This program is designed to simulate an order-placing and appointment scheduling system for a restaurant. The goal is to let the users choose different food items from a simple menu, specify how much they want, and then schedule a delivery appointment by entering a date and time. The program uses a swing interface, which makes it easy to understand and gives them a visual so it is less confusing.Its target audience is introductory Java students and even small businesses who need online ordering systems.

The program demonstrates the fundamentals of object-oriented programming such as classes, objects, inheritance, and interfaces. It also uses arrays, loops, and a file output to write the order/appointment summary to a text file. One of its strengths is that it is simple and straightforward to the user and the code is also easy to follow.The lack of colors and images in the swing interface is a weakness as it makes it less appealing .Improvements could include further improvements of the UI layout, incorporation of more features like menu images, and better exception handling. Overall, it is a solid foundation representing what we covered in class and a bit of my own knowledge.

Strengths:

- Easy to use graphical interface
- demonstration of Java Swing integration
- Maintains solid object-oriented design and structure

Weaknesses:

- Increased complexity in error handling
- Swing dialogs can sometimes be less flexible than a custom GUI

Future Improvements:

- Custom GUI design with panels and buttons instead of just boxes
- Improved error handling and validation for inputs such as dates and hours
- More features such as order modification and setting multiple appointments

Pseudo Code

```
START PROGRAM

 // Set up our basic variables and data structures.
 Create an array called "orders" that can hold up to 10 orders.
 Set a variable "orderCount" to 0 (this will track how many orders have been added).
 Create an array "menuItems" that holds the items: Burger, Pizza, Salad, Pasta.
 Create a corresponding array "menuPrices" with the prices for each item.
 Set a boolean variable "ordering" to true (this will control our loop).

 WHILE (ordering is true AND orderCount is less than 10) DO:
    // Build a menu string to show all options.
    Create a string "menu" that lists each menu item with its number and price.
    Also add an option labeled "5" to finish ordering and set the appointment.

    // Show the menu to the user using a Swing input dialog.
    Display the "menu" string in a dialog and ask the user to enter their choice.

    IF the user cancels the input THEN:
       Set ordering to false (we're done taking orders) and exit the loop.
    ENDIF

    TRY to convert the user’s input into an integer called "choice".
    IF the conversion fails (meaning the input wasn’t a number) THEN:
       Display an error dialog saying "Invalid input, please enter a number."
       Continue to the next loop iteration.
    ENDIF

    IF "choice" is 1, 2, 3, or 4 THEN:
       Compute the index as (choice - 1).
       Ask the user (via another Swing input dialog) for the quantity of the selected item.
       TRY to convert this quantity input to an integer.

       IF the conversion fails OR the quantity is less than or equal to 0 THEN:
          Display an error dialog: "Invalid quantity. Please enter a positive number."
          Skip the rest of this loop iteration (do not add an order).
       ELSE:
            Create a new Order object using:
              - The menu item from "menuItems" at the computed index,
              - The corresponding price from "menuPrices",
              - And the quantity entered by the user.
            Add this new Order object to the "orders" array.
            Increase "orderCount" by 1.
            Display a confirmation dialog that the item was successfully added.
          ENDIF

        ELSE IF "choice" equals 5 THEN:
          Set ordering to false (user is finished ordering) and exit the loop.

        ELSE:
          Display an error dialog stating "Invalid option. Please select a valid menu option."
        ENDIF

      END WHILE

      IF orderCount is 0 THEN:
        Display a dialog "No orders placed." and TERMINATE the program.
      ENDIF

      // Now, get the appointment details.
      Use a Swing input dialog to ask the user for the appointment date (format: dd/mm/yyyy).
      Use another Swing input dialog to ask for the appointment time (format: HH:mm, 24-hour
    clock).
      Create an Appointment object with the date and time provided.

      // Build the order summary.
      Initialize a string "summary" with a header like "Order Summary:".
      Set a variable "grandTotal" to 0.

      FOR each order from index 0 to orderCount - 1 in the "orders" array DO:
        Append the details of the current order (item, quantity, price per item, total cost) to
    "summary".
        Add the total cost of this order to "grandTotal".
      END FOR

      Append the grand total and the appointment details to "summary".

      // Display the summary in a nice format.
      Create a non-editable text area containing the "summary" string.
      Place the text area in a scroll pane (with a fixed preferred size).
      Show a Swing dialog that displays this scroll pane so the user can review their order.
```

```
  // Write the summary to a text file.
  TRY to open a file (for example, "order_summary.txt") for writing.
      Write the "summary" string into the file.
      Close the file.
      Display a confirmation dialog that the file has been saved.
  CATCH any errors during file writing and display an error dialog if needed.

END PROGRAM
```