

16.03 Virtual Lecture Notes

An insertion algorithm for arrays is always a little tricky, due to the fact that an array has a fixed size. 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 moved. If the array is full, then the last element will be lost.

Let us assume we have the same five items as in the previous lesson:

```
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 would Pillows go? Since the array is full, Pillows would be lost. Since the array cannot be expanded, that is what would happen in this instance.

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

```
public static void insertItem1(InventoryItem[] list,
                               int location, String addN, int addS)
{
    //move items down in the array - last item is lost
    for(int index = list.length - 1; index > location; index--)
        list[index] = list[index-1];

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

Notice that the 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.

To add our Relaxation Kits, we would call our method like this:

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

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[] list,
```

```
String find, String addN, int dds)
{
    int location = 0;

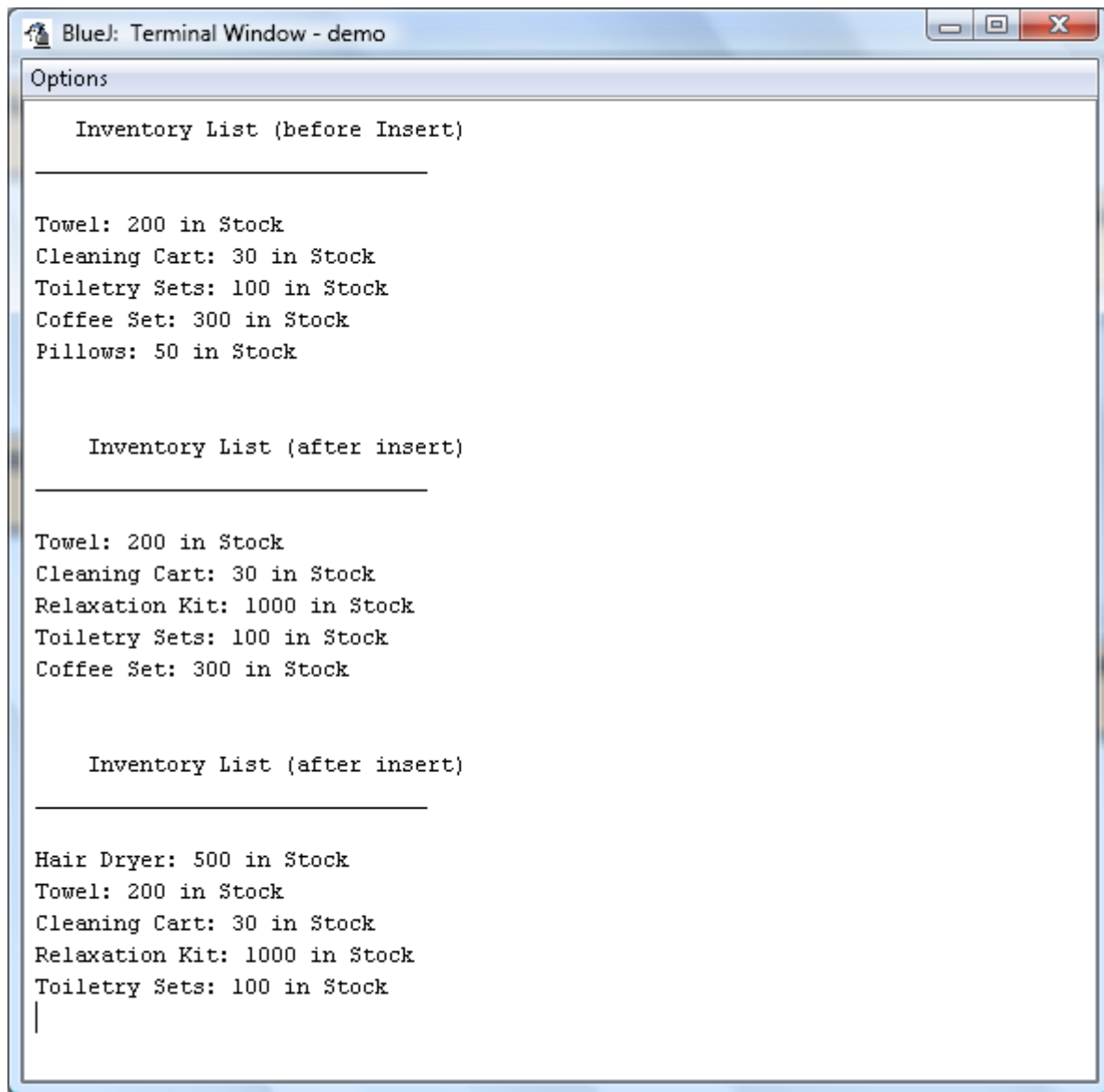
    // find location of item you want to insert before
    for(int index = 0; index < list.length; index++)
        if (list[index].getName().equals(find))
            location = index;

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

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

Notice that in this method, we first have to find the location of the item in front of which we want to insert. Then we insert the same way as in the first insert item method shown above.

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



```
BlueJ: Terminal Window - demo
Options

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

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

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
|
```

- Download the TestInventory5.java file to your module 16 demo programs directory and open it.
- Run the file and make sure you understand it before you continue.

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. This is more like the way Waylan would add inventory items.

The first method would be modified like this:

```
public static void insertItem1(ArrayList<InventoryItem> list,
                               int location, String addN, int addS)
{
    // insert item into ArrayList
```

```
list.add(location, new InventoryItem(addN, addS));  
}
```

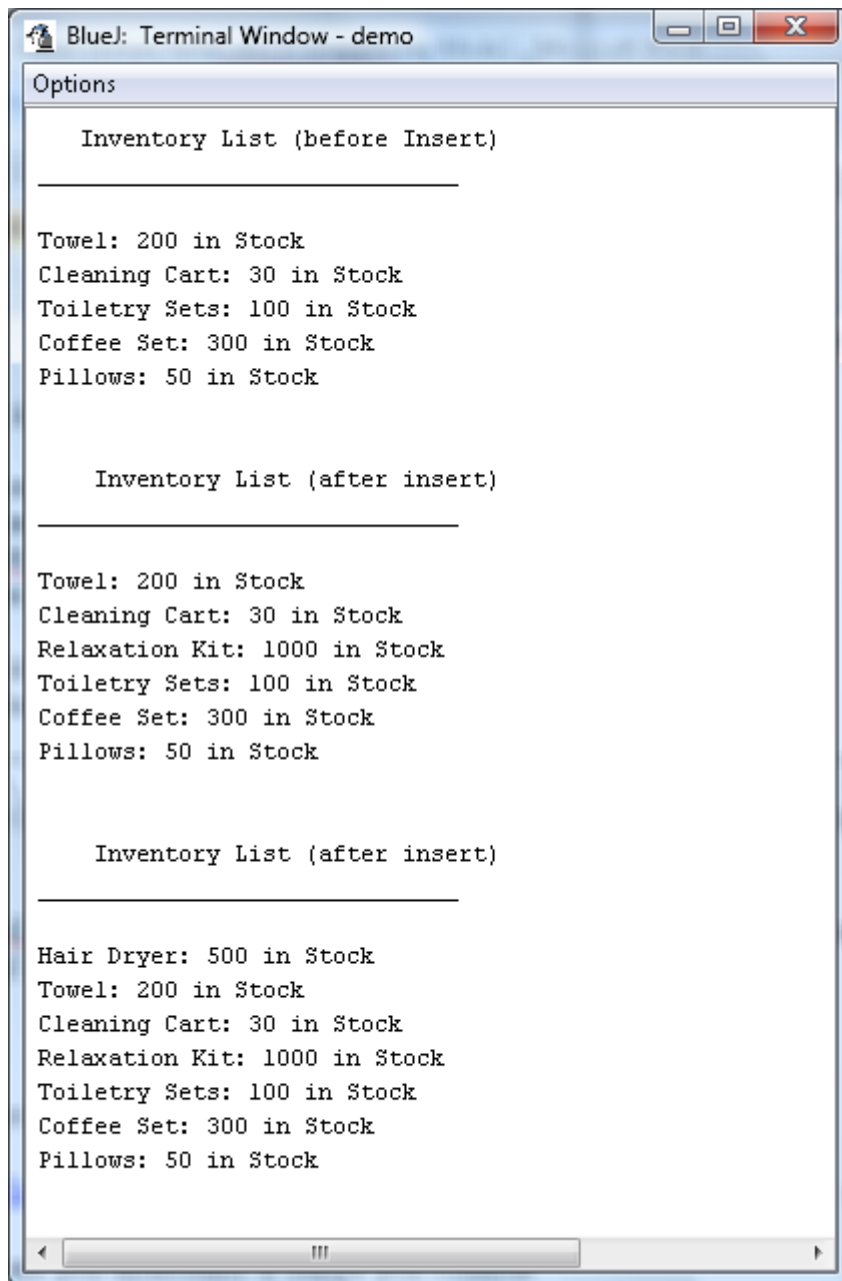
Since we are using an ArrayList, in this first version, we only need to use the **add()** method of the ArrayList. So insertion in this case is extremely easy.

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

```
public static void insertItem2(ArrayList<InventoryItem> list,  
                                String find, String addN, int addS)  
{  
    int location = 0;  
  
    // find location of item you want to insert before  
    for(int index = 0; index < list.size(); index++)  
        if (list.get(index).getName().equals(find))  
            location = index;  
  
    // insert item into ArrayList  
    list.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 much different than that of the array program, as no items are lost as a result of the insertion.



```
BlueJ: Terminal Window - demo
Options

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

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
Pillows: 50 in Stock

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
Coffee Set: 300 in Stock
Pillows: 50 in Stock
```

Take a look at the demo program **TestInventory6.java**.

- Download the TestInventory6.java file to your module 16 demo programs directory and open it.
- Run the file and make sure you understand it before you continue.