

<b>We want to go shopping this exercise will make a list</b>	
<b>Create a project in eclipse called ListOfItems.</b>	<ol style="list-style-type: none"><li>1) File → New → Java Project</li><li>2) Give it a name</li><li>3) Click finish</li></ol>
<b>Create a package to group all of our classes from this project</b>	<ol style="list-style-type: none"><li>1) Right click on the ListOfItems project</li><li>2) Choose New → package</li><li>3) Notice it is in the source folder for the project you created ListOfItems</li><li>4) Give the package a name of: com.cosc210</li></ol>
We want to go to a store or to a web-site or shop over the phone and purchase an item or many items.  What is the first class we will need? <b>The Item class</b>	<ol style="list-style-type: none"><li>1) In the new project you created right click on the package you just created:</li><li>2) Choose File → New → class</li><li>3) Give it a name of Item and choose the Finish button</li></ol> <pre>public class Item {</pre>
What might be needed for each Item we want to purchase? Name or description Price Quantity Etc... There might be many more, let us start with those three.  Remember: most instance variables will be private	<ol style="list-style-type: none"><li>1) add the three instance variables below to Item</li></ol> <pre>package com.cosc210;  Public class Item {     private String name = "default";     private double cost = 0.0;     private int quantity = 0; }</pre>

<p>If our instance variables are private what do we need to get at them?</p> <p>Getters and Setters</p> <ol style="list-style-type: none"><li>1) In eclipse go to the Item.java source file you created as if you were going to edit it.</li><li>2) Right click in the source file</li><li>3) Go to the source menu</li><li>4) Choose generate getters and setters and you will see what is on the right →</li></ol> <p>Notice: they are public and they allow us to give our private instance variables a value and obtain whatever value is in our private instance variables</p>	<pre>package com.cosc210;  public class Item {     private String name = "default";     private double cost = 0.0;     private int quantity = 0;      public String getName() {         return name;     }     public void setName(String name) {         this.name = name;     }     public double getCost() {         return cost;     }     public void setCost(double cost) {         this.cost = cost;     }     public int getQuantity() {         return quantity;     }     public void setQuantity(int quantity) {         this.quantity = quantity;     } }</pre>
<p>Because we need to get into the habit of always having a default constructor, add a default constructor to Item.</p> <p>Whenever you start creating constructors with parameters for a class, you should always add a default constructor. No implicit default constructor is added as soon as you create a constructor with parameters.</p> <p>The implicit super() call to a super class will cause issues if there is no default constructor in the super class. For this reason we make it a habit to add default constructors to our classes.</p>	<pre>package com.cosc210;  public class Item {     private String name = "default";     private double cost = 0.0;     private int quantity = 0;      public Item(){     }     public String getName() {         return name;     }     public void setName(String name) {         this.name = name;     }     public double getCost() {         return cost;     }     public void setCost(double cost) {         this.cost = cost;     }     public int getQuantity() {         return quantity;     }     public void setQuantity(int quantity) {         this.quantity = quantity;     } }</pre>

<p>We have a single Item defined, now what?</p>	<p>We will probably need to make a list of items at some point, believe me as you get older your memory fails and you need lists.</p>
<p>Create Class ItemList</p>	<p>4) Right click on the package you created: 5) Choose File → New → class 6) Give it a name of ItemList and choose the Finish button</p> <pre>package com.cosc210;  public class ItemList {  }</pre>
<p>What might a class called ItemList need as an attribute(s)?</p> <p>We want an actual list of items, what have we learned about that will allow us to store a group of similar things?</p> <p>An array</p> <p>In our code to the right we have created an empty array of Items, we will do stuff with it in future updates in this exercise.</p>	<p>1) Add an array to you ItemList class to store Items</p> <pre>package com.cosc210;  public class ItemList {     Item items[] = null; }</pre>
<p>What might a class called ItemList need as behaviors?</p> <p>Maybe:</p> <ul style="list-style-type: none"> <li>addItem</li> <li>deleteItem</li> <li>createList</li> <li>displayItems</li> </ul> <p>For adding an item we will need have the Item object and its index into the array: no return value</p> <p>For deleting an item we will need only the index of the array to set that element of the array to null, if we wanted to check and compare the item in the array to some item object we thought we wanted to delete, we might also pass in an item object as a parameter: no return value</p>	<p><b>DON'T TYPE JUST READ THIS SECTION</b></p> <p>What might the parameters and return types be for these 4 methods if we were going to create them?</p> <p>Remember: all we have in the ItemList class so far is a single array that can store Item objects that currently is set to null</p> <pre>public void addItem(Item itemIn, int index)     implementation comes later  public void deleteItem(Item itemIn, int index)     implementation comes later</pre>

<p>We have an array, we want to display what is in the array, the parameter would be the array of which we want to display its contents: no return type we will simply print its contents to standard out</p> <p><b>If we are creating something we usually need to return it, so the return type would be an array of Item objects. Since we know it is an array and arrays need a size when they are initialized we pass an integer parameter representing the size of the array</b></p>	<pre>public void displayItems(Item list[])     implementation comes later</pre> <pre>public Item[] createList(int listSize)     implementation comes later</pre>
<p>Create the addItem method</p> <p>As we said above the addItem method has an Item and an int as parameters. All we have to do much like a setter method is take the input parameters and use them to add the item into the array of items.</p>	<pre>package com.cosc210;  public class ItemList {     Item items[] = null;      public void addItem(Item itemIn, int index){         if (items[index].equals(null)){             System.out.println(                 "adding Item to an empty slot in the array");             items[index] = itemIn;         } else {             System.out.println(                 "replacing item " + items[index].getName());             System.out.println(                 "with " + itemIn.getName());              items[index] = itemIn;         }     } }</pre>

<p>Create the deleteItem method</p> <p>As we said above the deleteItem method has an Item and an int as parameters. All we have to do much like a getter method is take the input parameters and use them to delete the item from the array of items.</p> <p>For deletes, you usually want to make sure what you are deleting is actually what you think it is, so we will put a check in to make sure the item at the index passed in has the same name as the item passed in as a parameter</p>	<pre>package com.cosc210; public class ItemList {     Item items[] = null;      public void addItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}     }      public void deleteItem(Item itemIn, int index){         if             (items[index].                 getName().                     equals(itemIn.getName())){             System.out.println(items[index].getName() +                 " is deleted from the list");             items[index] = null;         } else {             System.out.println(items[index].getName() +                 " is in the list at the supplied index");             System.out.println(itemIn.getName() +                 " does not match what is in the list");              System.out.println("nothing deleted");         }     } }</pre>
<p>Create the displayItems method</p> <p>We know we have an array of Items, we will iterate( or loop one at a time) through the array printing the name of each item</p> <p>We can get the size of the array to use in a loop → We can get the name of each array element to print →</p>	<pre>package com.cosc210; public class ItemList {     Item items[] = null;     public void addItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}     }      public void deleteItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}     }      public void displayItems(Item list[]){         int sizeOfList = list.length;         for (int n=0;n&lt;sizeOfList;n++){             System.out.println("Item is " +                 list[n].getName());         }     } }</pre>

<p>Create the createList method</p> <p>We can add an item We can delete an item We can display the item Names</p> <p>This method although we implement it last actually creates the list</p> <ol style="list-style-type: none"> <li>1) Instantiates an array of Items</li> <li>2) Instantiates Scanner objects to collect information about each item</li> <li>3) Loops using the parameter passed in to decide how many items will be in the list, thus how many times to loop</li> </ol> <p>Notice we used the setters from Item to do this</p> <ol style="list-style-type: none"> <li>4) Returns the array of Item objects we just built</li> </ol>	<pre>package com.cosc210; public class ItemList {     Item items[] = null;     public void addItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}}     }      public void deleteItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}}     }      public void displayItems(Item list[]){         {{{{{{no changes to this method}}}}}}     }      public Item[] createList(int listSize) {         Item currentItems[] = new Item[listSize];          Scanner inputName = new Scanner(System.in);         Scanner inputCost = new Scanner(System.in);         Scanner inputQuantity = new Scanner(System.in);          for (int x=0;x&lt;listSize;x++){             currentItems[x] = new Item();             System.out.println("Enter an item name: ");              currentItems[x].setName(inputName.next());             System.out.println("Enter item cost: ");              currentItems[x].setCost(inputCost.nextDouble());             System.out.println("Enter quantity of this item: ");              currentItems[x].setQuantity(inputQuantity.nextInt());         }         return currentItems;     } }</pre>
<p>What might we want to do when an object of type ItemList gets instantiated, gets created in some test program somewhere with a main method?</p> <p>Right now there is no constructor for ItemList.</p> <ol style="list-style-type: none"> <li>1) create a default constructor that does nothing</li> </ol>	<pre>package com.cosc210; import java.util.Scanner; public class ItemList {     Item items[] = null;     public ItemList(){     }     public void addItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}}     }     public void deleteItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}}     }     public void displayItems(Item list[]){         {{{{{{no changes to this method}}}}}}     }     public Item[] createList(int listSize){         {{{{{{no changes to this method}}}}}}     } }</pre>

<p>We might want to execute the createList method when an object of type ItemList is instantiated.</p> <p>Since we have created a list of Items lets also display the list at creation or instatiation time.</p> <p>How do we do this? One way is too:</p> <ol style="list-style-type: none"><li>1) By passing an integer to the ItemList objects constructor to use as the size of the List.</li><li>2) Create a constructor that takes an integer as a parameter</li><li>3) Invoke the createList method using that integer</li><li>4) Invoke the displayList method using the returned array from createList</li></ol>	<pre>package com.cosc210; import java.util.Scanner; public class ItemList {     Item items[] = null;      public ItemList(){     }      public ItemList(int numberOfItems){         items = this.createList(numberOfItems);         this.displayItems(items);     }      public void addItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}}     }      public void deleteItem(Item itemIn, int index){         {{{{{{no changes to this method}}}}}}     }      public void displayItems(Item list[]){         {{{{{{no changes to this method}}}}}}     }      public Item[] createList(int listSize){         {{{{{{no changes to this method}}}}}}     } }</pre>
<p>We have created enough code to test. Let us go shopping by creating a java application called GoShopping.</p> <ol style="list-style-type: none"><li>1) Create the GoShopping class in package com.cosc210</li><li>2) Check the box for public static void main(String args[]) in eclipse</li></ol>	<p>The initial code for GoShopping is created like this:</p> <pre>package com.cosc210; public class GoShopping {     public static void main(String[] args) {     } }</pre>

For this exercise we want to create a list and display the items.

We made the constructor for ItemList we know it creates a list and displays it .

We also know it takes an integer representing the number of items in the list.

As a result we should:

- 1) prompt the user to enter the number of items and
- 2) instantiate an ItemList object using that number of items.

The ItemList object will instantiate an array of Items and prompt us to enter the attributes of each item than display them in standard out(or the console)

```
package com.cosc210;
import java.util.Scanner;

public class GoShopping {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println(
            "Enter how many items you will be buying today: ");
        int numberOfItems = input.nextInt();

        ItemList myList = new ItemList(numberOfItems);
    }
}
```