

# **COMPUTER SHOP MANAGEMENT SYSTEM IN JAVA**

# GROUP MEMBER

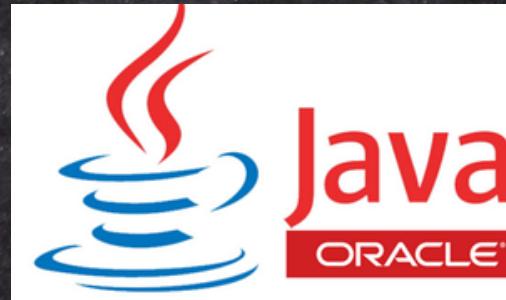


- 1.** ANDYDERIS PUTRA AJI  
SYABANA - 296530
  
- 2.** MUHAMAD  
HAMIEZI - 301785

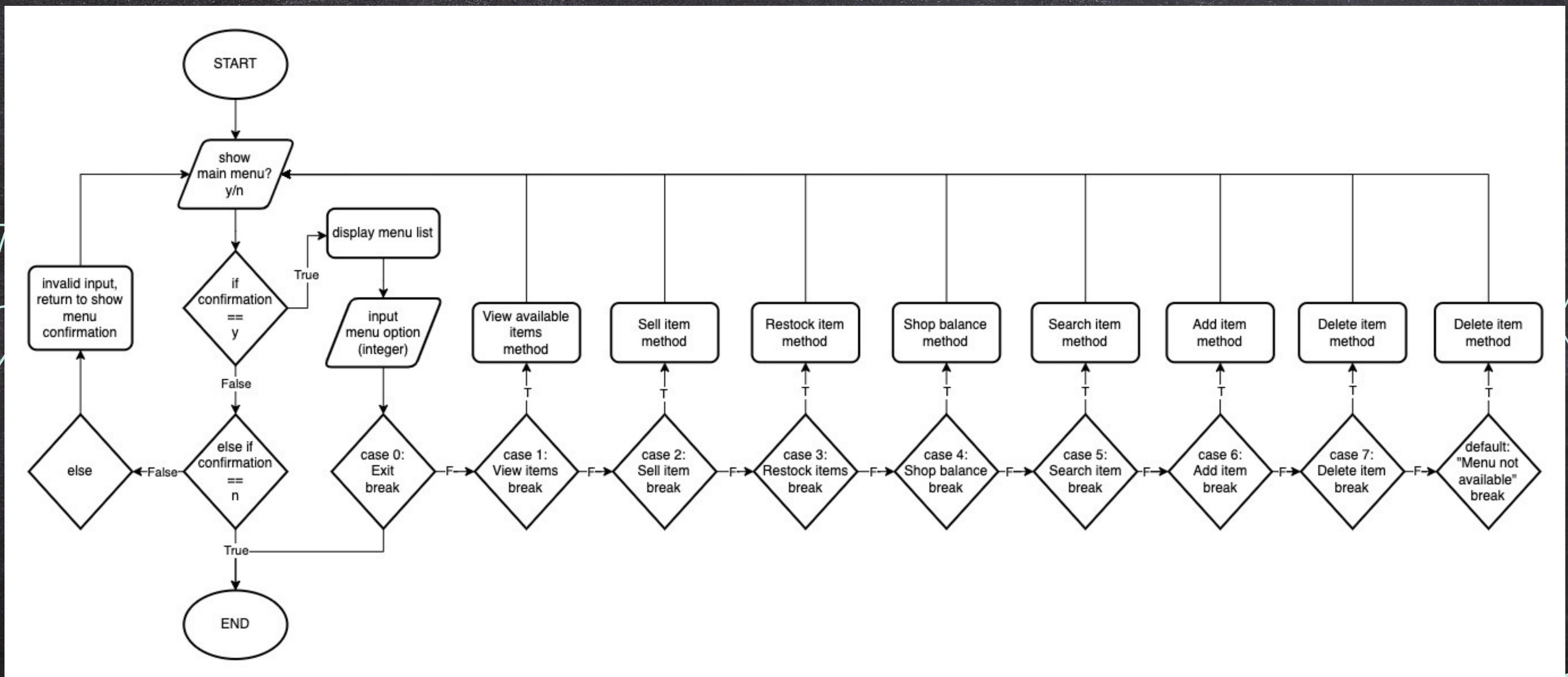
# INTRODUCTION TO THE GROUP PROJECT

Our journey begins with a warm welcome to the Computer Shop Management System, an intricately designed Java program that serves as the backbone for efficient computer shop operations. As we delve into its features, you'll discover how this system is tailored to streamline various tasks within a computer shop.

Supported by:



# PROGRAM FLOW



# SYSTEM ARCHITECTURE



## Key Components:

- Main class: `App`.
- Menu handling: `switchMenu` method.
- Menu options: Displayed using `displayMenu` method.
- Operations method: View items, sell item, restock item, view shop balance, search item, add new item, delete item.

```
Code File Edit Selection View Go Run Terminal Window Help
```

```
App.java 1, M  MenuList.java M
```

```
import java.util.Scanner;
```

```
public class App {
```

```
    public static void main(String[] args) { ... }
```

```
    public static void switchMenu() { ... }
```

```
    public static int displayMenu() { ... }
```

```
Code File Edit Selection View Go Run Terminal Window Help
```

```
App.java 1, M  MenuList.java M
```

```
//Method for menu 1
```

```
public static void viewItems() { ... }
```

```
//Method for menu 2
```

```
public static void sellItem(Scanner scanner) { ... }
```

```
//Method for menu 3
```

```
public static void restockItem(Scanner scanner) { ... }
```

```
//Method for menu 4
```

```
public static void viewShopBalance() { ... }
```

```
//Method for menu 5
```

```
public static void searchItem(Scanner scanner) { ... }
```

```
//Method for menu 6
```

```
public static void addItem(Scanner scanner) { ... }
```

```
//Method for menu 7
```

```
public static void deleteItem(Scanner scanner) { ... }
```



# MENU OPERATIONS

- View Items: `MenuList.viewItems()`
- Sell Item: `MenuList.sellItem(scanner)`
- Restock Item:  
`MenuList.restockItem(scanner)`
- View Shop Balance:  
`MenuList.viewShopBalance()`
- Search Item:  
`MenuList.searchItem(scanner)`
- Add New Item:  
`MenuList.addItem(scanner)`
- Delete Item:  
`MenuList.deleteItem(scanner)`

```
switch (menu) {  
    case 0:  
        System.out.println(x:"Exiting the program. Goodbye! \n");  
        break;  
    case 1:  
        MenuList.viewItems();  
        switchMenu();  
        break;  
    case 2:  
        MenuList.sellItem(scanner);  
        switchMenu();  
        break;  
    case 3:  
        MenuList.restockItem(scanner);  
        switchMenu();  
        break;  
    case 4:  
        MenuList.viewShopBalance();  
        switchMenu();  
        break;  
    case 5:  
        MenuList.searchItem(scanner);  
        switchMenu();  
        break;  
    case 6:  
        MenuList.addItem(scanner);  
        switchMenu();  
        break;  
    case 7:  
        MenuList.deleteItem(scanner);  
        switchMenu();  
        break;  
    default:  
        System.out.println(x:"Menu not available. Please choose menu  
        switchMenu();
```

# USER INTERACTION



## User Input Handling:

- Scanner usage for user input.
- Validations for correct input.
- Navigation through the menu options.

The screenshot shows a Java IDE interface with two tabs open: "App.java 1, M" and "MenuList.java M". The "App.java" tab is active, displaying the following code:82  
83 public static int displayMenu() {  
84 // Menu Option Display  
85 System.out.println(x:" ");  
86 System.out.println(x:"\*\*\*\*\*");  
87 System.out.println(x:"\* MAIN MENU \*");  
88 System.out.println(x:"\*");  
89 System.out.println(x:"\* 0. Exit \*");  
90 System.out.println(x:"\* 1. Show Available Items \*");  
91 System.out.println(x:"\* 2. Sell Item \*");  
92 System.out.println(x:"\* 3. Restock Item \*");  
93 System.out.println(x:"\* 4. Show Shop Balance \*");  
94 System.out.println(x:"\* 5. Search Item \*");  
95 System.out.println(x:"\* 6. Add New Item \*");  
96 System.out.println(x:"\* 7. Delete Item \*");  
97 System.out.println(x:"\*\*\*\*\*");  
98 System.out.print(s:"Enter your menu: ");  
99  
100 Scanner menu = new Scanner(System.in);  
101  
102 // Validate that the user entered correct input  
103 while (!menu.hasNextInt()) {  
104 System.out.println(x:"Invalid input please enter an integer! \n");  
105 menu.next();  
106 }  
107 System.out.print(s:"\n");  
108 return menu.nextInt();  
109 }  
110  
111 }

# LET'S RUN CODE...

The screenshot shows a Mac OS X desktop with a dark-themed Visual Studio Code window open. The title bar of the window reads "GroupAssignment-1". The left sidebar of the code editor includes icons for "EXPLORER", "SEARCH", "RUN", and "PROJECTS". The "PROJECTS" section shows a folder structure with ".vscode", "bin" (containing "App.class", "App.java.txt", "MenuList....", "MenuList.java.txt", and "settings.json"), "lib", and "src" (containing "App.java" and "MenuList.java"). The "src" folder is currently selected.

The main editor area has two tabs: "App.java 1, M" and "MenuList.java M". The "App.java" tab displays the following code:

```
import java.util.Scanner;
public class App {
    public static void main(String[] args) {
        System.out.println(" ");
        System.out.println("*****");
        System.out.println("* Welcome to Computer Shop Management System *");
        System.out.println("*");
        System.out.println("*****");
        System.out.println(" ");
        switchMenu();
    }

    public static void switchMenu() {
        System.out.println("\nDo you want to show Main Menu? (y/n) ");
        Scanner scanner = new Scanner(System.in);
        // Get user confirmation
        char confirmation = scanner.next().charAt(0);

        if (confirmation == 'y' || confirmation == 'Y') {
            int menu = displayMenu();
            do {
                // Now you can use the 'menu' variable as needed
                switch (menu) {
                    case 0:
                        System.out.println("Exiting the program. Goodbye!");
                        break;
                    case 1:
                        MenuList.viewItems();
                        switchMenu();
                        break;
                    case 2:
                        MenuList.sellItem(scanner);
                        switchMenu();
                }
            } while (menu != 0);
        }
    }
}
```

The "MenuList.java" tab displays the following code:

```
import java.util.Arrays;
import java.util.Scanner;
public class MenuList {
    // Assuming these arrays and variable are defined in the App class
    private static String[] items = { "mouse", "keyboard", "monitor", "desk" };
    private static double[] itemPrices = { 20.0, 30.0, 150.0, 500.0 };
    private static int[] itemStocks = { 50, 30, 20, 10 };
    private static double shopBalance = 10000.0;

    //Method for menu 1
    public static void viewItems() {
        System.out.println("=====");
        System.out.println("Menu 1 - Show Available Items");
        System.out.println("=====");
        System.out.println("\nItems available in the shop: \n");
        for (int i = 0; i < items.length; i++) {
            String itemName = items[i];
            double itemPrice = itemPrices[i];
            int itemStock = itemStocks[i];
            System.out.println(itemName + " - RM " + itemPrice + " | Stock: " + itemStock);
        }
    }

    //Method for menu 2
    public static void sellItem(Scanner scanner) {
        System.out.println("=====");
        System.out.println("Menu 2 - Sell Item");
        System.out.println("=====");

        for (String itemName : items) {
            System.out.println(itemName);
        }

        System.out.print("\nEnter the item name to sell: ");
        String itemName = scanner.next();
        int itemIndex = Arrays.asList(items).indexOf(itemName);

        if (itemIndex != -1 && itemStocks[itemIndex] > 0) {
            double itemPrice = itemPrices[itemIndex];
            itemStocks[itemIndex]--;
            shopBalance += itemPrice;
            System.out.println("Sold " + itemName + " for RM" + itemPrice);
        } else {
            System.out.println("Item not available or out of stock.");
        }
    }
}
```

The status bar at the bottom of the code editor shows "Array-version\*" and "Ln 23, Col 6".



REVIEW SOURCE  
CODE ON GITHUB

JUST SCAN THEN OPEN  
"SRC" FOLDER...



# Q & A

# THANK YOU!