itemList,
String find,
                                String addN, int addS)
{
    int location = 0;

    // find location of item you want to insert before
    for(int index = 0; index < itemList.size(); index++)
        if(itemList.get(index).getName().equals(find))
            location = index;

    // insert item into ArrayList
    itemList.add(location, new InventoryItem(addN, addS));
}
```

Notice this method begins exactly as in the previous example for arrays, and then the item is inserted using the `add` method.

The output from the `ArrayList` test program is a bit different than that of the array program, as no items are lost as a result of the insertion. That means the size of the list gets longer. Run the TestInventory6.java program to see for yourself!

Print