**Testing Plan/Units**

**Final testing**

Each method of every controller was tested by inserting different types of values and observing the results.

**Controller:** AddNewCustomerContoller

**Methods tested:**

* public void cancelButtonClicked(ActionEvent event) – closes the window when user clicks on “close” button (takes no input and works on click)
* public void addCustomerButtonClicked(ActionEvent event) – creates a new customer object and adds it into the array list of customers on “add customer” button click (takes following inputs: String, String, String; works on click; cannot take null inputs; if any of the inputs entered by user are null, composes a corresponding text message)
* private void showAlert(String header, String message) – displays the corresponding error message to the user in a new window if null values were entered (takes following inputs: String, String; works when null inputs are entered by the user and “add customer” button is clicked)
* public void setCallingController(NewOrderDetailsController newOrderController) – sets a different control to refer to (takes controller name as an input)
* public void setCustomerIDLabel(String input) – setter method (takes string as an input)

**Controller:** CustomerInfoContoller

**Methods tested:**

* public void setCustomerName(String name) – setter method (takes string as an input)
* public void setCustomerAdd(String add) – setter method (takes string as an input)
* public void setCustomerPhone(String phone) – setter method (takes string as an input)
* public void closeWindow(ActionEvent event) - closes the window when user clicks on “close” button (takes no input and works on click)

**Controller:** InventoryOrderController

**Methods tested:**

* public ArrayList<Inventory> getTotalInvent() – getter method (takes no arguments and returns an array list)
* public void setCallingController (OrdersController ordersController) – setter method (sets a calling controller and takes controller’s name as an argument)
* public void initialize() – sets values in observable lists for colors and sizes, and sets defaults for each of them (takes no arguments)
* public void setInvLabels() – setter method (sets inventory labels equal to a corresponding quantity taken from the hash map; takes no arguments)
* public void reportButtonClicked(ActionEvent event) – opens a Report.fxml window when user clicks on “report” button (takes no inputs and works on click)
* public void reStockButtonClicked() – adds a number of Thneed of certain color and size to the type of Thneed specified by user when “restock” button is clicked and saves the current date (requires following inputs to be specified by user: Integer, Integer, String; works on click; cannot take null inputs; if any of the inputs entered by user are null, composes a corresponding text message)
* public void doneButtonClicked(ActionEvent event) – saves the changes made to the inventory when user clicks on “done” button and closes the window (uses serialization to create currInv.ser and schedInv.ser files)
* public HashMap<Integer, Integer> getInventoryData() – getter method (takes no arguments and returns a hash map)
* public void setInventoryData(HashMap<Integer, Integer> inventoryData) – setter method (takes a hash map as an argument and updates inventory data)

**Controller:** NewOrderDetailsController

**Methods tested:**

* public void setCustomerIDCombo() – setter method (takes no arguments and sets customer ID in the observable list)
* public void setCallingController(OrdersController orderController) - setter method (sets a calling controller and takes controller’s name as an argument)
* public void initialize() - sets values in observable lists for colors and sizes, and sets defaults for each of them (takes no arguments)
* public void setOrderIDLabel(String input) – setter method (takes a String as an argument and sets OrderId Label)
* public void newCustomerButtonClicked(ActionEvent event) - Opens the newCustomerController window when user clicks on “New Customer” button (takes no inputs and works on click)
* public void placeOrderButtonClicked(ActionEvent event) - adds user input data into an arrayList called <Order> when user clicks “Place Order” button (requires user to enter following inputs: array list of theeds (consists from color, quantity and size inputs), customer ID; works on click; cannot take null inputs; if any of the inputs entered by user are null, composes a corresponding text message)
* public void addToCartClicked (ActionEvent event) - adds user input data into an arrayList called <Thneed> when user clicks on “Add to Cart” button (requires user to enter following inputs: color (String), quantity (Integer), size (String) and customer ID; works on click; cannot take null inputs; if any of the inputs entered by user are null, composes a corresponding text message)
* public void cancelButtonClicked(ActionEvent event) - closes the window when user clicks on “close” button (takes no input and works on click)
* private void showAlert(String header, String message) – displays the corresponding error message to the user in a new window if null values were entered (takes following inputs: String, String; works when null inputs are entered by the user and “Add to Cart” or “Place Order” button is clicked)

**Controller:** OrdersController

**Methods tested:**

* public void initialize() – initializes inventory, connects Order and Inventory classes and updates inventory based on the Dates specified by user (takes no arguments)
* public void saveBackorder() – saves the items in backorder (takes no arguments and uses serialization to create backorder.ser file)
* public void savePopular() – saves items that are popularized (takes no arguments and uses serialization to create popular.ser file)
* public ArrayList<Inventory> getSchedData() – getter method (used deserialization to open schedInv.ser and returns and inventory array list)
* public void customerInfoClicked(ActionEvent event) – opens customerInfoController window and displays customer information when user selects and order from filled or unfilled orders box and clicks on “Customer Info” button (works on click; requires user to select an order; if order is not selected composes a corresponding text message)
* public void inventoryClicked(ActionEvent event) - opens InventoryOrderController window when user clicks on “Inventory” button (takes no input and works on click)
* public void updateButtonClicked() – fills the selected order from unfilled orders box and displays it in the filled orders box when user clicks “Update” button (works on click; requires user to select an order; if order is not selected composes a corresponding text message)
* public void populateUnfilledOrders() – populates the unfilled orders box (takes no arguments)
* public void saveDataCalled(ActionEvent event) – saves orders and customers data to data.txt when user clicks on “Save” button (takes no inputs and works on clicks)
* public void newOrderButtonClicked(ActionEvent event) - opens NewOrderDetailsController when user clicks on “New Order” button (takes no input and works on click)
* private void showAlert(String header, String message) - displays the corresponding error message to the user in a new window if null values were entered (takes following inputs: String, String; works when no order is selected and “Update” or “Customer Info” button is clicked)
* public HashMap<Integer, Integer> getInventoryData() – getter method (takes no arguments and returns a hash map)
* public void setInventoryData(HashMap<Integer, Integer> inventoryData) – setter method (takes a hash map as an argument and updates inventory data)

**Controller:** ReportController

**Methods tested:**

* public void setCallingController(InventoryOrderController inventoryController) - setter method (sets a calling controller and takes controller’s name as an argument)