**List of bugs at the beginning of phase 2**

* IOE exception message is returned for both customers and orders when opening application for the first time (should be changed to a message that tells a user there was no file created yet).
* “Object has been serialized” message is returned when save button is clicked without specifying whether customers and orders were successfully saved.
* An error massage is returned when add to cart button is clicked without selecting anything.
* An error massage is returned when add to cart button is clicked without selecting one of the options (size, quantity and color (order details)). At the same time, those details that were selected are also displayed to the user.
* The order selection window comes before customer selection window (design).
* Null orders are allowed.
* Null customers are allowed.
* Creating an order without a customer is allowed.
* An error message is returned when customer info button is clicked and unfilled order with a null customer is selected.
* An error message is returned when customer info button is clicked and filled order is selected (does not matter if customer is null or not).
* When a new unfilled order is placed, the first order in the filled orders gets duplicated.
* When saving to file, only customers.txt file is created (no orders.txt is present).

**Fix all bugs at the beginning of phase 2**

1. Create the DataHolder.class to save data.

Source code cited: https://docs.oracle.com/cd/E13211\_01/wle/javadoc/m3java/com/beasys/rmi/iiop/DataHolder.html)

* Set value to each field.
* Constructor Detail
* Get info of Customer and Order by ID
* Set string for new order and new customer
* Create two serialized files “Customer” & “Order”
* InitData Function: Read the data first, then check the biggest ID. It is helped add new customer and new order.
* Read-Write data: Saving data of customerlist and orderlist in dataholder.
* Calling get methods of DataHolder in other class.

1. Modify the Customer and Order Class

* Add StringBuilder Method. [StringBuilder](https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html)objects are like [String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html)objects, except that they can be modified. Internally, these objects are treated like variable-length arrays that contain a sequence of characters. At any point, the length and content of the sequence can be changed through method invocations. By StringBuilder Source code cited:

<https://docs.oracle.com/javase/tutorial/java/data/buffers.html>

1. Add Alert pop up window in three controllers (AddNewCustomerController, NewOrderDetailsController, OrdersController)

* The error message will be returned when any filed is not filled out.
* By Alert Popup Window Source code cited:

<http://code.makery.ch/blog/javafx-dialogs-official/>

1. Add .clear() method into unfilled to fix the bug of order duplication.

**Project Breakdown/Architecture**

The main controller of the application is the OrdersController. All the features of the application can be accessed only through this window. All the data of the application is loaded through this controller using serialization/deserialization. The data of this application is stored in 6 main files which store the following:

* Orders (Filled and Unfilled – identified by dateFilled values) (ArrayList)
* Customers (ArrayList)
* Current Inventory (HashMap)
* Scheduled Inventory (ArrayList)
* Most/Least Popular Items (Integer Array)
* Backorder Items (Integer Array)

All of the following files store data that uses 3 major data structures within the application, namely – HashMaps, ArrayLists of the type – instance of a class, and basic Integer Arrays.

There are a total of 5 classes:

* Customer – stores customer information
* Order – stores order information
* Thneed – stores details of each item in the order
* Inventory – stores the scheduled inventory information
* DataHolder – Helps in reading and writing customer/order files

There are a total of 6 controllers:

* OrdersController (Main Page)
* NewOrderDetailsController (Place an order, Add a new customer)
* AddNewCustomerController (Add a new customer)
* CustomerInfoController (View information of the customer who placed the selected order in the list)
* InventoryOrderController (Restock items, View current inventory, and View report)
* ReportController (Displays order statistics based on the order history)

**Daily Work Progress**

11/28

DONE

- Fix all bugs in Phase 1

- Work on extension class in Phase 2

11/30

DONE

- Discuss the GUI for Inventory.

- Modify the other UI frames.

12/01

DONE

- Create the inventory class

- Connect each windows

12/05

DONE

- Continue working on the inventory class

12/07

DONE

- Continue working on how to display the table in the inventory

12/08

DONE

- Fix Bugs

- Work on the report.

**Unresolved Issues/Bugs**

* The report GUI is not getting loaded and not displaying the statistics. Order fill rate is not calculated
* The scheduled inventory is not getting updated in the initialize function in OrdersController.