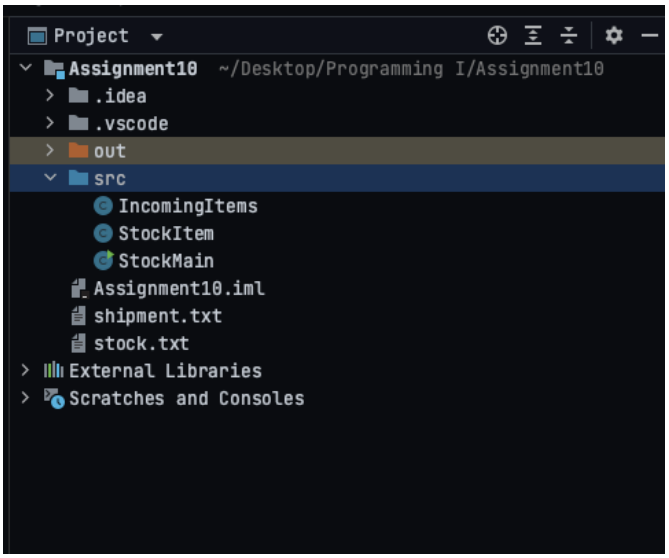
System structure



New Class: IncomingItems

```
import java.io.FileWriter;
import java.util.Scanner;

public class IncomingItems {

    private String name;
    private int quantity;

    public IncomingItems() {
    }

    public IncomingItems(String name, int quantity) {
        this.name = name;
        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public void Serialize(FileWriter fw){
        try {
            fw.write(name);
            fw.write("\n");
            fw.write(Integer.toString(quantity));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static IncomingItems Deserialize(Scanner fileInput){
        String n = fileInput.nextLine();
        int q = Integer.parseInt(fileInput.nextLine());
        return new IncomingItems(n, q);

    }
}
```

Function Implement:

5) Add New Stock Item

- Code snippet

```
private static void AddNewStockItem() {
    Scanner input = new Scanner(System.in);
    System.out.printf(" Name of New Stock Item > ");
    String name = input.nextLine();

    System.out.print(" Cost Price > ");
    float costPrice = input.nextFloat();
    System.out.printf(" Sale Price > ");
    float salePrice = input.nextFloat();
    StockItem newItem = new StockItem(name, 0, costPrice,
salePrice);
    stockItems.add(newItem);
    //print out new item information
    System.out.println( name + " added to StockItems. We now have
" + stockItems.size() + " items.");

}
```

- Output

```
*** STOCK MANAGEMENT PROGRAM ***
Read 4 stock items from file stock.txt.

1) List Stock Items
2) Add Stock
3) Remove Stock
4) Save File
5) Add New Stock Item
6) Delete a Stock Item
7) List Low Stock Items
8) Record Incoming Stock Shipment
9) Calculate Cost of All Stock
10) Exit
Command > 5
 Name of New Stock Item > Green Mars
 Cost Price > 9.5
 Sale Price > 20
Green Mars added to StockItems. We now have 5 items.


1) List Stock Items
2) Add Stock
3) Remove Stock
4) Save File
5) Add New Stock Item
6) Delete a Stock Item
7) List Low Stock Items
8) Record Incoming Stock Shipment
9) Calculate Cost of All Stock
10) Exit
Command > 1

0> Name:              Harry Potter, Stock:   15, Cost:  8.00, Sale: 15.99
1> Name:     The Lord of the Rings, Stock:    4, Cost: 12.00, Sale: 25.99
2> Name:                  Red Mars, Stock:    3, Cost:  8.00, Sale: 18.99
3> Name:       The Call of Cthulhu, Stock:    4, Cost:  3.00, Sale:  9.99
4> Name:                Green Mars, Stock:    0, Cost:  9.50, Sale: 20.00
```

6) Delete a Stock Item

- Code snippet

```
private static void DeleteStockItem() {
    Scanner input = new Scanner(System.in);
    System.out.printf("\nDelete which Stock Item number? (0-%d) >
", stockItems.size() - 1);
    int idx = input.nextInt();

    System.out.println("The " + stockItems.get(idx).getName() + "
has been deleted.");
    if (idx >= 0 && idx < stockItems.size()) {
        stockItems.remove(idx);
    }
    System.out.println("We now have " + stockItems.size() + "
items.");
}
```

- Output

```
Command > 6

Delete which Stock Item number? (0-4) > 1
The The Lord of the Rings has been deleted.
We now have 4 items.



1) List Stock Items
2) Add Stock
3) Remove Stock
4) Save File
5) Add New Stock Item
6) Delete a Stock Item
7) List Low Stock Items
8) Record Incoming Stock Shipment
9) Calculate Cost of All Stock
10) Exit
Command > 1

0> Name:              Harry Potter, Stock:   15, Cost:  8.00, Sale: 15.99
1> Name:                  Red Mars, Stock:    3, Cost:  8.00, Sale: 18.99
2> Name:       The Call of Cthulhu, Stock:    4, Cost:  3.00, Sale:  9.99
3> Name:                Green Mars, Stock:    0, Cost:  9.50, Sale: 20.00
```

7) List Low Stock Items

- Code snippet

```
private static void ListLowStockItems() {
    System.out.printf("We have less than 5 of the following
items:");
    for (int i = 0; i < stockItems.size(); i++) {
        int stockLevel = stockItems.get(i).getStockLevel();
        if (stockLevel <= 5) {
            stockItems.get(i).Print(i);
        }
    }
}
```

- Output

```
Command > 1

0> Name:              Harry Potter, Stock:   15, Cost:  8.00, Sale: 15.99
1> Name:                  Red Mars, Stock:    3, Cost:  8.00, Sale: 18.99
2> Name:       The Call of Cthulhu, Stock:    4, Cost:  3.00, Sale:  9.99
3> Name:                Green Mars, Stock:    0, Cost:  9.50, Sale: 20.00

1) List Stock Items
2) Add Stock
3) Remove Stock
4) Save File
5) Add New Stock Item
6) Delete a Stock Item
7) List Low Stock Items
8) Record Incoming Stock Shipment
9) Calculate Cost of All Stock
10) Exit
Command > 7
We have less than 5 of the following items:
1> Name:                  Red Mars, Stock:    3, Cost:  8.00, Sale: 18.99
2> Name:       The Call of Cthulhu, Stock:    4, Cost:  3.00, Sale:  9.99
3> Name:                Green Mars, Stock:    0, Cost:  9.50, Sale: 20.00
```

8) Record Incoming Stock Shipment

- Code snippet

```
private static boolean RecordIncomingStockShipment() {
    int recordItemQty = 0;
    try (Scanner input = new Scanner(Paths.get(incomingItems))) {

        while (input.hasNext()) {
            //read name and quantity from file: incomingItems
            String itemName = input.nextLine();
            int itemQuantity = Integer.parseInt(input.nextLine());


            // Find the stock item with the given name
            boolean found = false;
            for (int i = 0; i < stockItems.size(); i++) {
                if (stockItems.get(i).getName().equals(itemName)) {
                    // Update the stock level of the found item
                    stockItems.get(i).ChangeStockLevel(itemQuantity);
                    recordItemQty++;
                    found = true;
                    break;
                }
                //print warning if item not found, just print one time
                if (i == stockItems.size() - 1) {
                    System.out.println("ERROR! No StockItem found with the
name:" + itemName);
                }
            }
        }
        System.out.println("Recorded " + recordItemQty + " items in shipment
file");
        input.close();
    } catch (IOException e) {
        System.out.printf("\nCannot load file %s\n", incomingItems);
        return false;
    }

    return true;

}
```

- Output

```
10) Exit
Command > 8
ERROR! No StockItem found with the name:The Lord of the Rings
Recorded 2 items in shipment file


1) List Stock Items
2) Add Stock
3) Remove Stock
4) Save File
5) Add New Stock Item
6) Delete a Stock Item
7) List Low Stock Items
8) Record Incoming Stock Shipment
9) Calculate Cost of All Stock
10) Exit
Command > 1

0> Name:             Harry Potter, Stock:   40, Cost:  8.00, Sale: 15.99
1> Name:                 Red Mars, Stock:   13, Cost:  8.00, Sale: 18.99
2> Name:       The Call of Cthulhu, Stock:    4, Cost:  3.00, Sale:  9.99
3> Name:               Green Mars, Stock:    0, Cost:  9.50, Sale: 20.00
```

9) Calculate Cost of All Stock

- Code snippet

```
private static void CalculateCostofAllStock() {
    float totalCost = 0;
    int totalStock = 0;
    for (int i = 0; i < stockItems.size(); i++) {
        totalCost += stockItems.get(i).getStockLevel() *
stockItems.get(i).getCostPrice();
        totalStock += stockItems.get(i).getStockLevel();
    }
    //Our total stock is 57 and total cost is 436.00.
    System.out.printf("Our total stock is %d and total cost
is %.2f.", totalStock, totalCost);
}
```

- Output