The System’s Java Code

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.*;
import java.text.SimpleDateFormat;

public class inventory extends JFrame {
    // Instance variables
    private Dessert[] collection; // Array to store Dessert objects
    private int itemCount; // Counter for the number of Desserts in
the collection
    private JTextField nameField, quantityField, priceField; //
Input fields for Dessert details
    private JFormattedTextField dateAddedField; // Input field for
the date added
    private JTable itemTable; // Table to display Dessert
information
    private DefaultTableModel tableModel; // Table model for the
JTable
    private JComboBox<String> sortComboBox; // ComboBox for sorting
options
    private SimpleDateFormat dateFormat; // Date format for parsing
and displaying dates

    public inventory() {
        // Initialize the JFrame
        collection = new Dessert[100]; // Initialize the Dessert
collection array
        itemCount = 0; // Set the initial item count to zero

        setTitle("Cafe Inventory");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(800, 1000);
        setLayout(new BorderLayout());
        getContentPane().setBackground(new Color(200, 162, 200)); //
Set the background color

        dateFormat = new SimpleDateFormat("yyyy/MM/dd"); //
Initialize the date format

        // Create and add the input panel to the JFrame
        JPanel inputPanel = createInputPanel();
        JPanel sortPanel = createSortPanel();
        JPanel itemTablePanel = createItemTablePanel();

        add(inputPanel, BorderLayout.NORTH);
        add(sortPanel, BorderLayout.CENTER);
        add(itemTablePanel, BorderLayout.SOUTH);
    }

    // Create the input panel
    private JPanel createInputPanel() {
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new GridLayout(7,2,10,10)); // Grid
layout with 7 rows and 2 columns
        inputPanel.setBackground(new Color(200, 162, 200)); // Set
the background color

        // Create and initialize GUI components
        JLabel title = new JLabel("X Cafe Inventory");
        title.setFont(new Font("Times New Roman", Font.BOLD, 30));
        JLabel empty = new JLabel("");
        JLabel nameLabel = new JLabel(" Product Name:");
        JLabel quantityLabel = new JLabel(" Quantity:");
        JLabel priceLabel = new JLabel(" Price:");
        JLabel dateAddedLabel = new JLabel(" Date Added:
YYYY/MM/DD");

        JButton addButton = new JButton("Add");
        addButton.addActionListener(new AddButtonListener());

        JButton removeButton = new JButton("Remove");
        removeButton.addActionListener(new RemoveButtonListener());

        nameField = new JTextField(20);
        quantityField = new JTextField(20);
        priceField = new JTextField(20);
        dateAddedField = new JFormattedTextField(dateFormat);
        String[] sortOptions = { "Product Name", "Price", "Date
Added" };
        sortComboBox = new JComboBox<>(sortOptions);

        // Add components to the input panel
        inputPanel.add(title, BorderLayout.LINE_START);
        inputPanel.add(empty);
        inputPanel.add(nameLabel);
        inputPanel.add(nameField);
        inputPanel.add(quantityLabel);
        inputPanel.add(quantityField);
        inputPanel.add(priceLabel);
        inputPanel.add(priceField);
        inputPanel.add(dateAddedLabel);
        inputPanel.add(dateAddedField);

        inputPanel.add(addButton);
        inputPanel.add(removeButton);

        return inputPanel;
    }

    // Create the sort panel
    private JPanel createSortPanel() {
        JPanel sortPanel = new JPanel();
        sortPanel.setBackground(new Color(200, 162, 200)); // Set
the background color
```

```
        // Create and initialize GUI components
        JLabel sortLabel = new JLabel("Sort by:");

        JButton sortButton = new JButton("Sort");
        sortButton.addActionListener(new SortButtonListener());

        JButton viewButton = new JButton("View");
        viewButton.addActionListener(new ViewButtonListener());

        // Add components to the sort panel
        sortPanel.add(sortLabel);
        sortPanel.add(sortComboBox);
        sortPanel.add(sortButton);
        sortPanel.add(viewButton);

        return sortPanel;
    }

    // Create the item table panel
    private JPanel createItemTablePanel() {
        JPanel itemTablePanel = new JPanel();
        itemTablePanel.setLayout(new BorderLayout()); // Use
BorderLayout for the item table panel
        itemTablePanel.setBackground(new Color(200, 162, 200)); //
Set the background color

        tableModel = new DefaultTableModel();
        tableModel.addColumn("Product Name"); // Add columns to the
table model
        tableModel.addColumn("Quantity");
        tableModel.addColumn("Price");
        tableModel.addColumn("Date Added");

        itemTable = new JTable(tableModel); // Create a JTable with
the table model

        JScrollPane scrollPane = new JScrollPane(itemTable); // Add
a scroll pane to the table

        itemTablePanel.add(scrollPane); // Add the scroll pane to
the item table panel

        return itemTablePanel;
    }

    // Refresh the item table with the updated Dessert collection
    private void refreshItemTable() {
        tableModel.setRowCount(0); // Clear the table model

        for (int i = 0; i < itemCount; i++) {
            Dessert dessert = collection[i];
            String dessertDetails = dessert.getDessertDetails();
            tableModel.addRow(dessertDetails.split(" - ")); // Add a
row to the table model
```

```
        }
    }

    // Inner class representing a Dessert
    private class Dessert {
        // Instance variables for Dessert
        private String name;
        private int quantity;
        private double price;
        private Date dateAdded;

        public Dessert(String name, int quantity, double price, Date 
dateAdded) {
            this.name = name;
            this.quantity = quantity;
            this.price = price;
            this.dateAdded = dateAdded;
        }

        // Getters for Dessert properties
        public String getName() {
            return name;
        }

        public int getQuantity() {
            return quantity;
        }

        public double getPrice() {
            return price;
        }

        public Date getDateAdded() {
            return dateAdded;
        }

        // Get the details of the Dessert as a formatted string
        public String getDessertDetails(){
            return getName() + " - " + getQuantity() + " - RM " + 
getPrice() + " - " + dateFormat.format(getDateAdded());
        }
    }

    // ActionListener for the Add button
    private class AddButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Get the input values from the text fields
            String name = nameField.getText();
            int quantity = 0;
            double price = 0.0;
            Date dateAdded = null;

            try {
                quantity =
Integer.parseInt(quantityField.getText());
                price = Double.parseDouble(priceField.getText());
                dateAdded =
dateFormat.parse(dateAddedField.getText());
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(inventory.this,
"Invalid input for quantity or price.");
                return;
            } catch (java.text.ParseException ex) {
                JOptionPane.showMessageDialog(inventory.this,
"Invalid date format. Please use the format: YYYY/MM/DD");
                return;
            }

            // Create a new Dessert object and add it to the
collection
            Dessert dessert = new Dessert(name, quantity, price,
dateAdded);
            collection[itemCount++] = dessert;

            // Clear the input fields
            nameField.setText("");
            quantityField.setText("");
            priceField.setText("");
            dateAddedField.setValue(null);

            // Display a confirmation message and update the item
table
            JOptionPane.showMessageDialog(inventory.this, "Dessert
added to the collection.");
            tableModel.addRow(new Object[]{dessert.getName(),
dessert.getQuantity(), "RM " +dessert.getPrice(),
dateFormat.format(dessert.getDateAdded())});
        }
    }

    // ActionListener for the Remove button
    private class RemoveButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            int selectedRowIndex = itemTable.getSelectedRow();
            if (selectedRowIndex >= 0 && selectedRowIndex <
itemCount) {
                // Remove the selected Dessert from the collection
and update the table
                for (int i = selectedRowIndex; i < itemCount - 1;
i++) {
                    collection[i] = collection[i + 1];
                }
                collection[itemCount - 1] = null;
                itemCount--;
                tableModel.removeRow(selectedRowIndex);
                JOptionPane.showMessageDialog(inventory.this,
"Dessert removed from the collection.");
```

```
            } else {
                JOptionPane.showMessageDialog(inventory.this, "No 
dessert selected or dessert not found in the collection.");
            }
        }
    }

    // ActionListener for the Sort button
    private class SortButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String selectedProperty = (String) 
sortComboBox.getSelectedItem();

            // Sort the collection based on the selected property
            switch (selectedProperty) {
                case "Product Name":
                    Arrays.sort(collection, 0, itemCount, 
Comparator.comparing(Dessert::getName));
                    break;
                case "Price":
                    Arrays.sort(collection, 0, itemCount, 
Comparator.comparingDouble(Dessert::getPrice));
                    break;
                case "Date Added":
                    Arrays.sort(collection, 0, itemCount, 
Comparator.comparing(Dessert::getDateAdded));
                    break;
                default:
                    break;
            }

            // Refresh the item table and display a confirmation 
message
            refreshItemTable();
            JOptionPane.showMessageDialog(inventory.this, "Items 
sorted by " + selectedProperty);
        }
    }

    // ActionListener for the View button
    private class ViewButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Create a new JFrame to display the item table
            JFrame tableFrame = new JFrame("View Inventory");
            tableFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CL
OSE);
            tableFrame.setSize(800, 600);

            JTable tableView = new JTable(tableModel); // Create a 
JTable with the table model
            JScrollPane scrollPane = new JScrollPane(tableView); // 
Add a scroll pane to the table
```