DATA STRUCTURES LAB FILE

Submitted by : Yashraj Rawani

Submitted to : SWATI MAM

Course : B.TECH CSE DATA SCIENCE

Semester : III

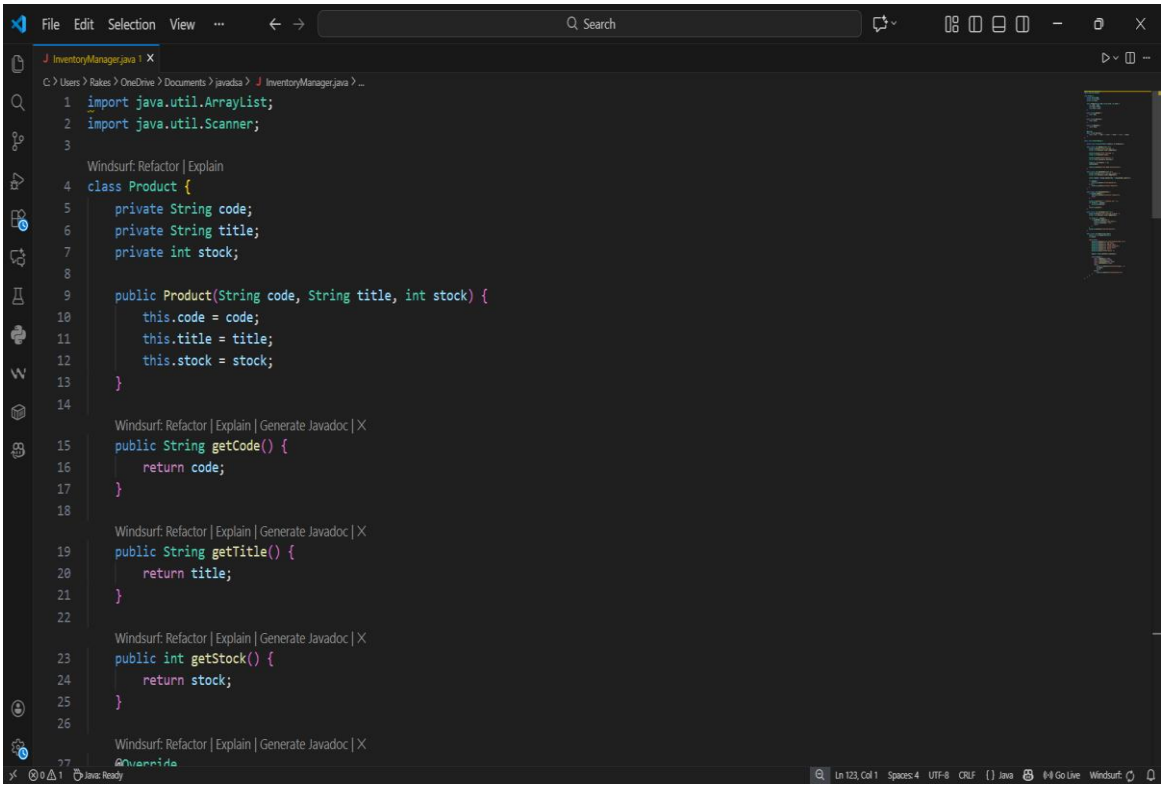
Roll No : 2401420037

TASK-1

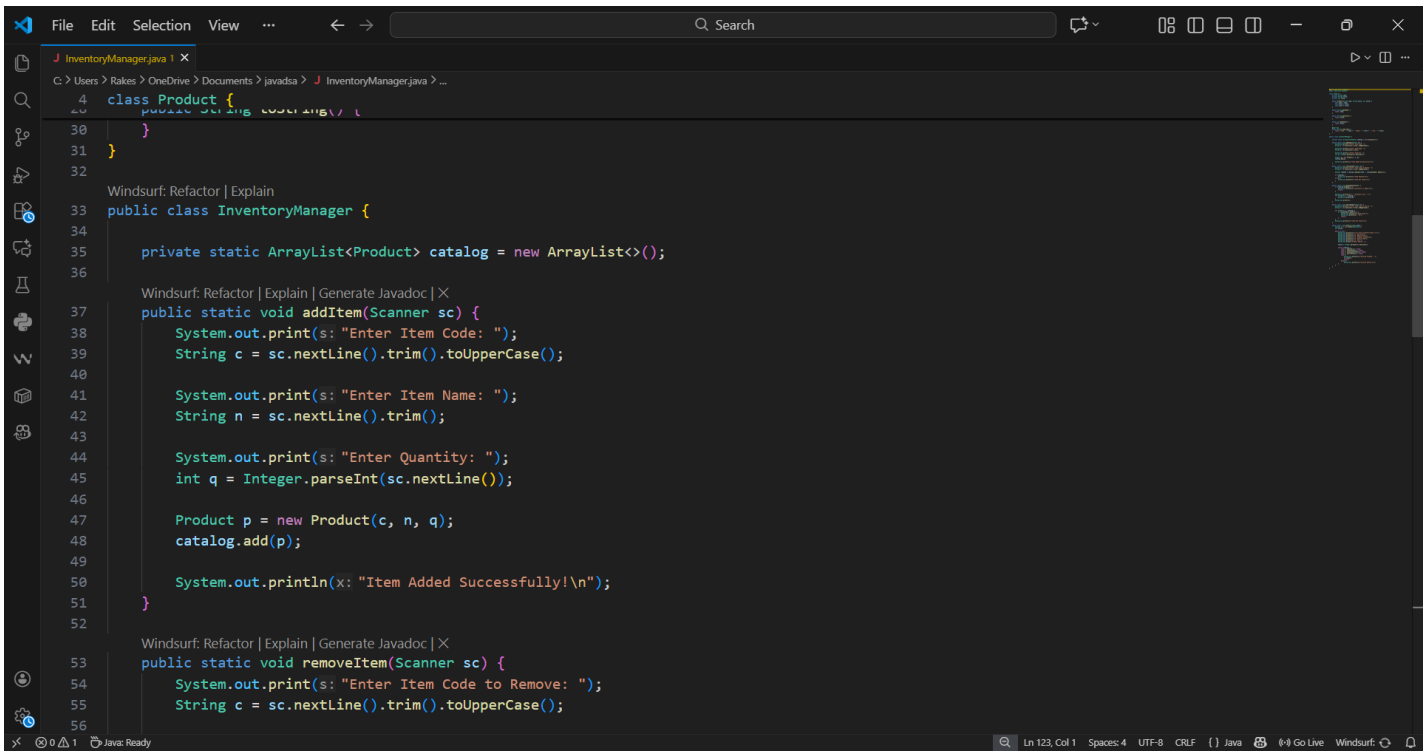
- Store Inventory:To manage items in inventory

{Add item,remove item,display inventory,search item}

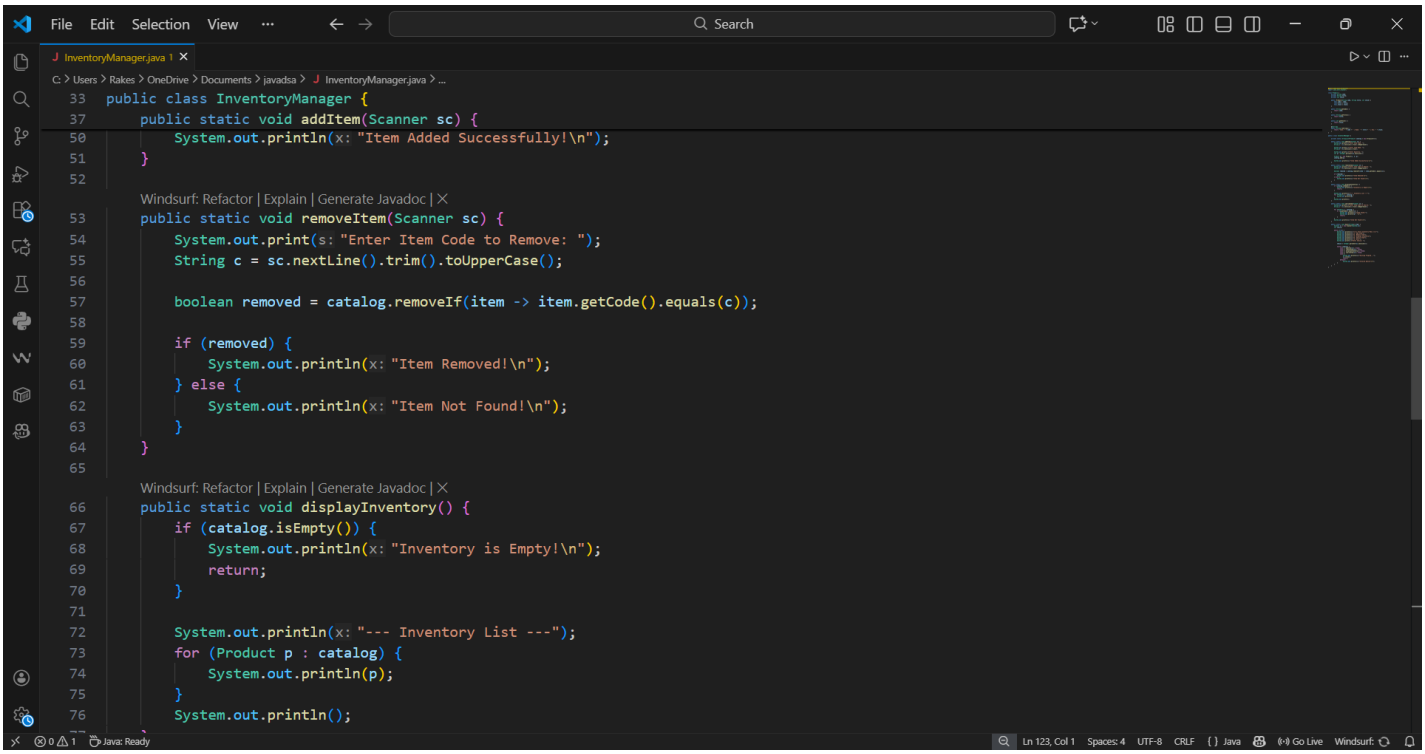
- Code:



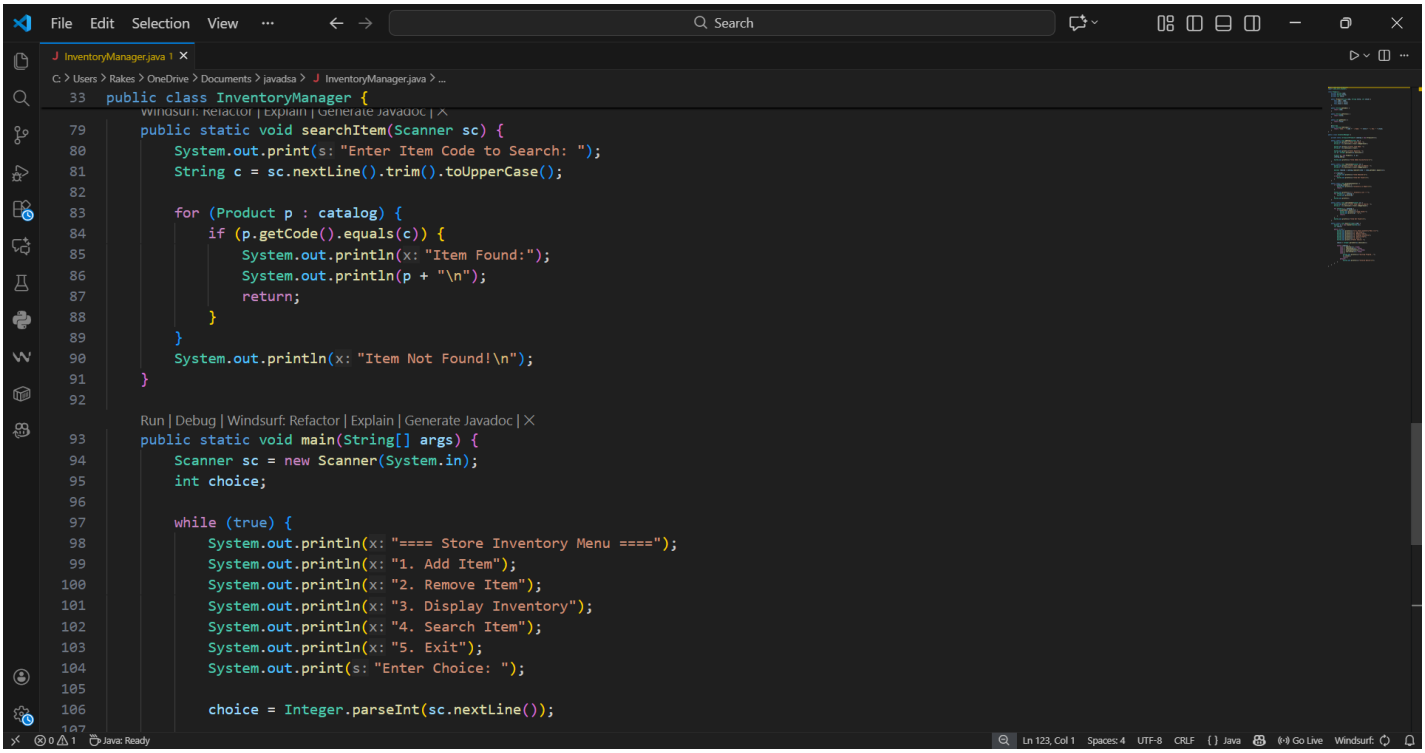
```java
import java.util.ArrayList;
import java.util.Scanner;

class Product {
    private String code;
    private String title;
    private int stock;

    public Product(String code, String title, int stock) {
        this.code = code;
        this.title = title;
        this.stock = stock;
    }

    public String getCode() {
        return code;
    }

    public String getTitle() {
        return title;
    }

    public int getStock() {
        return stock;
    }
```



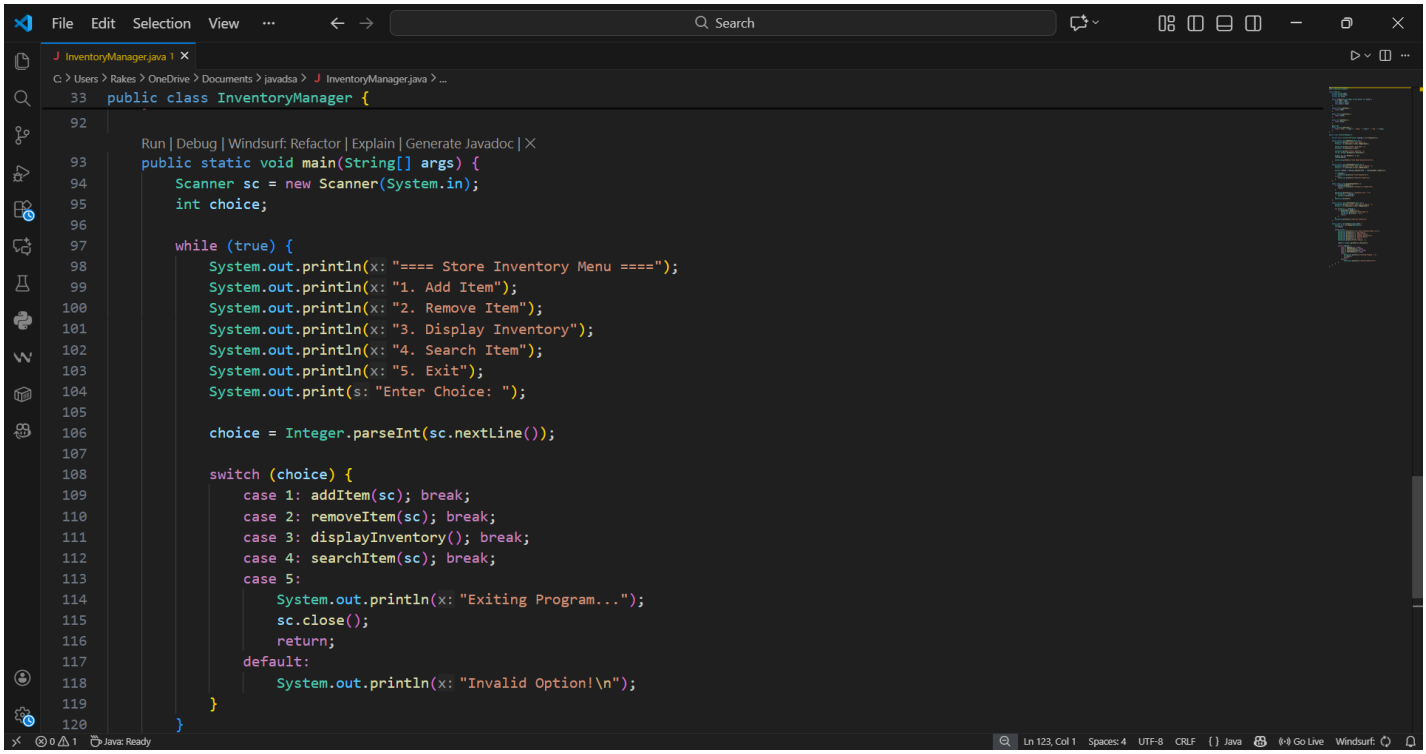
```java
    }
}

public class InventoryManager {

    private static ArrayList<Product> catalog = new ArrayList<>();

    public static void addItem(Scanner sc) {
        System.out.print("Enter Item Code: ");
        String c = sc.nextLine().trim().toUpperCase();

        System.out.print("Enter Item Name: ");
        String n = sc.nextLine().trim();

        System.out.print("Enter Quantity: ");
        int q = Integer.parseInt(sc.nextLine());

        Product p = new Product(c, n, q);
        catalog.add(p);

        System.out.println("Item Added Successfully!\n");
    }

    public static void removeItem(Scanner sc) {
        System.out.print("Enter Item Code to Remove: ");
        String c = sc.nextLine().trim().toUpperCase();
```



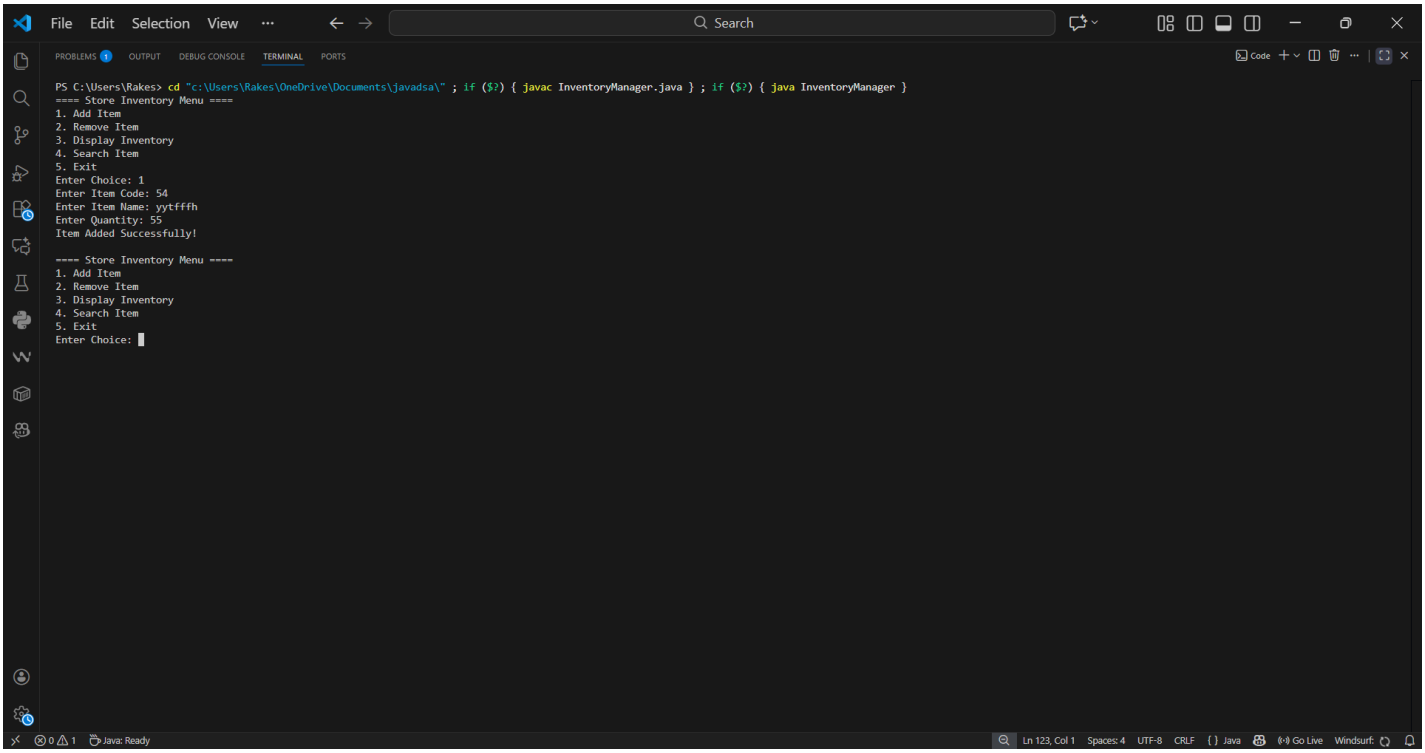
```
File  Edit  Selection  View  ···  Search

InventoryManager.java 1

C: > Users > Rakes > OneDrive > Documents > javadsa > InventoryManager.java > ...

33  public class InventoryManager {
37      public static void addItem(Scanner sc) {
50          System.out.println(x: "Item Added Successfully!\n");
51      }
52
        Windsurf: Refactor | Explain | Generate Javadoc | X
53      public static void removeItem(Scanner sc) {
54          System.out.print(s: "Enter Item Code to Remove: ");
55          String c = sc.nextLine().trim().toUpperCase();
56
57          boolean removed = catalog.removeIf(item -> item.getCode().equals(c));
58
59          if (removed) {
60              System.out.println(x: "Item Removed!\n");
61          } else {
62              System.out.println(x: "Item Not Found!\n");
63          }
64      }
65
        Windsurf: Refactor | Explain | Generate Javadoc | X
66      public static void displayInventory() {
67          if (catalog.isEmpty()) {
68              System.out.println(x: "Inventory is Empty!\n");
69              return;
70          }
71
72          System.out.println(x: "--- Inventory List ---");
73          for (Product p : catalog) {
74              System.out.println(p);
75          }
76          System.out.println();

0 1  Java: Ready   Ln 123, Col 1  Spaces: 4  UTF-8  CRLF  { } Java  Go Live  Windsurf:
```



```
File  Edit  Selection  View  ···  Search

InventoryManager.java 1

C: > Users > Rakes > OneDrive > Documents > javadsa > InventoryManager.java > ...

33  public class InventoryManager {
        Windsurf: Refactor | Explain | Generate Javadoc | X
79      public static void searchItem(Scanner sc) {
80          System.out.print(s: "Enter Item Code to Search: ");
81          String c = sc.nextLine().trim().toUpperCase();
82
83          for (Product p : catalog) {
84              if (p.getCode().equals(c)) {
85                  System.out.println(x: "Item Found:");
86                  System.out.println(p + "\n");
87                  return;
88              }
89          }
90          System.out.println(x: "Item Not Found!\n");
91      }
92
        Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | X
93      public static void main(String[] args) {
94          Scanner sc = new Scanner(System.in);
95          int choice;
96
97          while (true) {
98              System.out.println(x: "==== Store Inventory Menu ====");
99              System.out.println(x: "1. Add Item");
100             System.out.println(x: "2. Remove Item");
101             System.out.println(x: "3. Display Inventory");
102             System.out.println(x: "4. Search Item");
103             System.out.println(x: "5. Exit");
104             System.out.print(s: "Enter Choice: ");
105
106             choice = Integer.parseInt(sc.nextLine());

0 1  Java: Ready   Ln 123, Col 1  Spaces: 4  UTF-8  CRLF  { } Java  Go Live  Windsurf:
```

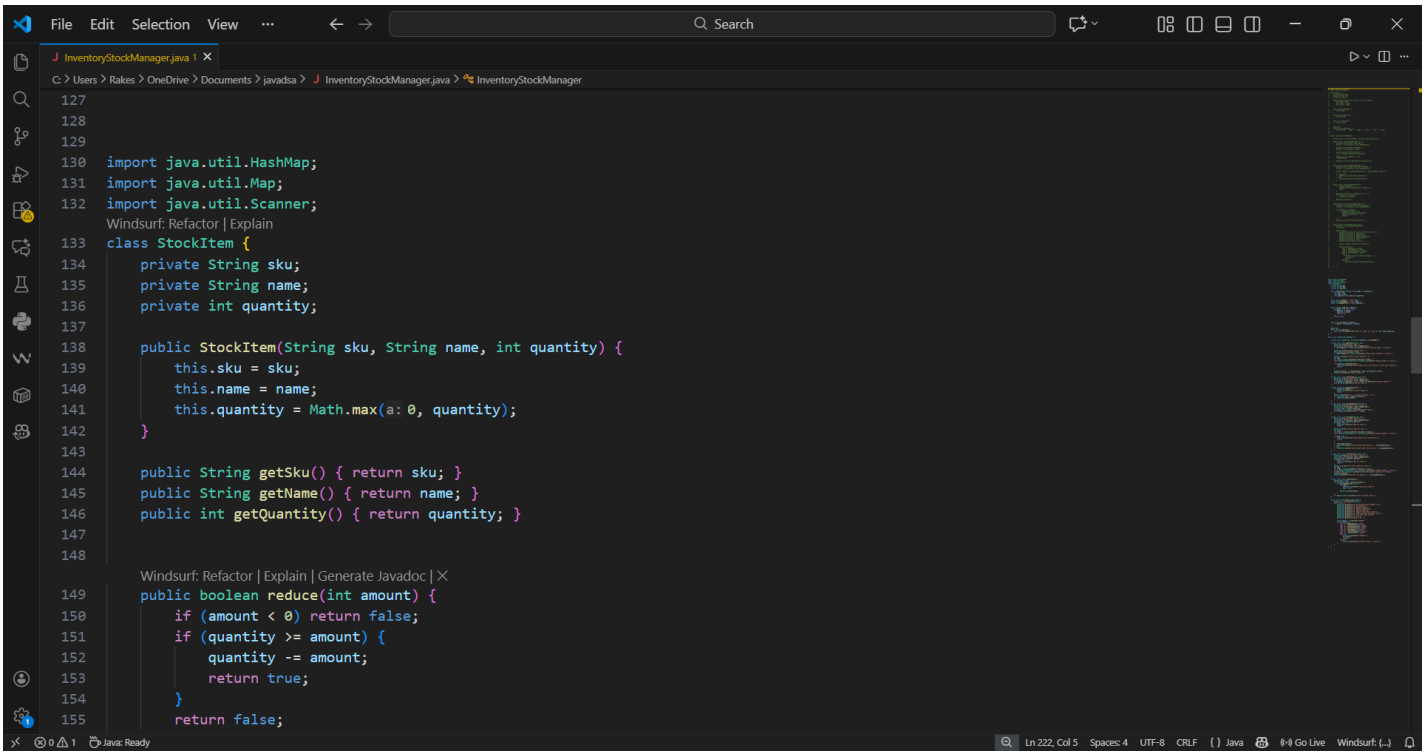

File Edit Selection View ···
Search
InventoryManager.java 1
C: > Users > Rakes > OneDrive > Documents > javadsa > InventoryManager.java > ...
public class InventoryManager {
Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int choice;
while (true) {
System.out.println(x: "==== Store Inventory Menu ====");
System.out.println(x: "1. Add Item");
System.out.println(x: "2. Remove Item");
System.out.println(x: "3. Display Inventory");
System.out.println(x: "4. Search Item");
System.out.println(x: "5. Exit");
System.out.print(s: "Enter Choice: ");
choice = Integer.parseInt(sc.nextLine());
switch (choice) {
case 1: addItem(sc); break;
case 2: removeItem(sc); break;
case 3: displayInventory(); break;
case 4: searchItem(sc); break;
case 5:
System.out.println(x: "Exiting Program...");
sc.close();
return;
default:
System.out.println(x: "Invalid Option!\n");
}
}
Java: Ready
Ln 123, Col 1
Spaces: 4
UTF-8
CRLF
Java
Go Live
Windsurf:

*Output:

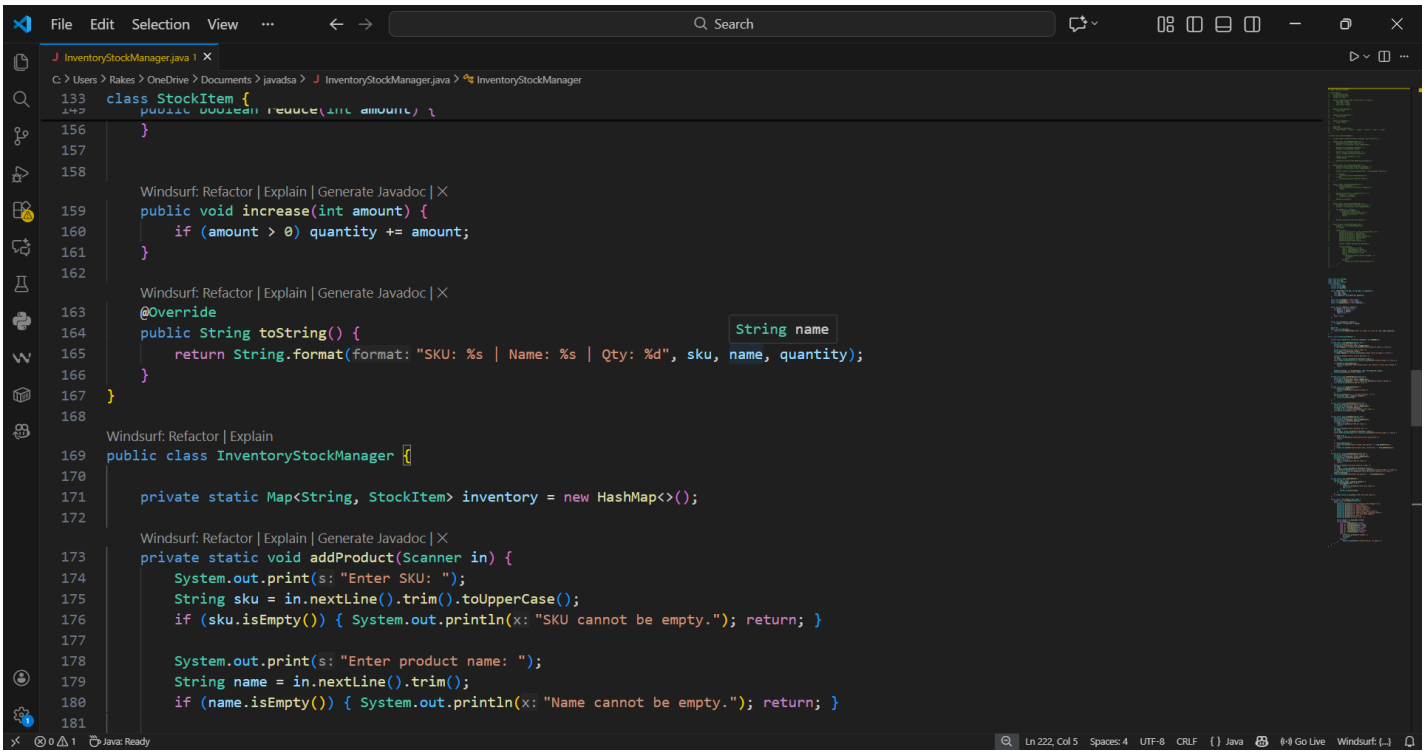

File Edit Selection View ···
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac InventoryManager.java } ; if ($?) { java InventoryManager }
==== Store Inventory Menu ====
1. Add Item
2. Remove Item
3. Display Inventory
4. Search Item
5. Exit
Enter Choice: 1
Enter Item Code: 54
Enter Item Name: yytfffh
Enter Quantity: 55
Item Added Successfully!
==== Store Inventory Menu ====
1. Add Item
2. Remove Item
3. Display Inventory
4. Search Item
5. Exit
Enter Choice:
Java: Ready
Ln 123, Col 1 Spaces: 4 UTF-8 CRLF { } Java Go Live Windsurf:

TASK-2

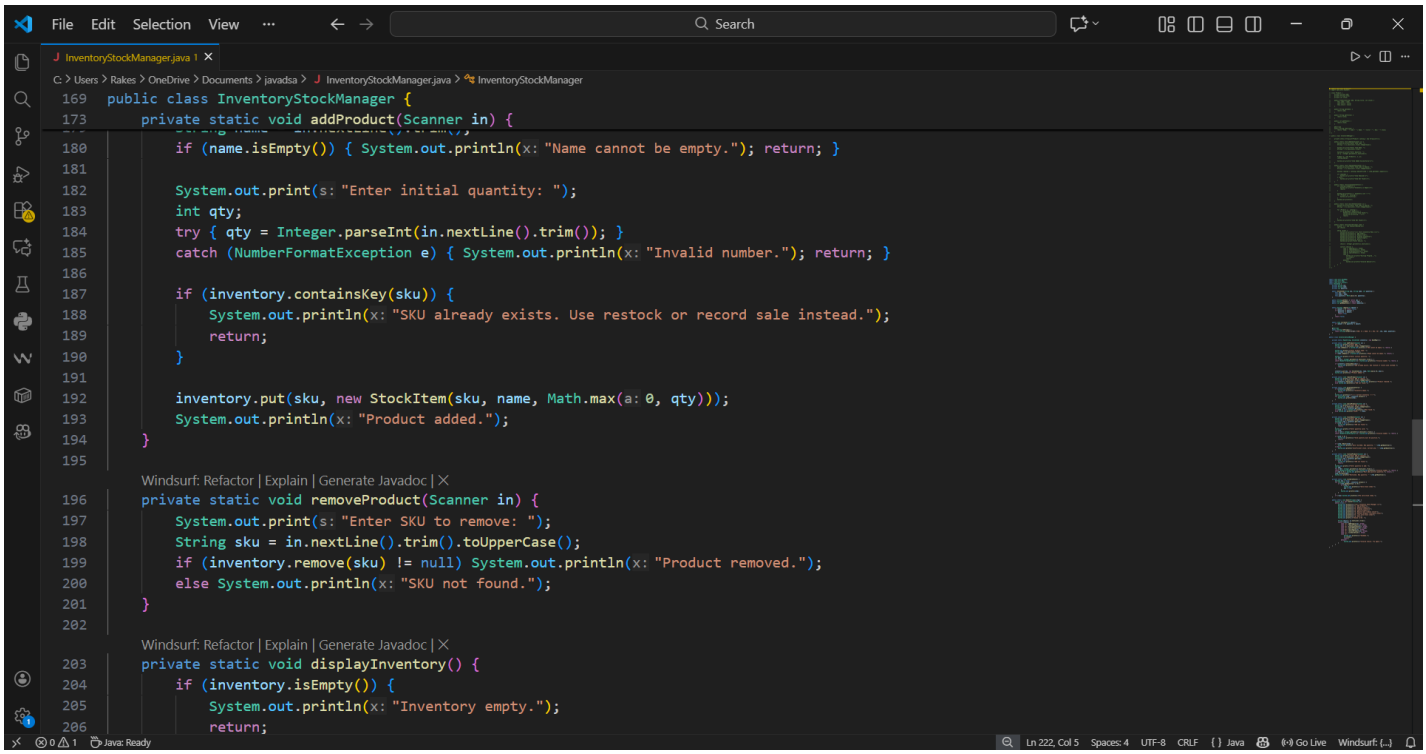
- Inventory Stock Manager:Reduce stock according to sales and identify zero stock.
  - Code:



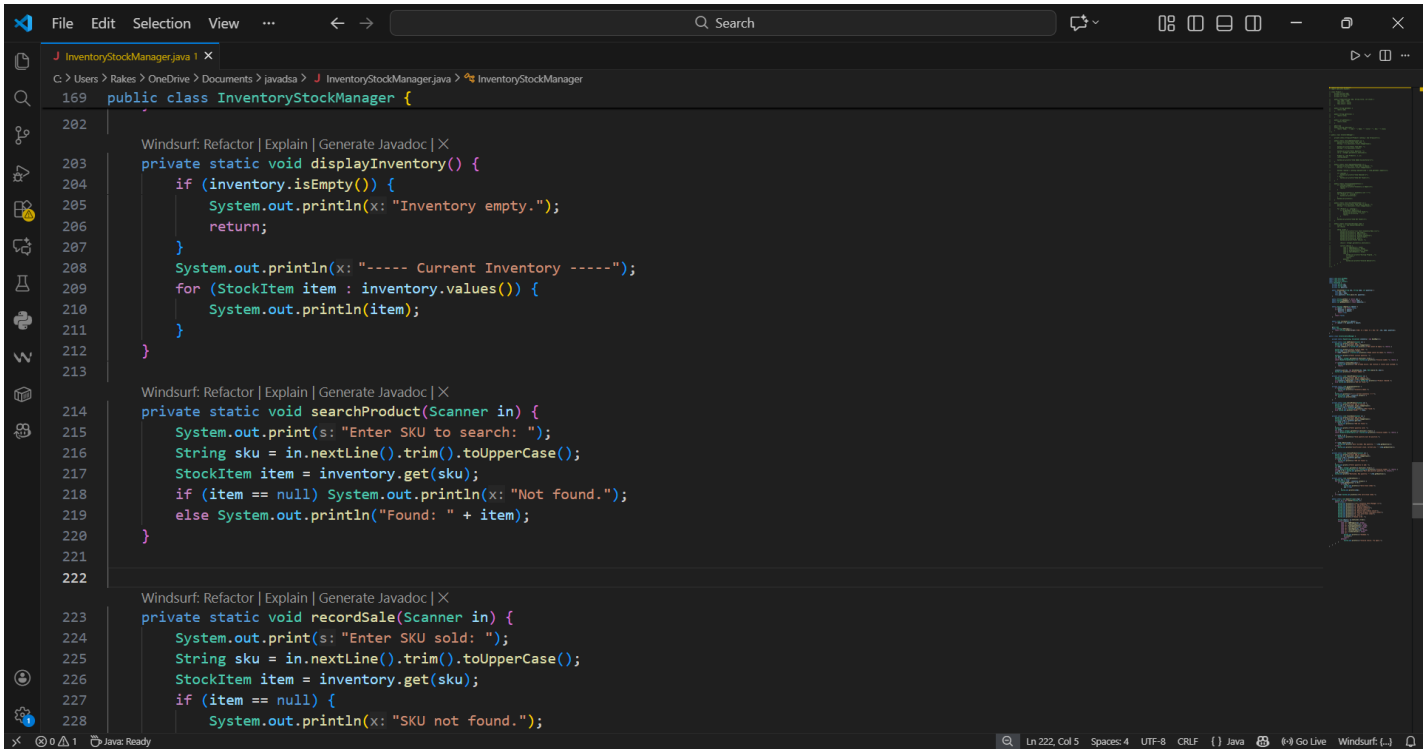
```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
Windsurf: Refactor | Explain
class StockItem {
    private String sku;
    private String name;
    private int quantity;

    public StockItem(String sku, String name, int quantity) {
        this.sku = sku;
        this.name = name;
        this.quantity = Math.max(a: 0, quantity);
    }

    public String getSku() { return sku; }
    public String getName() { return name; }
    public int getQuantity() { return quantity; }


    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public boolean reduce(int amount) {
        if (amount < 0) return false;
        if (quantity >= amount) {
            quantity -= amount;
            return true;
        }
        return false;
```



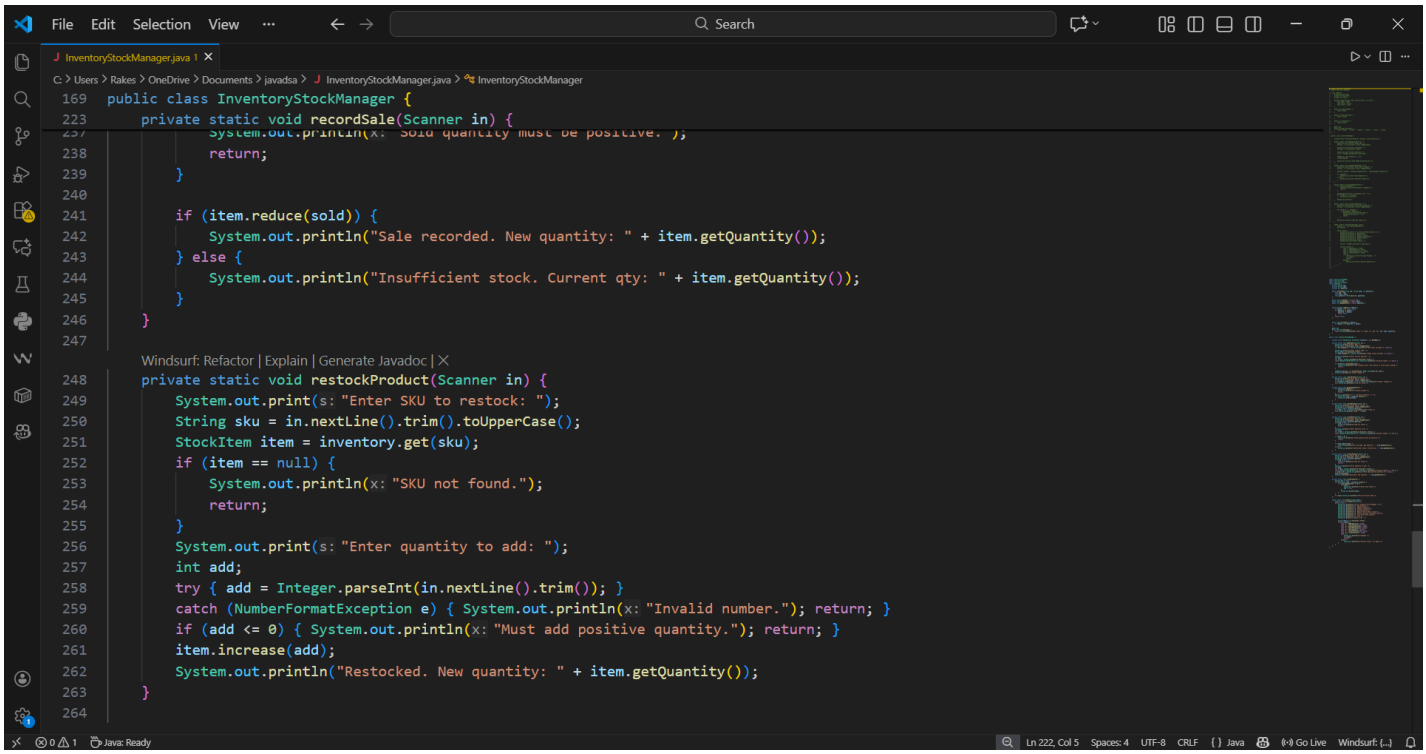
```
class StockItem {
    }


    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void increase(int amount) {
        if (amount > 0) quantity += amount;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    @Override
    public String toString() {                                  String name
        return String.format(format: "SKU: %s | Name: %s | Qty: %d", sku, name, quantity);
    }
}

Windsurf: Refactor | Explain
public class InventoryStockManager {

    private static Map<String, StockItem> inventory = new HashMap<>();

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    private static void addProduct(Scanner in) {
        System.out.print(s: "Enter SKU: ");
        String sku = in.nextLine().trim().toUpperCase();
        if (sku.isEmpty()) { System.out.println(x: "SKU cannot be empty."); return; }

        System.out.print(s: "Enter product name: ");
        String name = in.nextLine().trim();
        if (name.isEmpty()) { System.out.println(x: "Name cannot be empty."); return; }
```



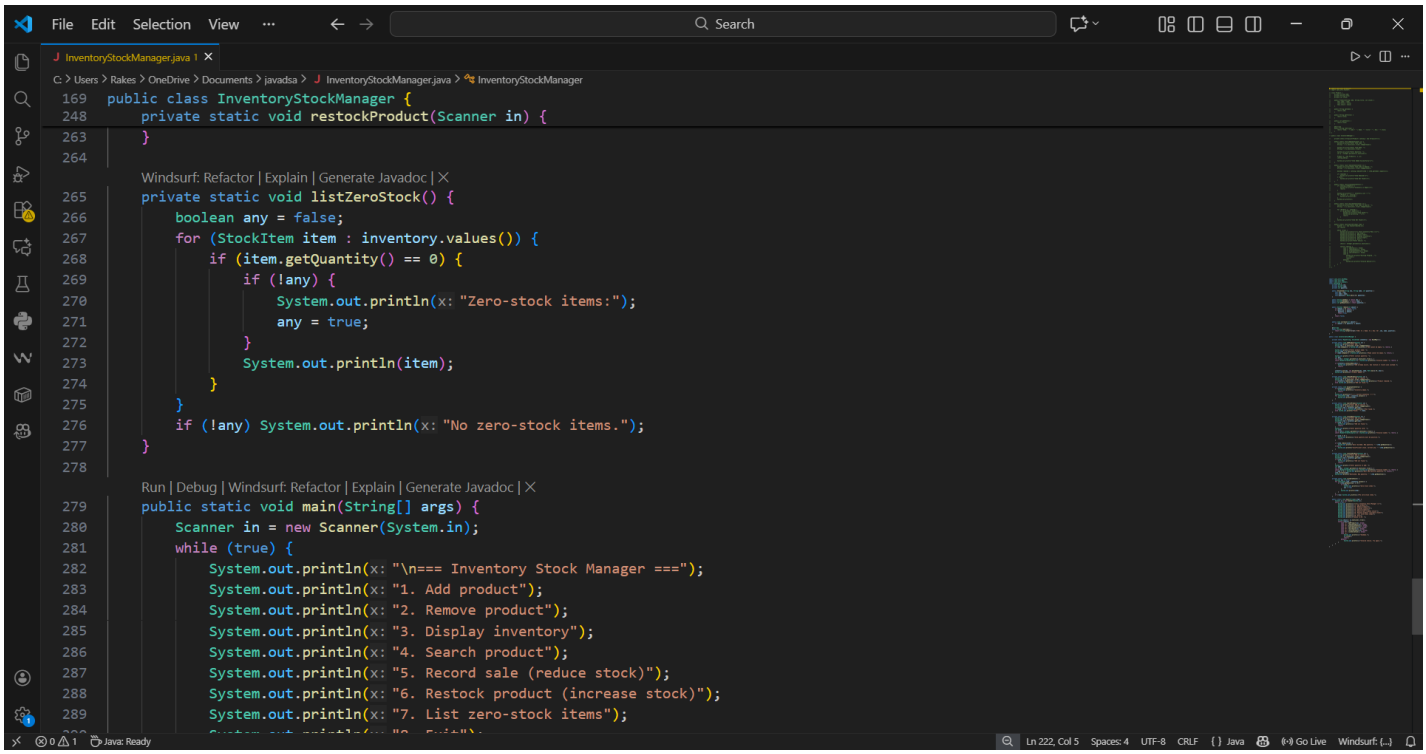
```
169  public class InventoryStockManager {
173      private static void addProduct(Scanner in) {
180          if (name.isEmpty()) { System.out.println(x: "Name cannot be empty."); return; }
181
182          System.out.print(s: "Enter initial quantity: ");
183          int qty;
184          try { qty = Integer.parseInt(in.nextLine().trim()); }
185          catch (NumberFormatException e) { System.out.println(x: "Invalid number."); return; }
186
187          if (inventory.containsKey(sku)) {
188              System.out.println(x: "SKU already exists. Use restock or record sale instead.");
189              return;
190          }
191
192          inventory.put(sku, new StockItem(sku, name, Math.max(a: 0, qty)));
193          System.out.println(x: "Product added.");
194      }
195
          Windsurf: Refactor | Explain | Generate Javadoc | ×
196      private static void removeProduct(Scanner in) {
197          System.out.print(s: "Enter SKU to remove: ");
198          String sku = in.nextLine().trim().toUpperCase();
199          if (inventory.remove(sku) != null) System.out.println(x: "Product removed.");
200          else System.out.println(x: "SKU not found.");
201      }
202
          Windsurf: Refactor | Explain | Generate Javadoc | ×
203      private static void displayInventory() {
204          if (inventory.isEmpty()) {
205              System.out.println(x: "Inventory empty.");
206              return;
```



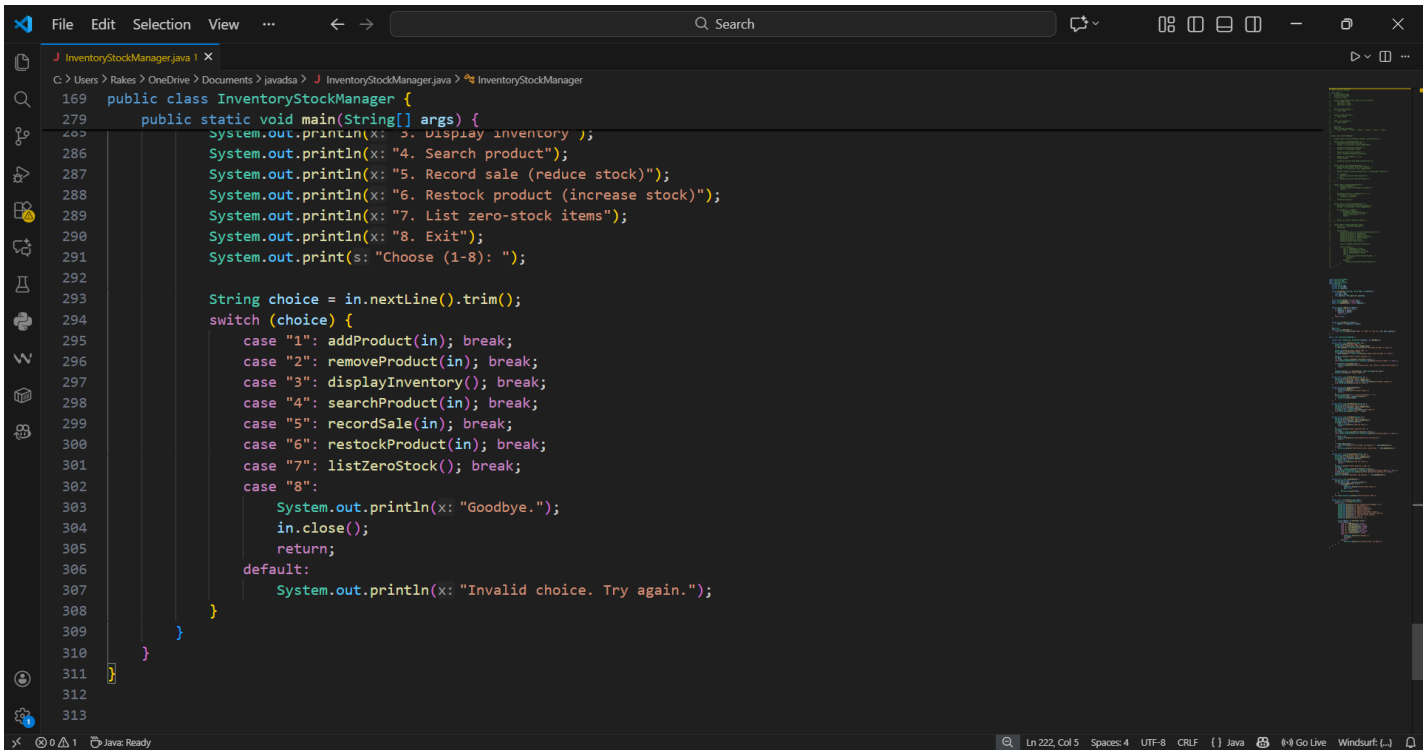
```
169  public class InventoryStockManager {
202
          Windsurf: Refactor | Explain | Generate Javadoc | ×
203      private static void displayInventory() {
204          if (inventory.isEmpty()) {
205              System.out.println(x: "Inventory empty.");
206              return;
207          }
208          System.out.println(x: "----- Current Inventory -----");
209          for (StockItem item : inventory.values()) {
210              System.out.println(item);
211          }
212      }
213
          Windsurf: Refactor | Explain | Generate Javadoc | ×
214      private static void searchProduct(Scanner in) {
215          System.out.print(s: "Enter SKU to search: ");
216          String sku = in.nextLine().trim().toUpperCase();
217          StockItem item = inventory.get(sku);
218          if (item == null) System.out.println(x: "Not found.");
219          else System.out.println("Found: " + item);
220      }
221
222
          Windsurf: Refactor | Explain | Generate Javadoc | ×
223      private static void recordSale(Scanner in) {
224          System.out.print(s: "Enter SKU sold: ");
225          String sku = in.nextLine().trim().toUpperCase();
226          StockItem item = inventory.get(sku);
227          if (item == null) {
228              System.out.println(x: "SKU not found.");
```



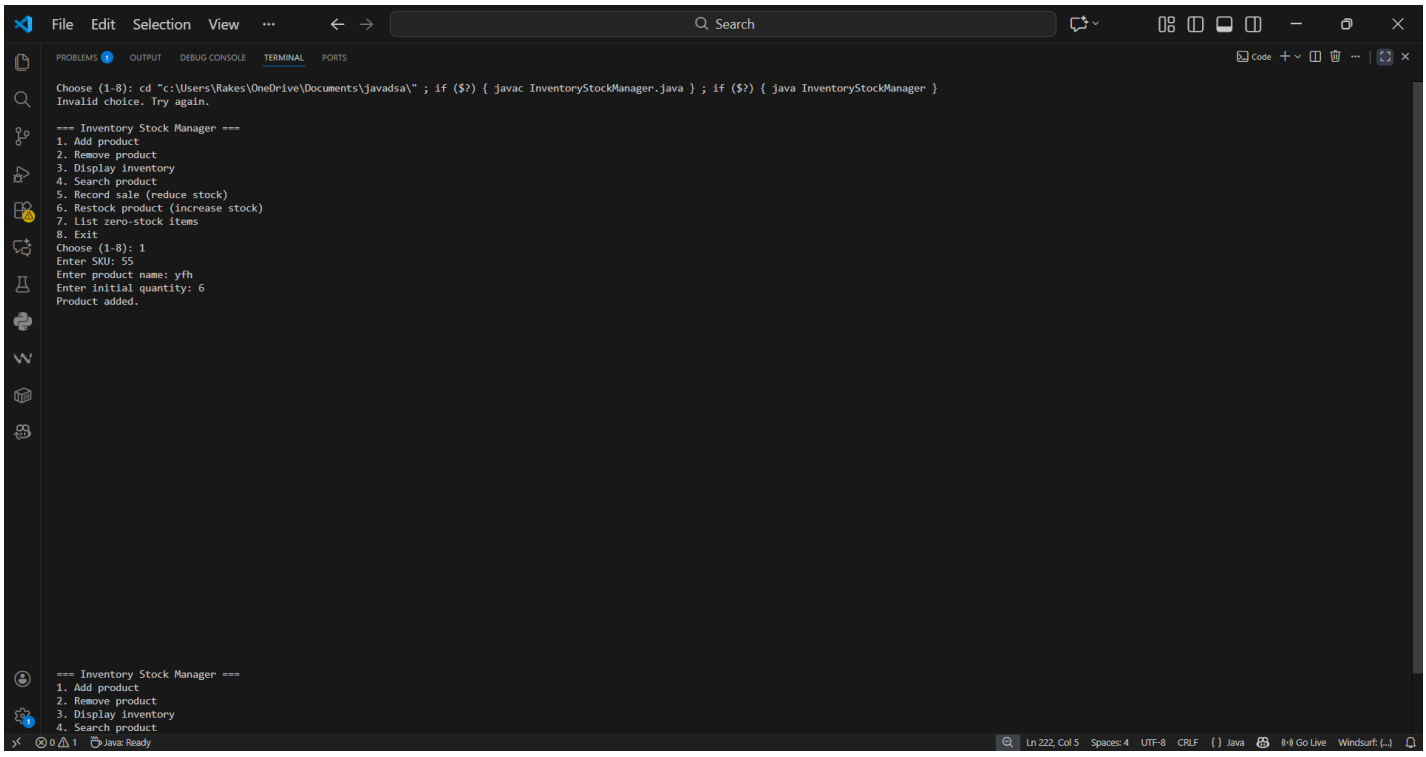
```
169  public class InventoryStockManager {
223      private static void recordSale(Scanner in) {
238              return;
239          }
240
241          if (item.reduce(sold)) {
242              System.out.println("Sale recorded. New quantity: " + item.getQuantity());
243          } else {
244              System.out.println("Insufficient stock. Current qty: " + item.getQuantity());
245          }
246      }
247
         Windsurf: Refactor | Explain | Generate Javadoc | X
248      private static void restockProduct(Scanner in) {
249          System.out.print(s: "Enter SKU to restock: ");
250          String sku = in.nextLine().trim().toUpperCase();
251          StockItem item = inventory.get(sku);
252          if (item == null) {
253              System.out.println(x: "SKU not found.");
254              return;
255          }
256          System.out.print(s: "Enter quantity to add: ");
257          int add;
258          try { add = Integer.parseInt(in.nextLine().trim()); }
259          catch (NumberFormatException e) { System.out.println(x: "Invalid number."); return; }
260          if (add <= 0) { System.out.println(x: "Must add positive quantity."); return; }
261          item.increase(add);
262          System.out.println("Restocked. New quantity: " + item.getQuantity());
263      }
264
```



```
169  public class InventoryStockManager {
248      private static void restockProduct(Scanner in) {
263      }
264
         Windsurf: Refactor | Explain | Generate Javadoc | X
265      private static void listZeroStock() {
266          boolean any = false;
267          for (StockItem item : inventory.values()) {
268              if (item.getQuantity() == 0) {
269                  if (!any) {
270                      System.out.println(x: "Zero-stock items:");
271                      any = true;
272                  }
273                  System.out.println(item);
274              }
275          }
276          if (!any) System.out.println(x: "No zero-stock items.");
277      }
278
         Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | X
279      public static void main(String[] args) {
280          Scanner in = new Scanner(System.in);
281          while (true) {
282              System.out.println(x: "\n=== Inventory Stock Manager ===");
283              System.out.println(x: "1. Add product");
284              System.out.println(x: "2. Remove product");
285              System.out.println(x: "3. Display inventory");
286              System.out.println(x: "4. Search product");
287              System.out.println(x: "5. Record sale (reduce stock)");
288              System.out.println(x: "6. Restock product (increase stock)");
289              System.out.println(x: "7. List zero-stock items");
```


File Edit Selection View
Search
InventoryStockManager.java 1
C: > Users > Rakes > OneDrive > Documents > javadsa > InventoryStockManager.java > InventoryStockManager
public class InventoryStockManager {
public static void main(String[] args) {
System.out.println("4. Search product");
System.out.println("5. Record sale (reduce stock)");
System.out.println("6. Restock product (increase stock)");
System.out.println("7. List zero-stock items");
System.out.println("8. Exit");
System.out.print("Choose (1-8): ");
String choice = in.nextLine().trim();
switch (choice) {
case "1": addProduct(in); break;
case "2": removeProduct(in); break;
case "3": displayInventory(); break;
case "4": searchProduct(in); break;
case "5": recordSale(in); break;
case "6": restockProduct(in); break;
case "7": listZeroStock(); break;
case "8":
System.out.println("Goodbye.");
in.close();
return;
default:
System.out.println("Invalid choice. Try again.");
}
}
}
}

Output:


File Edit Selection View
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Choose (1-8): cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac InventoryStockManager.java } ; if ($?) { java InventoryStockManager }
Invalid choice. Try again.
=== Inventory Stock Manager ===
1. Add product
2. Remove product
3. Display inventory
4. Search product
5. Record sale (reduce stock)
6. Restock product (increase stock)
7. List zero-stock items
8. Exit
Choose (1-8): 1
Enter SKU: 55
Enter product name: yfh
Enter initial quantity: 6
Product added.
=== Inventory Stock Manager ===
1. Add product
2. Remove product
3. Display inventory
4. Search product

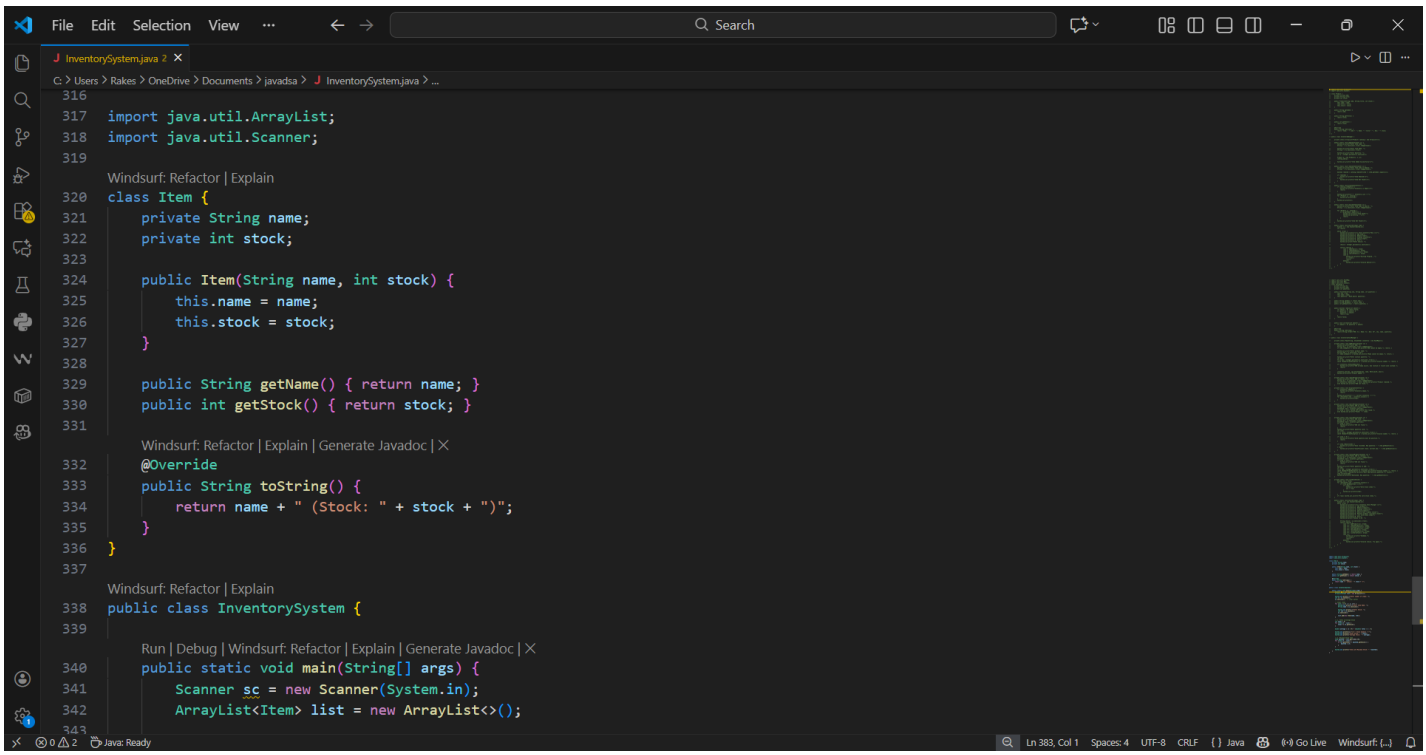
TASK-3

Inventory system with two functionalities:

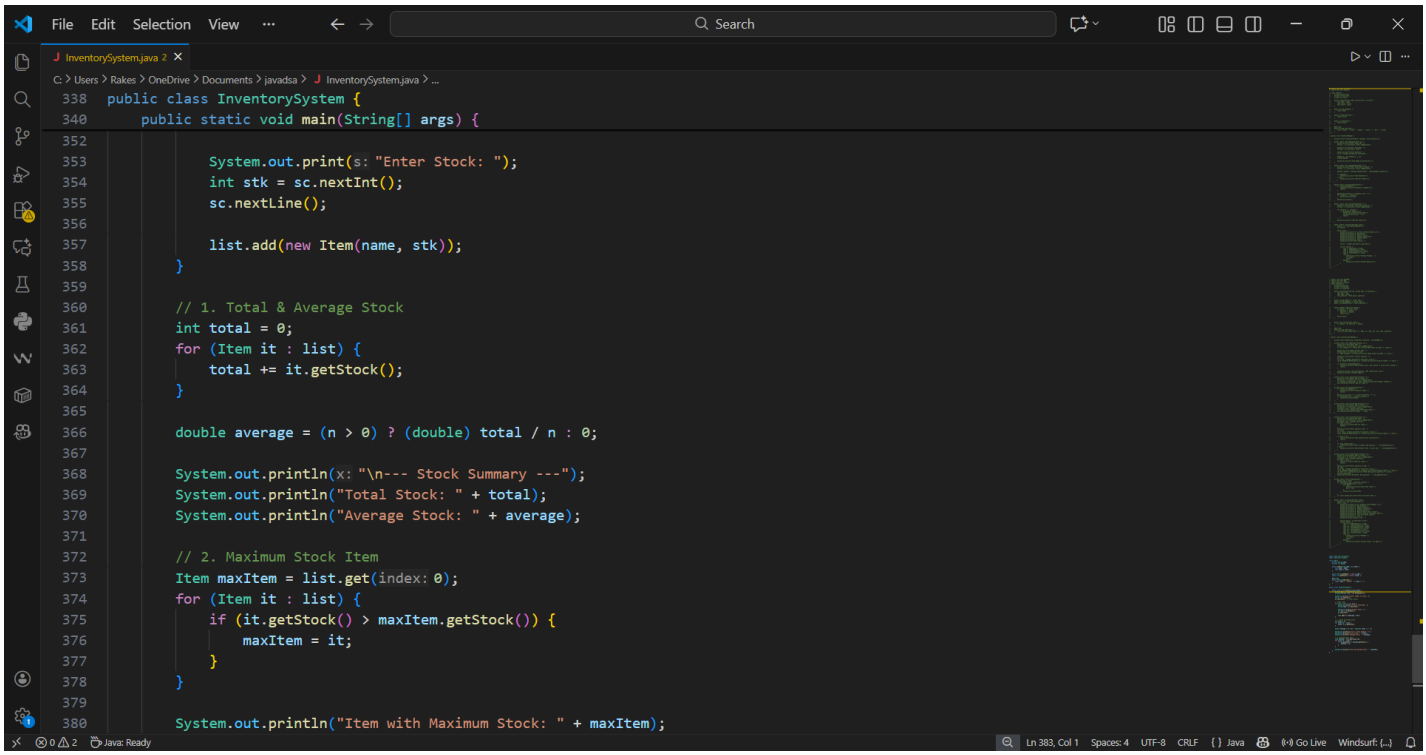
1. Calculate Total & Average Stock

2. Find Maximum Stock Item

*CODE:

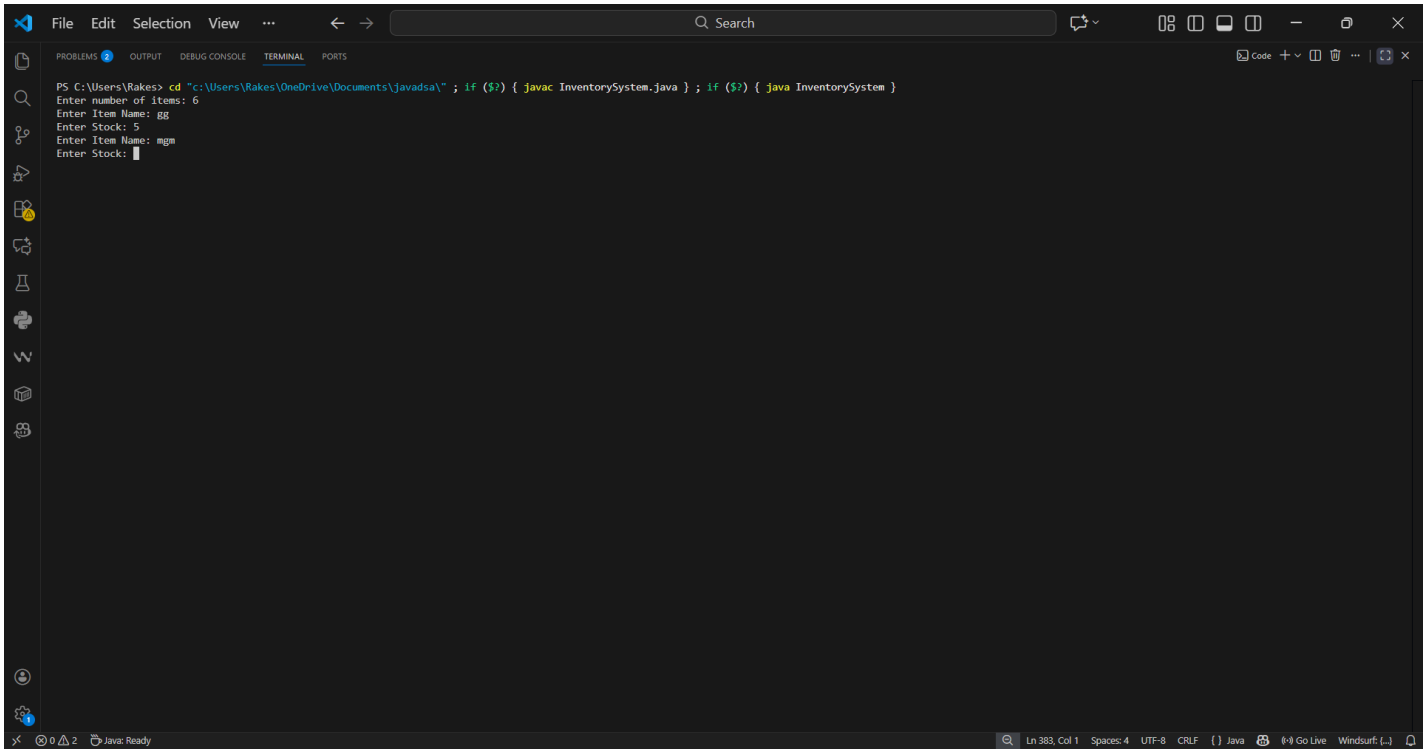


```java
import java.util.ArrayList;
import java.util.Scanner;

Windsurf: Refactor | Explain
class Item {
    private String name;
    private int stock;

    public Item(String name, int stock) {
        this.name = name;
        this.stock = stock;
    }

    public String getName() { return name; }
    public int getStock() { return stock; }

    Windsurf: Refactor | Explain | Generate Javadoc |
    @Override
    public String toString() {
        return name + " (Stock: " + stock + ")";
    }
}

Windsurf: Refactor | Explain
public class InventorySystem {

    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<Item> list = new ArrayList<>();
```



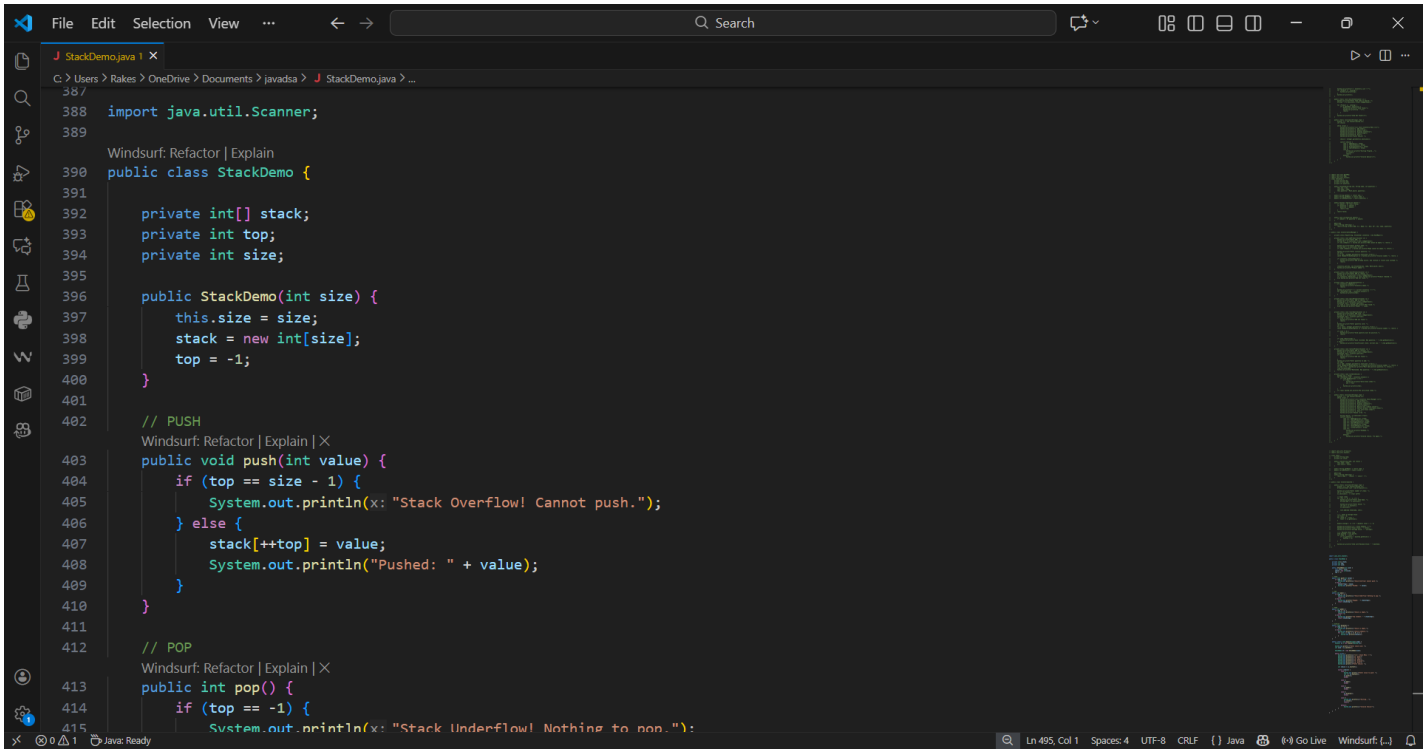
```java
public class InventorySystem {
    public static void main(String[] args) {

            System.out.print(s: "Enter Stock: ");
            int stk = sc.nextInt();
            sc.nextLine();

            list.add(new Item(name, stk));
        }

        // 1. Total & Average Stock
        int total = 0;
        for (Item it : list) {
            total += it.getStock();
        }

        double average = (n > 0) ? (double) total / n : 0;

        System.out.println(x: "\n--- Stock Summary ---");
        System.out.println("Total Stock: " + total);
        System.out.println("Average Stock: " + average);

        // 2. Maximum Stock Item
        Item maxItem = list.get(index: 0);
        for (Item it : list) {
            if (it.getStock() > maxItem.getStock()) {
                maxItem = it;
            }
        }

        System.out.println("Item with Maximum Stock: " + maxItem);
```

OUTPUT:

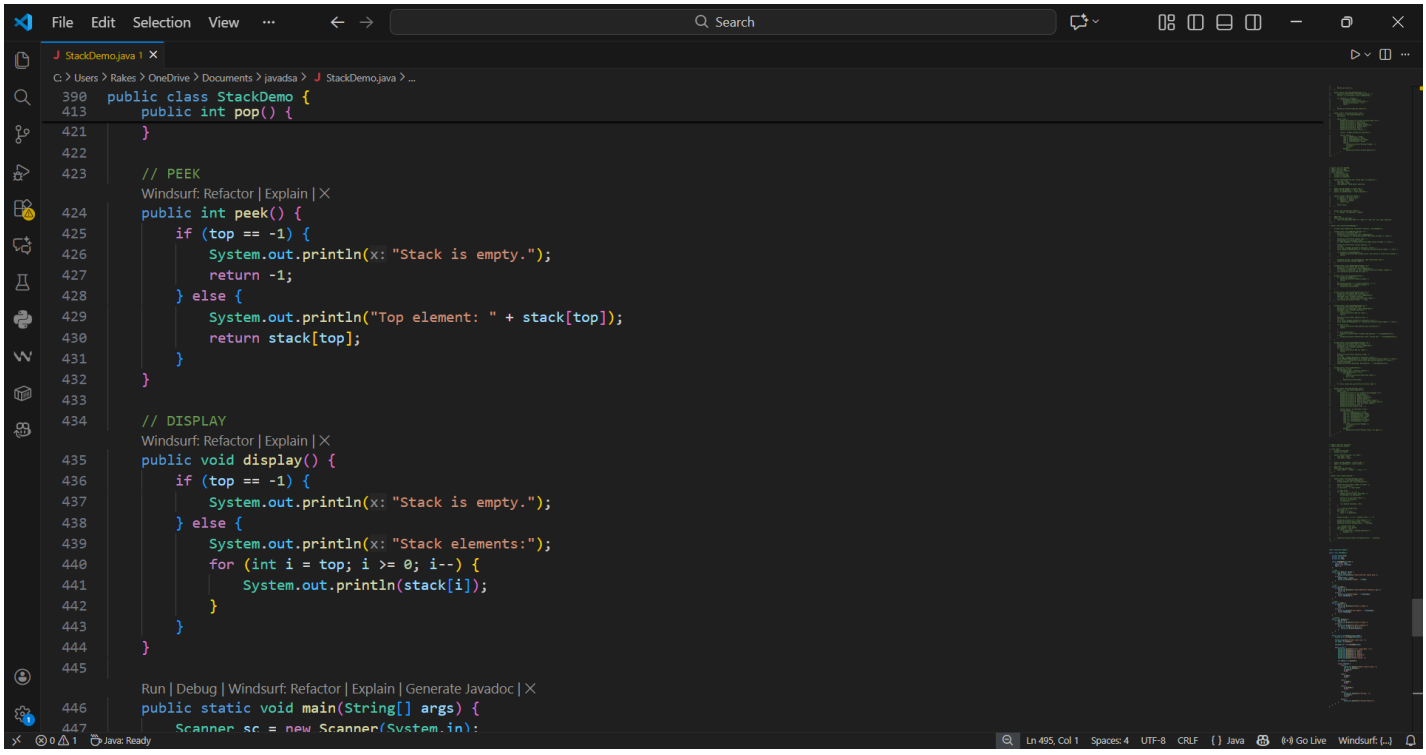

File Edit Selection View
Search
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac InventorySystem.java } ; if ($?) { java InventorySystem }
Enter number of items: 6
Enter Item Name: gg
Enter Stock: 5
Enter Item Name: mgm
Enter Stock:
Java: Ready
Ln 383, Col 1 Spaces: 4 UTF-8 CRLF { } Java Go Live

TASK 4: STACK OPERATIONS: {push,pop,peek,display}

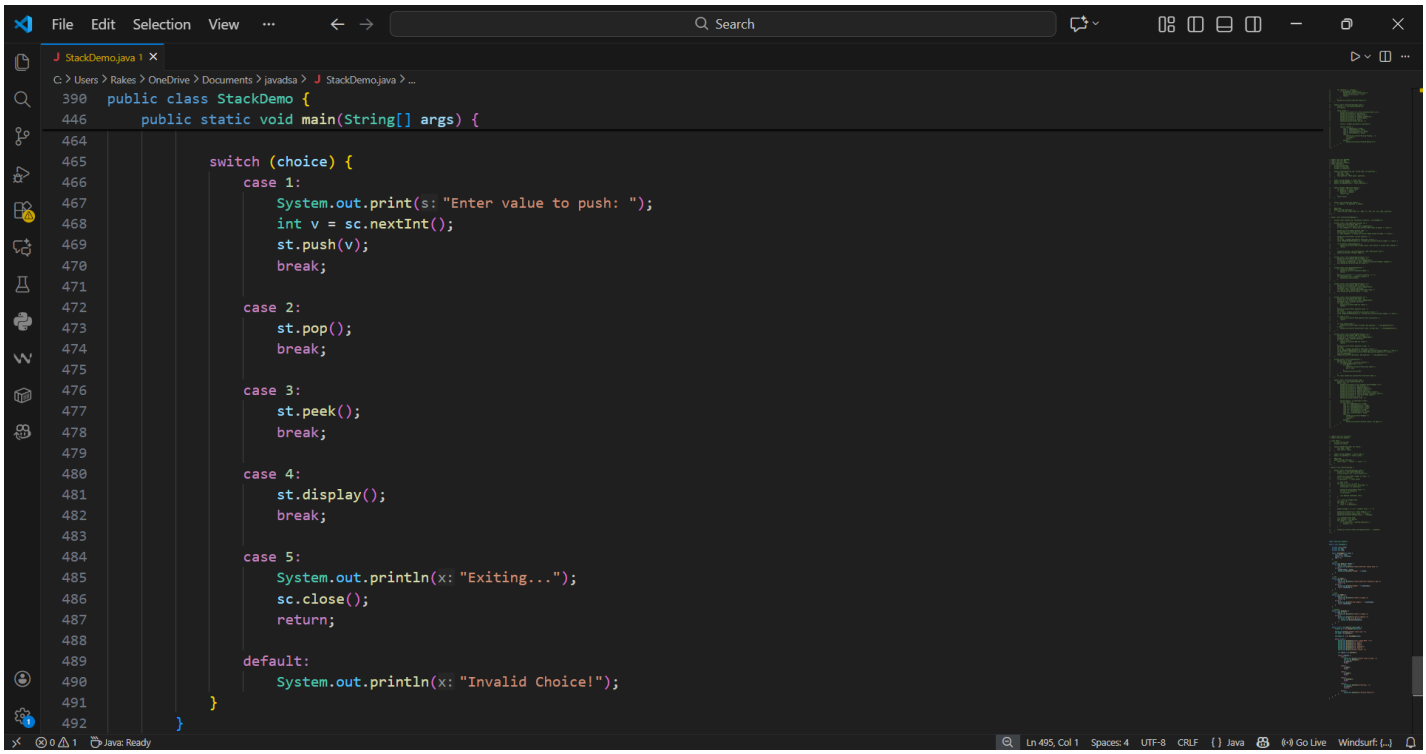
CODE:



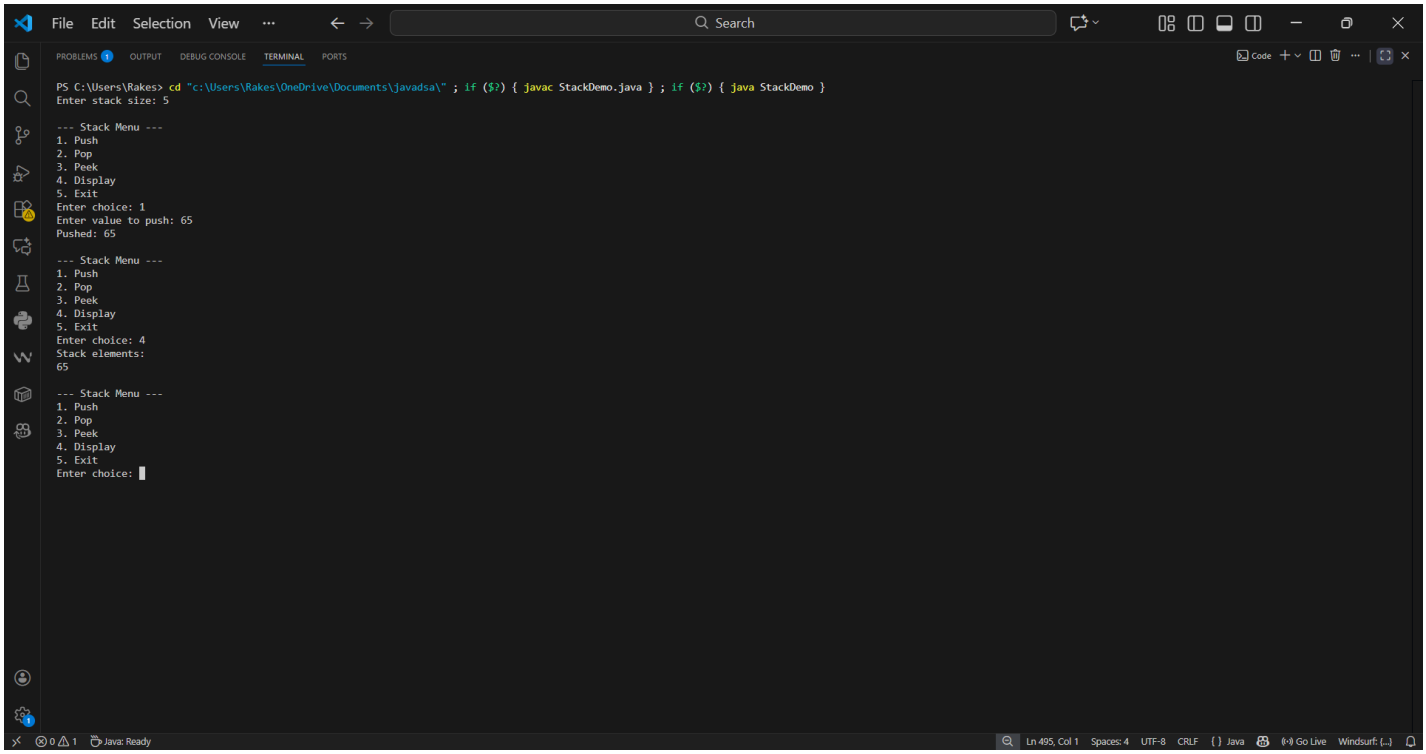
```
import java.util.Scanner;

Windsurf: Refactor | Explain
public class StackDemo {

    private int[] stack;
    private int top;
    private int size;

    public StackDemo(int size) {
        this.size = size;
        stack = new int[size];
        top = -1;
    }

    // PUSH
    Windsurf: Refactor | Explain |
    public void push(int value) {
        if (top == size - 1) {
            System.out.println(x: "Stack Overflow! Cannot push.");
        } else {
            stack[++top] = value;
            System.out.println("Pushed: " + value);
        }
    }

    // POP
    Windsurf: Refactor | Explain |
    public int pop() {
        if (top == -1) {
            System.out.println(x: "Stack Underflow! Nothing to pop.");
```



```
public class StackDemo {
    public int pop() {
    }

    // PEEK
    Windsurf: Refactor | Explain |
    public int peek() {
        if (top == -1) {
            System.out.println(x: "Stack is empty.");
            return -1;
        } else {
            System.out.println("Top element: " + stack[top]);
            return stack[top];
        }
    }

    // DISPLAY
    Windsurf: Refactor | Explain |
    public void display() {
        if (top == -1) {
            System.out.println(x: "Stack is empty.");
        } else {
            System.out.println(x: "Stack elements:");
            for (int i = top; i >= 0; i--) {
                System.out.println(stack[i]);
            }
        }
    }

    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

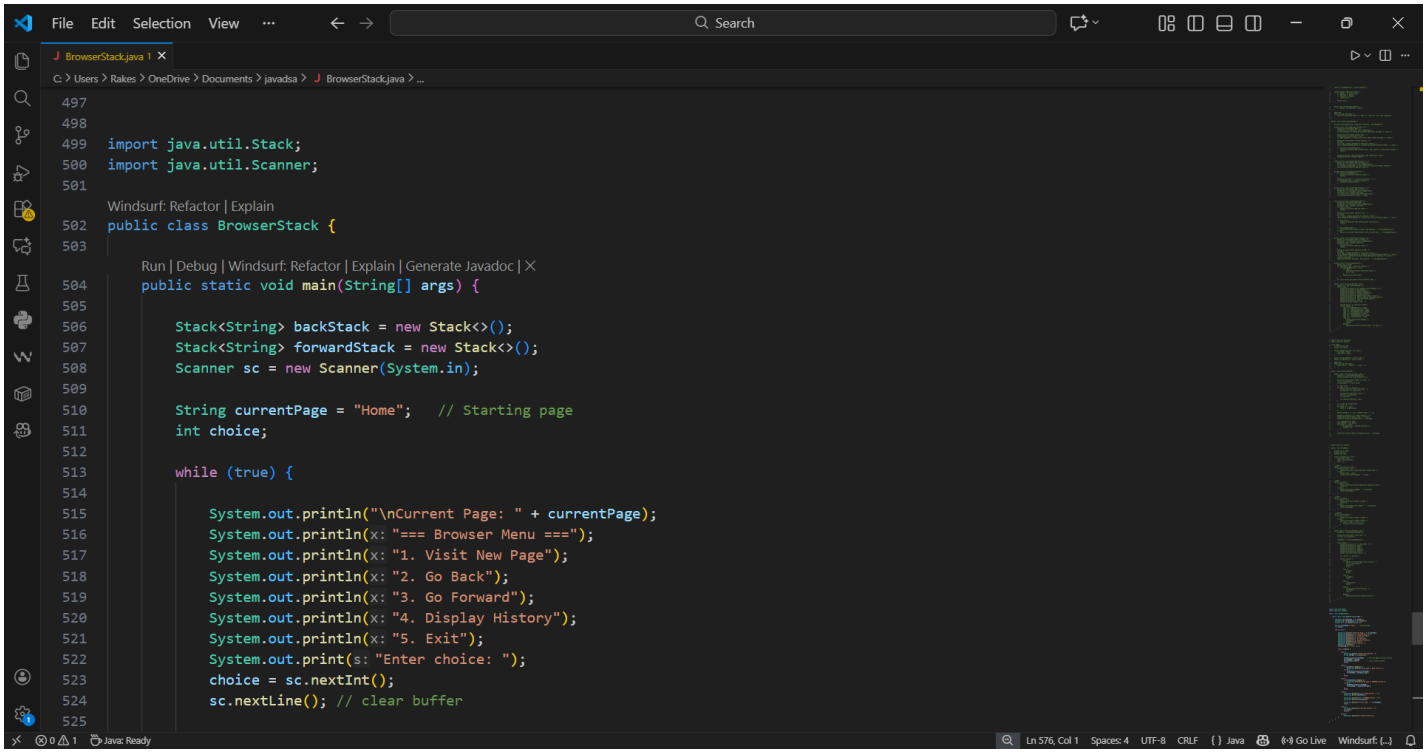

File Edit Selection View ···
Search
StackDemo.java 1
C: > Users > Rakes > OneDrive > Documents > javadsa > StackDemo.java > ...
public class StackDemo {
public static void main(String[] args) {
switch (choice) {
case 1:
System.out.print(s: "Enter value to push: ");
int v = sc.nextInt();
st.push(v);
break;
case 2:
st.pop();
break;
case 3:
st.peek();
break;
case 4:
st.display();
break;
case 5:
System.out.println(x: "Exiting...");
sc.close();
return;
default:
System.out.println(x: "Invalid Choice!");
}
}
Java: Ready
Ln 495, Col 1
Spaces: 4
UTF-8
CRLF
Java
Go Live

Output:

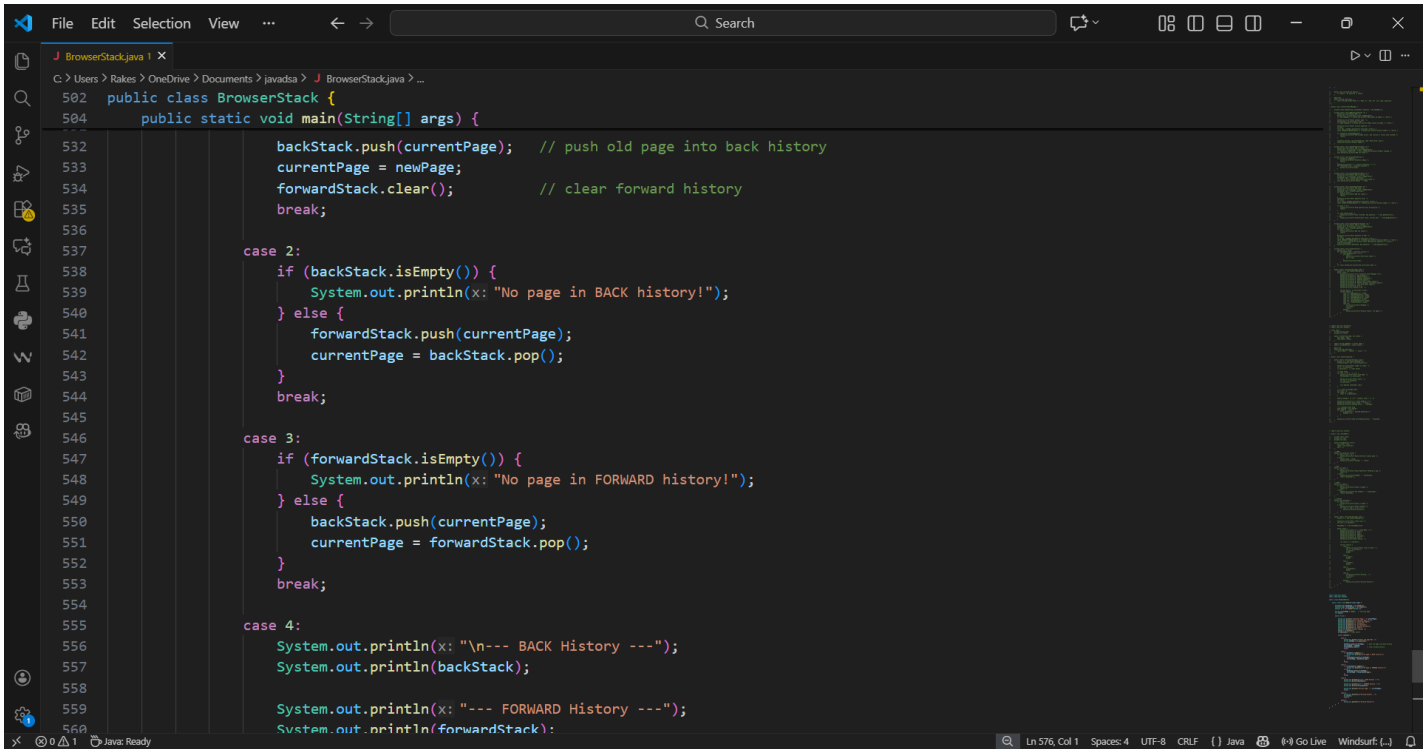

File Edit Selection View
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac StackDemo.java } ; if ($?) { java StackDemo }
Enter stack size: 5
--- Stack Menu ---
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 1
Enter value to push: 65
Pushed: 65
--- Stack Menu ---
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice: 4
Stack elements:
65
--- Stack Menu ---
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter choice:
Java: Ready
Ln 495, Col 1 Spaces: 4 UTF-8 CRLF Java Go Live

TASK-5: Implementation of stack browser:

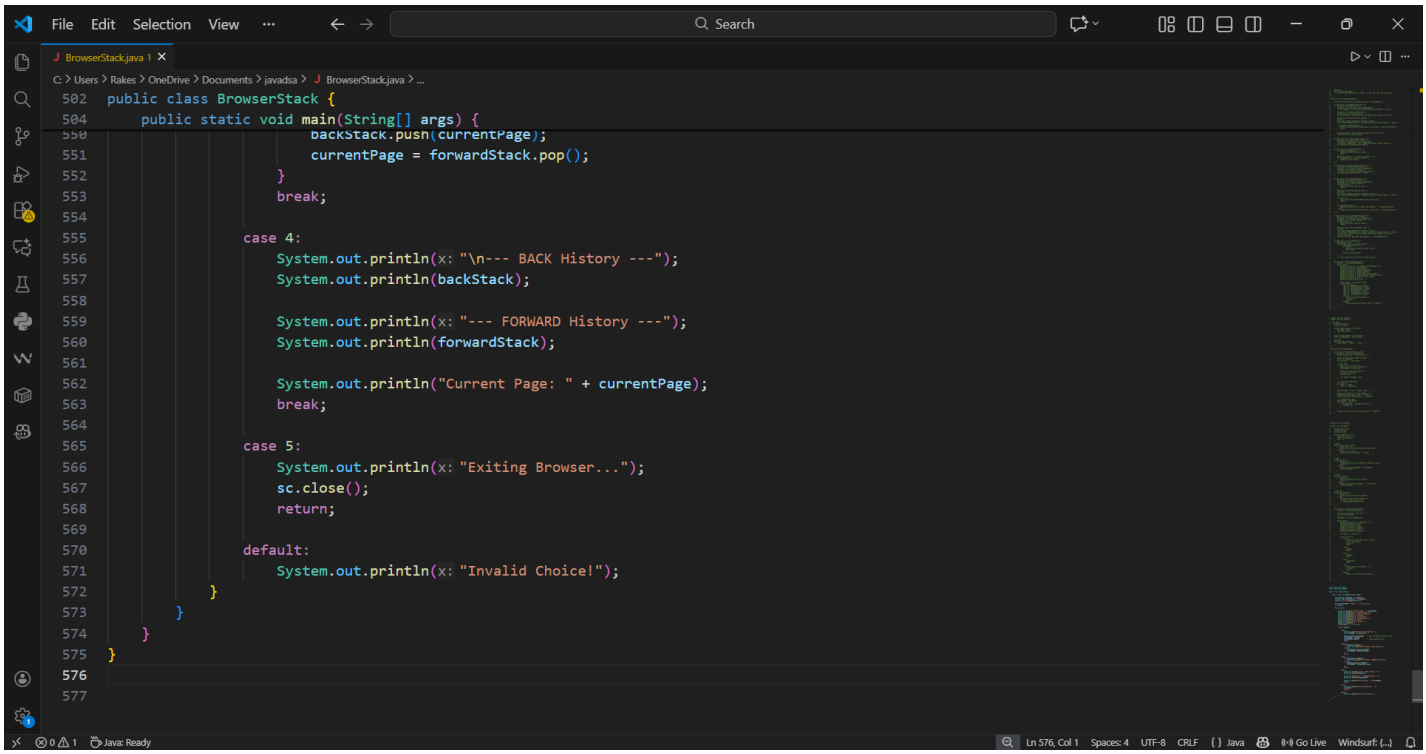
Code:



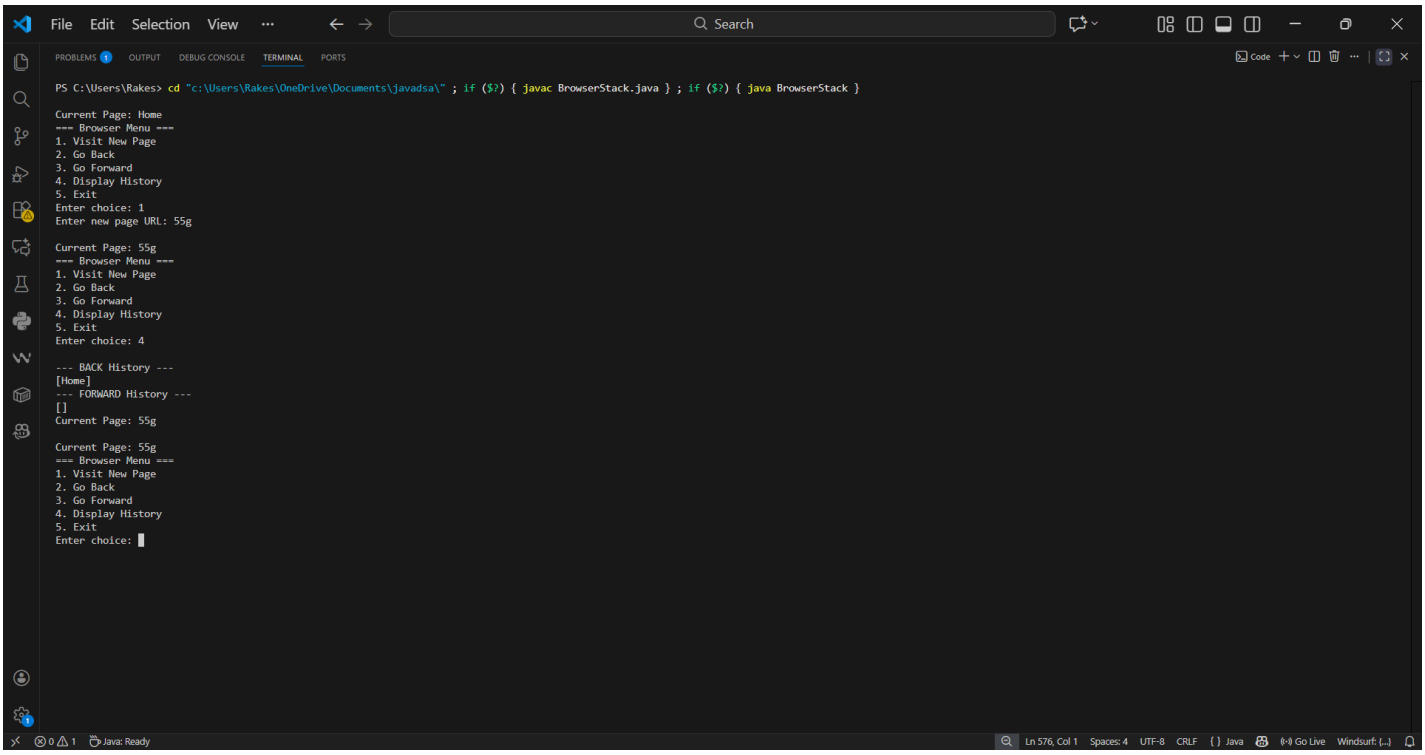
```
import java.util.Stack;
import java.util.Scanner;

Windsurf: Refactor | Explain
public class BrowserStack {

    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
    public static void main(String[] args) {

        Stack<String> backStack = new Stack<>();
        Stack<String> forwardStack = new Stack<>();
        Scanner sc = new Scanner(System.in);

        String currentPage = "Home";   // Starting page
        int choice;

        while (true) {

            System.out.println("\nCurrent Page: " + currentPage);
            System.out.println("=== Browser Menu ===");
            System.out.println("1. Visit New Page");
            System.out.println("2. Go Back");
            System.out.println("3. Go Forward");
            System.out.println("4. Display History");
            System.out.println("5. Exit");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();
            sc.nextLine(); // clear buffer
```



```
public class BrowserStack {
    public static void main(String[] args) {
                    backStack.push(currentPage);   // push old page into back history
                    currentPage = newPage;
                    forwardStack.clear();          // clear forward history
                    break;

                case 2:
                    if (backStack.isEmpty()) {
                        System.out.println("No page in BACK history!");
                    } else {
                        forwardStack.push(currentPage);
                        currentPage = backStack.pop();
                    }
                    break;

                case 3:
                    if (forwardStack.isEmpty()) {
                        System.out.println("No page in FORWARD history!");
                    } else {
                        backStack.push(currentPage);
                        currentPage = forwardStack.pop();
                    }
                    break;

                case 4:
                    System.out.println("\n--- BACK History ---");
                    System.out.println(backStack);

                    System.out.println("--- FORWARD History ---");
                    System.out.println(forwardStack);
```

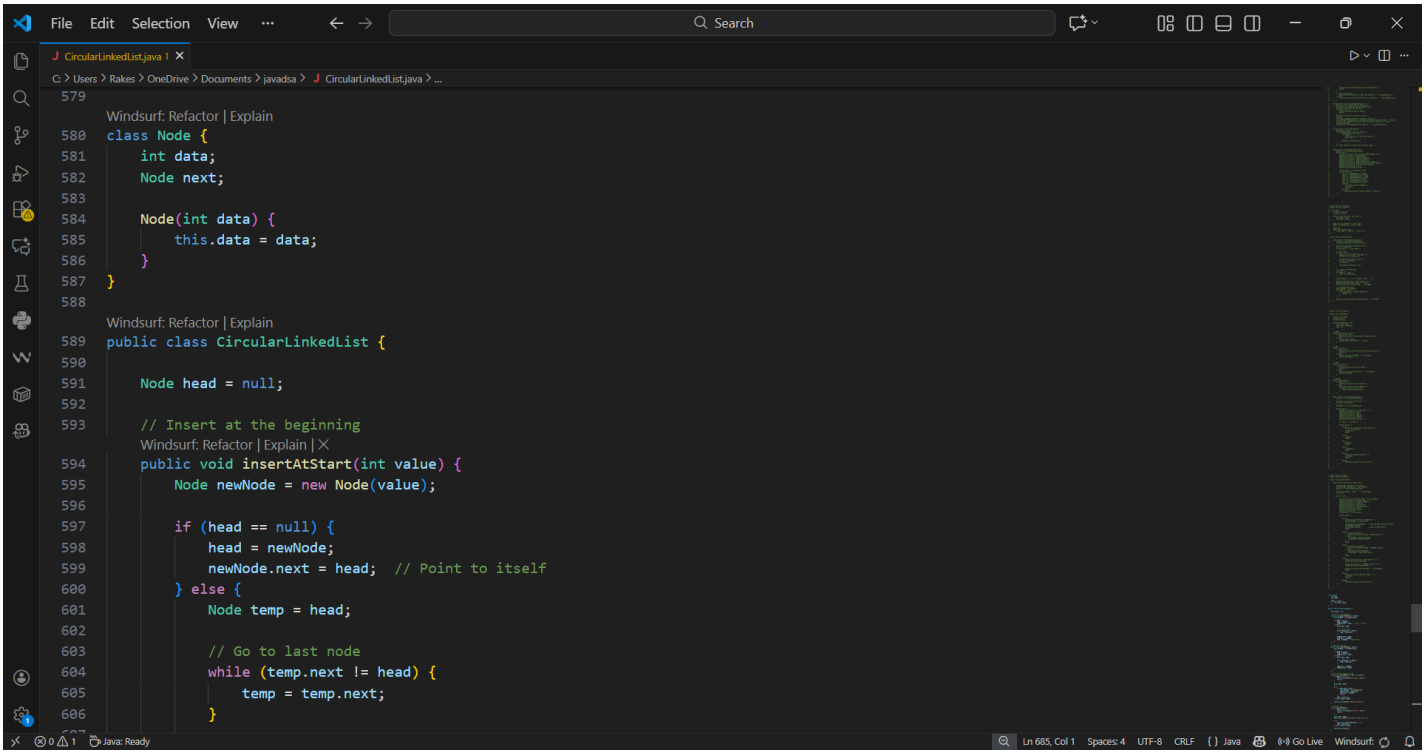

File Edit Selection View ···
Search
BrowserStack.java 1
C > Users > Rakes > OneDrive > Documents > javadsa > BrowserStack.java > ...
public class BrowserStack {
public static void main(String[] args) {
backStack.push(currentPage);
currentPage = forwardStack.pop();
}
break;
case 4:
System.out.println(x: "\n--- BACK History ---");
System.out.println(backStack);
System.out.println(x: "--- FORWARD History ---");
System.out.println(forwardStack);
System.out.println("Current Page: " + currentPage);
break;
case 5:
System.out.println(x: "Exiting Browser...");
sc.close();
return;
default:
System.out.println(x: "Invalid Choice!");
}
}
}
Java: Ready
Ln 576, Col 1 Spaces: 4 UTF-8 CRLF { } Java Go Live Windsurf: (...)

Output:

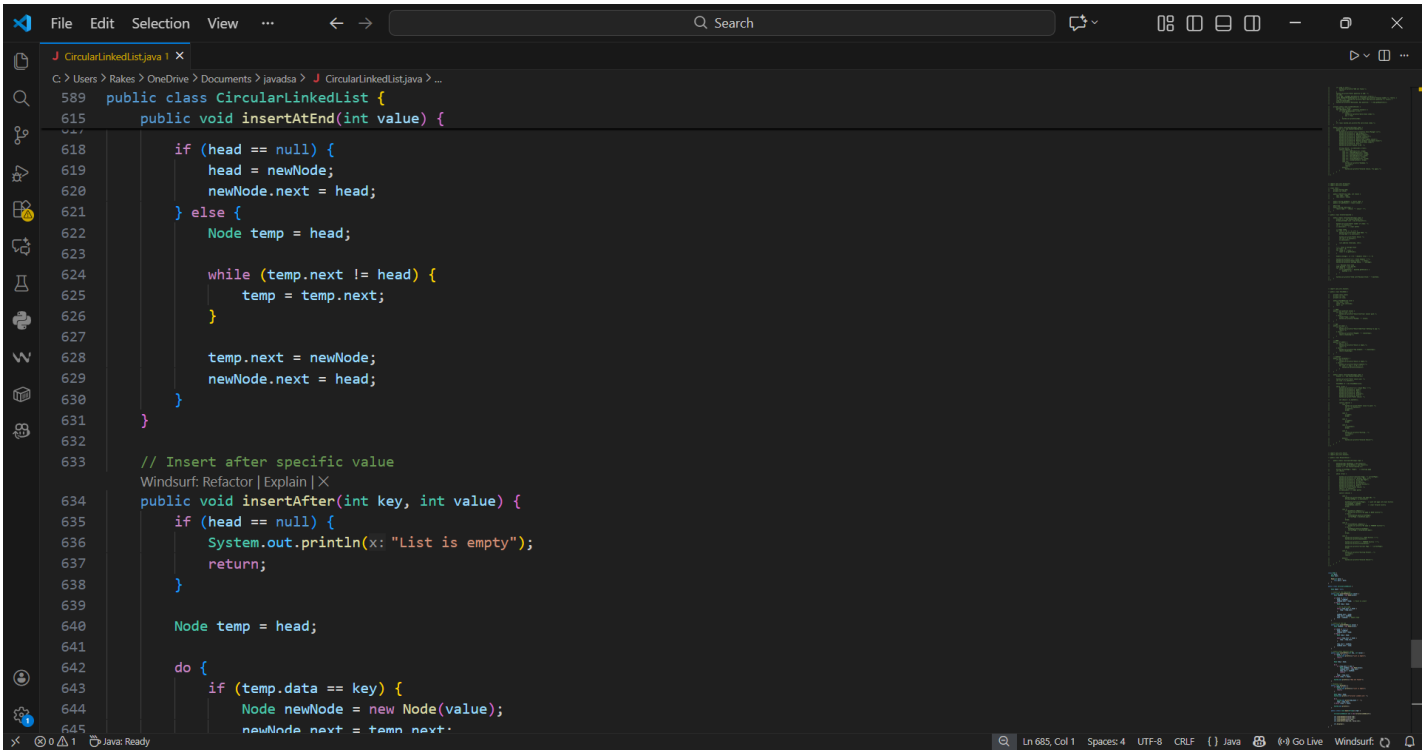

File Edit Selection View
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac BrowserStack.java } ; if ($?) { java BrowserStack }
Current Page: Home
=== Browser Menu ===
1. Visit New Page
2. Go Back
3. Go Forward
4. Display History
5. Exit
Enter choice: 1
Enter new page URL: 55g
Current Page: 55g
=== Browser Menu ===
1. Visit New Page
2. Go Back
3. Go Forward
4. Display History
5. Exit
Enter choice: 4
--- BACK History ---
[Home]
--- FORWARD History ---
[]
Current Page: 55g
Current Page: 55g
=== Browser Menu ===
1. Visit New Page
2. Go Back
3. Go Forward
4. Display History
5. Exit
Enter choice:
Java: Ready
Ln 576, Col 1 Spaces: 4 UTF-8 CRLF Java Go Live

TASK-6: Insertion in circular linkedlist

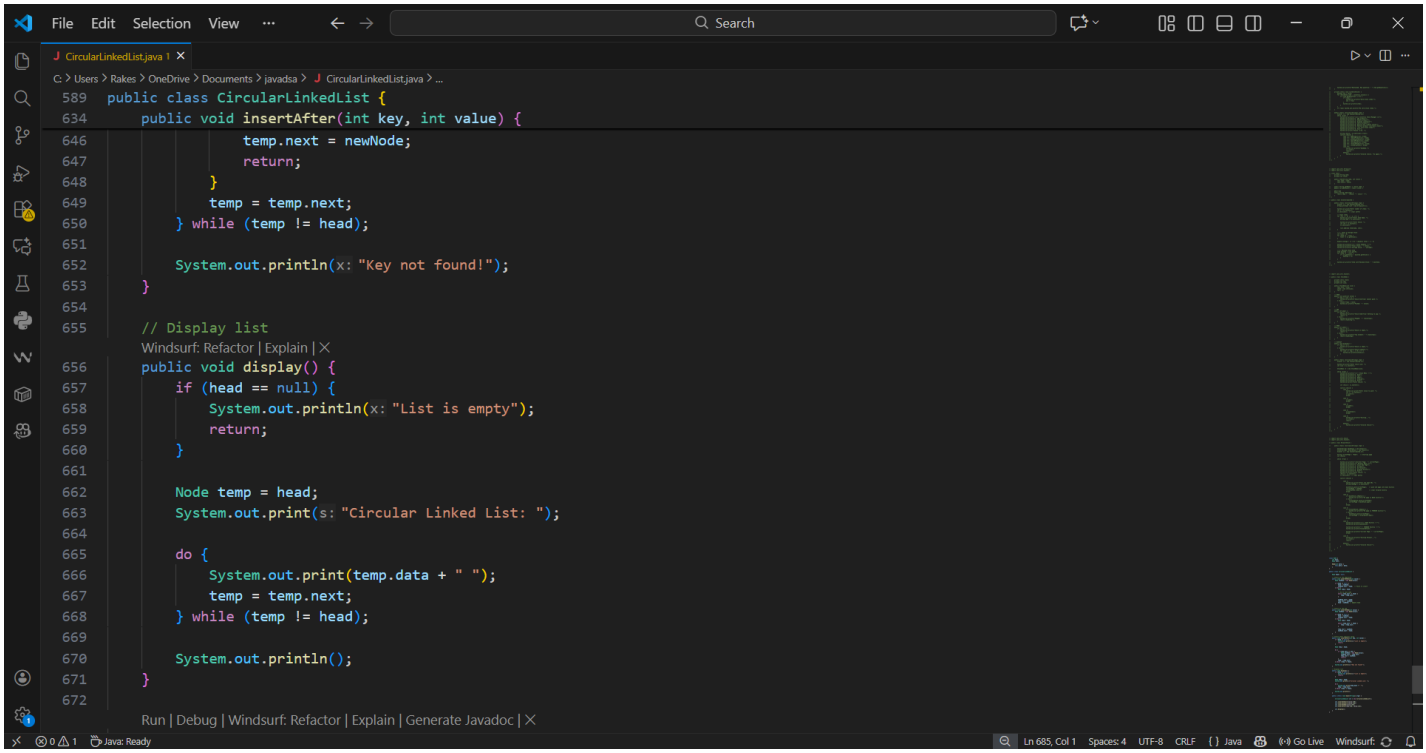
CODE:

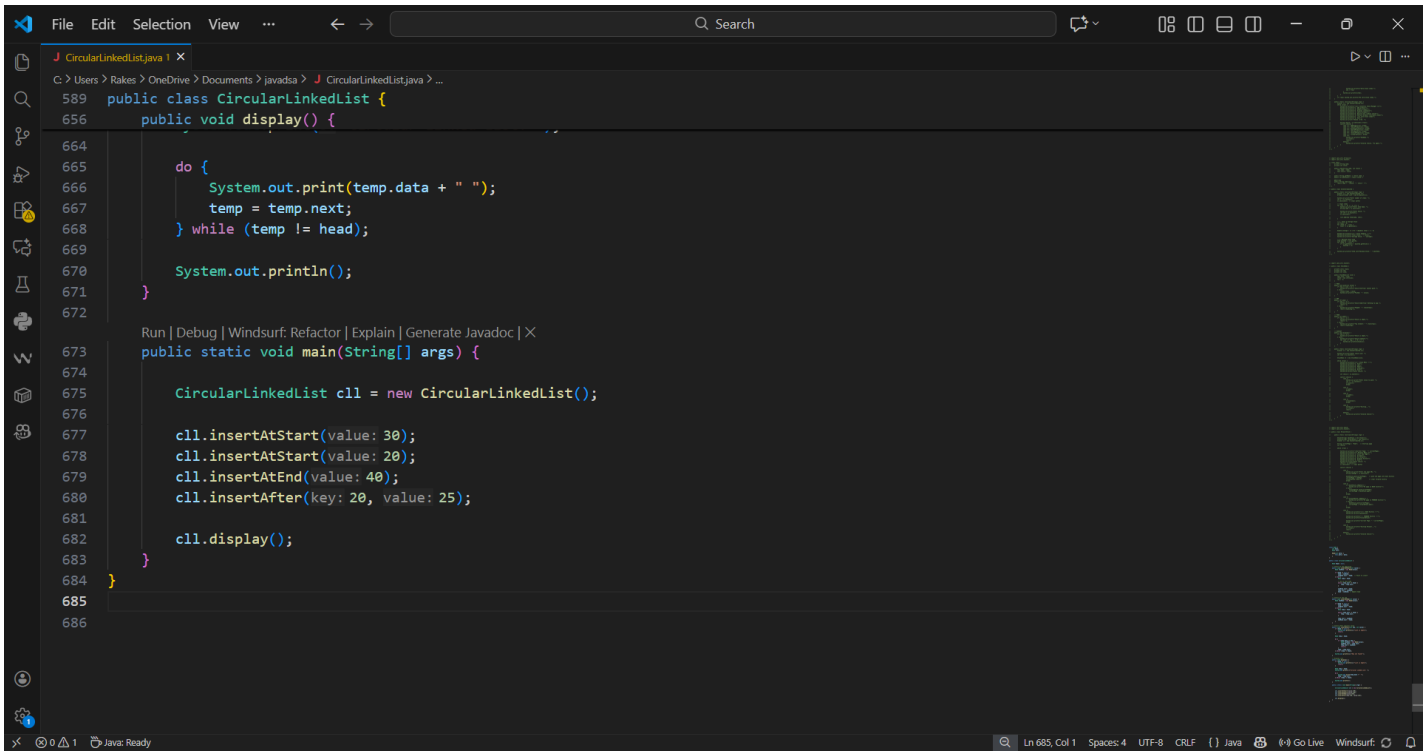


```
Windsurf: Refactor | Explain
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
    }
}

Windsurf: Refactor | Explain
public class CircularLinkedList {

    Node head = null;

    // Insert at the beginning
    Windsurf: Refactor | Explain |
    public void insertAtStart(int value) {
        Node newNode = new Node(value);

        if (head == null) {
            head = newNode;
            newNode.next = head;  // Point to itself
        } else {
            Node temp = head;

            // Go to last node
            while (temp.next != head) {
                temp = temp.next;
            }
```

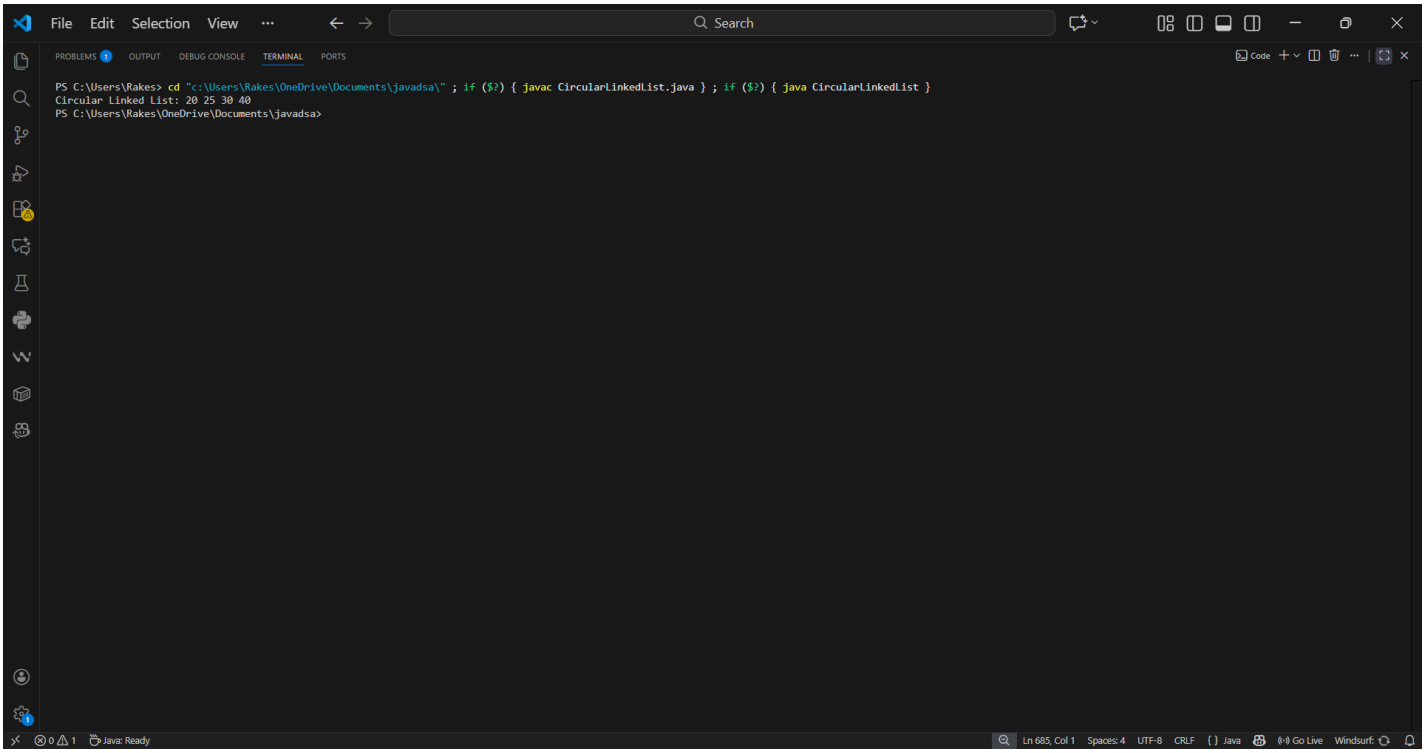


```
public class CircularLinkedList {
    public void insertAtEnd(int value) {

        if (head == null) {
            head = newNode;
            newNode.next = head;
        } else {
            Node temp = head;

            while (temp.next != head) {
                temp = temp.next;
            }

            temp.next = newNode;
            newNode.next = head;
        }
    }

    // Insert after specific value
    Windsurf: Refactor | Explain |
    public void insertAfter(int key, int value) {
        if (head == null) {
            System.out.println(x: "List is empty");
            return;
        }

        Node temp = head;

        do {
            if (temp.data == key) {
                Node newNode = new Node(value);
```


File Edit Selection View ···
Search
CircularLinkedList.java 1
C: > Users > Rakes > OneDrive > Documents > javadsa > CircularLinkedList.java > ...
public class CircularLinkedList {
public void insertAfter(int key, int value) {
temp.next = newNode;
return;
}
temp = temp.next;
} while (temp != head);
System.out.println(x: "Key not found!");
}
// Display list
Windsurf: Refactor | Explain |
public void display() {
if (head == null) {
System.out.println(x: "List is empty");
return;
}
Node temp = head;
System.out.print(s: "Circular Linked List: ");
do {
System.out.print(temp.data + " ");
temp = temp.next;
} while (temp != head);
System.out.println();
}
Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
Java: Ready
Ln 685, Col 1 Spaces: 4 UTF-8 CRLF { } Java Go Live Windsurf:


File Edit Selection View ···
Search
CircularLinkedList.java 1
C: > Users > Rakes > OneDrive > Documents > javadsa > CircularLinkedList.java > ...
public class CircularLinkedList {
public void display() {
do {
System.out.print(temp.data + " ");
temp = temp.next;
} while (temp != head);
System.out.println();
}
Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
public static void main(String[] args) {
CircularLinkedList cll = new CircularLinkedList();
cll.insertAtStart(value: 30);
cll.insertAtStart(value: 20);
cll.insertAtEnd(value: 40);
cll.insertAfter(key: 20, value: 25);
cll.display();
}
}
Java: Ready
Ln 685, Col 1 Spaces: 4 UTF-8 CRLF { } Java Go Live Windsurf:

OUTPUT:


File Edit Selection View
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac CircularLinkedList.java } ; if ($?) { java CircularLinkedList }
Circular Linked List: 20 25 30 40
PS C:\Users\Rakes\OneDrive\Documents\javadsa>
0 1 Java: Ready
Ln 685, Col 1 Spaces: 4 UTF-8 CRLF { } Java Go Live Windsurf:

TASK-7:

CODE:

```
class CNodeDel:
    def __init__(self, value):
        self.data = value
        self.next = None
class CircularListDelete:
    def __init__(self):
        self.head = None
    def append(self, value):
        new_node = CNodeDel(value)
        if self.head is None:
            self.head = new_node
            new_node.next = self.head
            return
        temp = self.head
        while temp.next != self.head:
            temp = temp.next
        temp.next = new_node
        new_node.next = self.head
    def display(self):
        if self.head is None:
            print("List is empty")
            return
        print("Circular Linked List:", end=" ")
        temp = self.head
        while True:
            print(temp.data, end=" -> ")
            temp = temp.next
            if temp == self.head:
                break
        print("(back to head)")
    def delete_begin(self):
        if self.head is None:
            print("List is empty, nothing to delete.")
            return
        if self.head.next == self.head:
            self.head = None
            print("Deleted the only node in the list.")
            return
        last = self.head
        while last.next != self.head:
            last = last.next
        self.head = self.head.next
        last.next = self.head
        print("Node deleted from beginning.")
    def delete_end(self):
        if self.head is None:
            print("List is empty, nothing to delete.")
            return
```

```
        if self.head.next == self.head:
            self.head = None
            print("Deleted the only node in the list.")
            return
        prev = None
        temp = self.head
        while temp.next != self.head:
            prev = temp
            temp = temp.next
        prev.next = self.head
        print("Node deleted from end.")
    def delete_at_pos(self, pos):
        if self.head is None:
            print("List is empty, nothing to delete.")
            return
        if pos == 1:
            self.delete_begin()
            return
        prev = None
        temp = self.head
        count = 1
        while count < pos and temp.next != self.head:
            prev = temp
            temp = temp.next
```

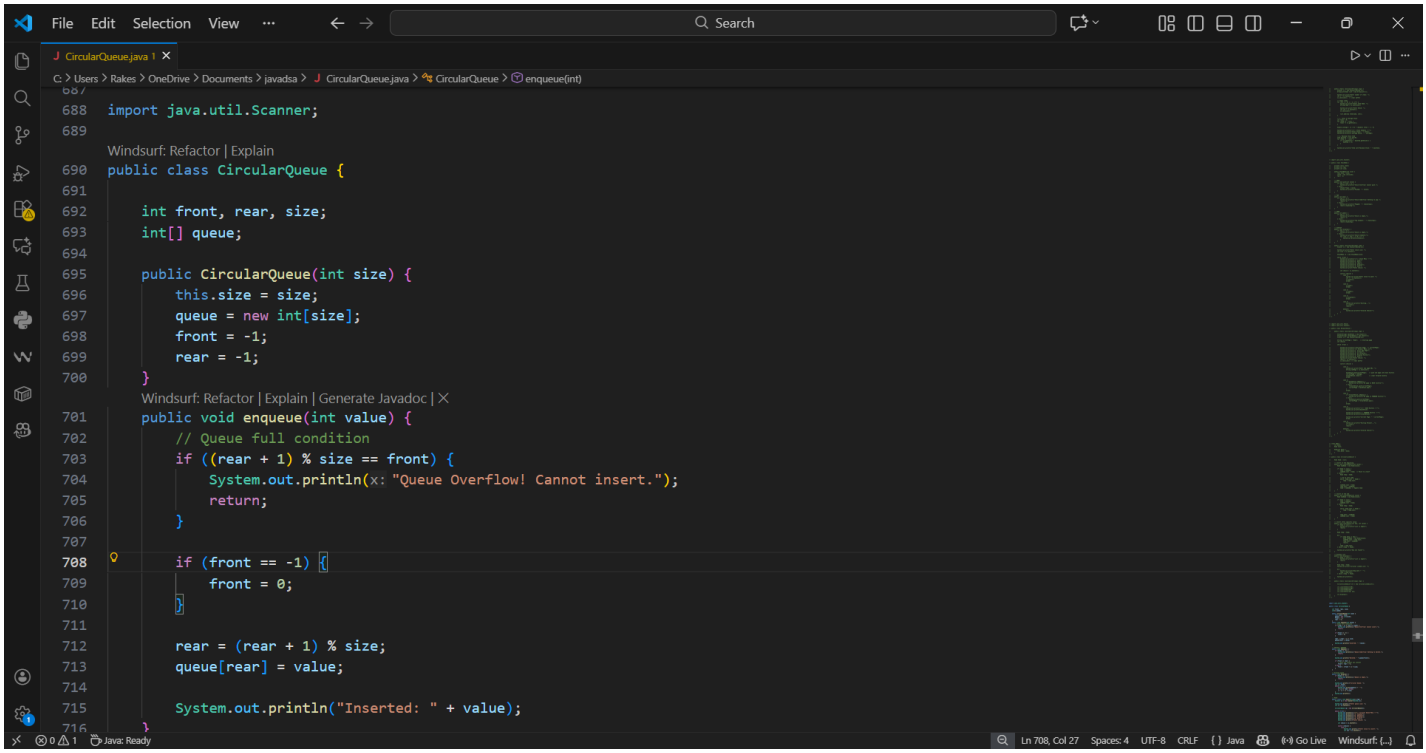
```
            count += 1
        if count != pos:
            print("Position out of range.")
            return
        prev.next = temp.next
        print(f"Node deleted from position {pos}.")
cl_del = CircularListDelete()
cl_del.append(10)
cl_del.append(20)
cl_del.append(30)
cl_del.append(40)
print("Initial List:")
cl_del.display()
cl_del.delete_begin()
cl_del.display()
cl_del.delete_end()
cl_del.display()
cl_del.delete_at_pos(2)
cl_del.display()
```

OUTPUT:

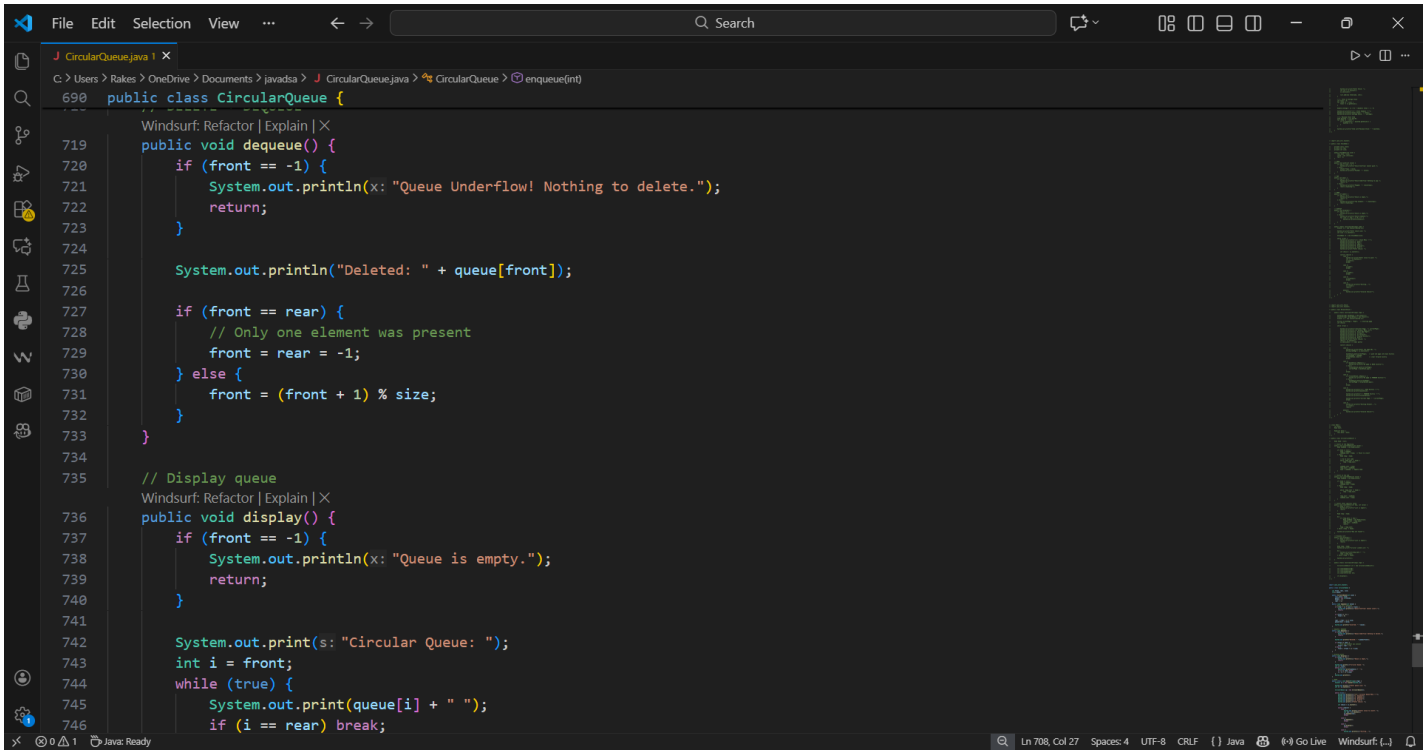
```
Initial List:
Circular Linked List: 10 -> 20 -> 30 -> 40 -> (back to head)
Node deleted from beginning.
Circular Linked List: 20 -> 30 -> 40 -> (back to head)
Node deleted from end.
Circular Linked List: 20 -> 30 -> (back to head)
Node deleted from position 2.
Circular Linked List: 20 -> (back to head)

=== Code Execution Successful ===
```

TASK 8: Circular queue insertion and deletion

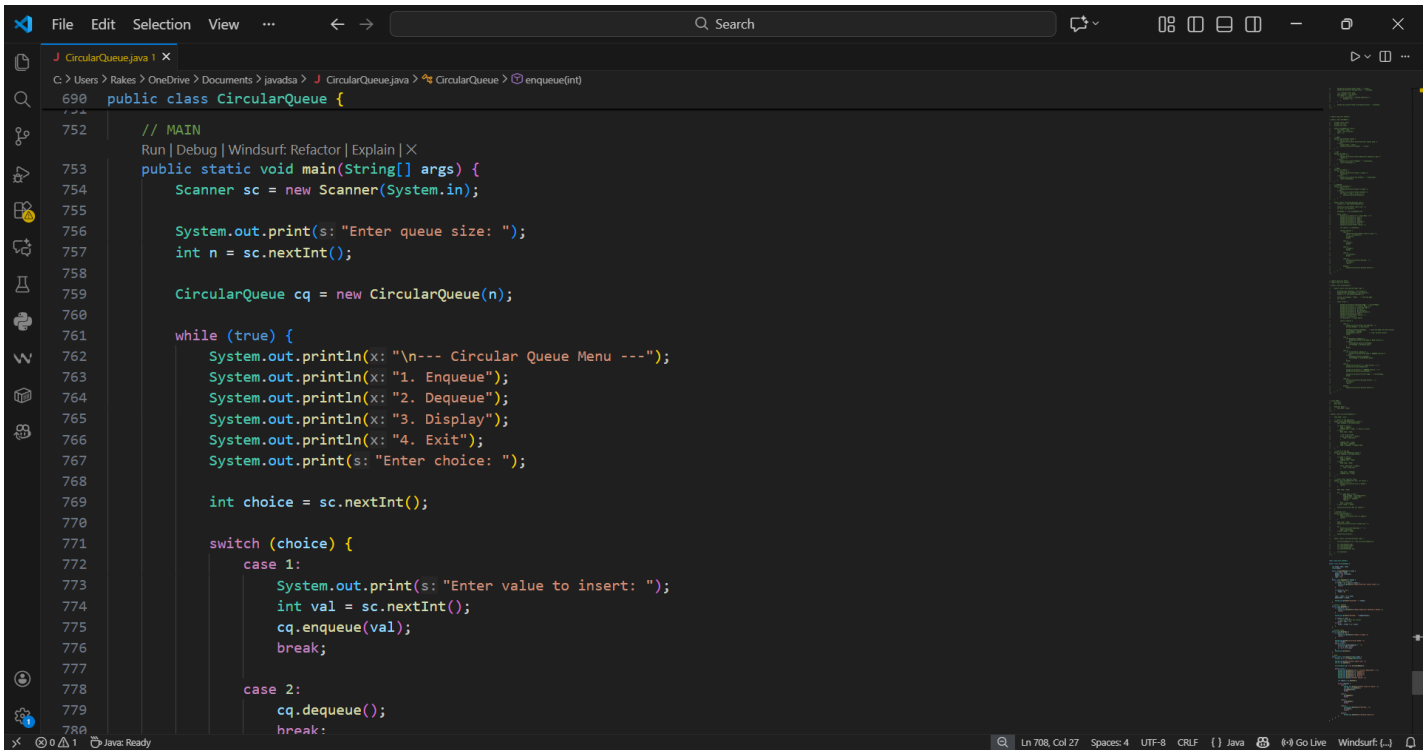
CODE:



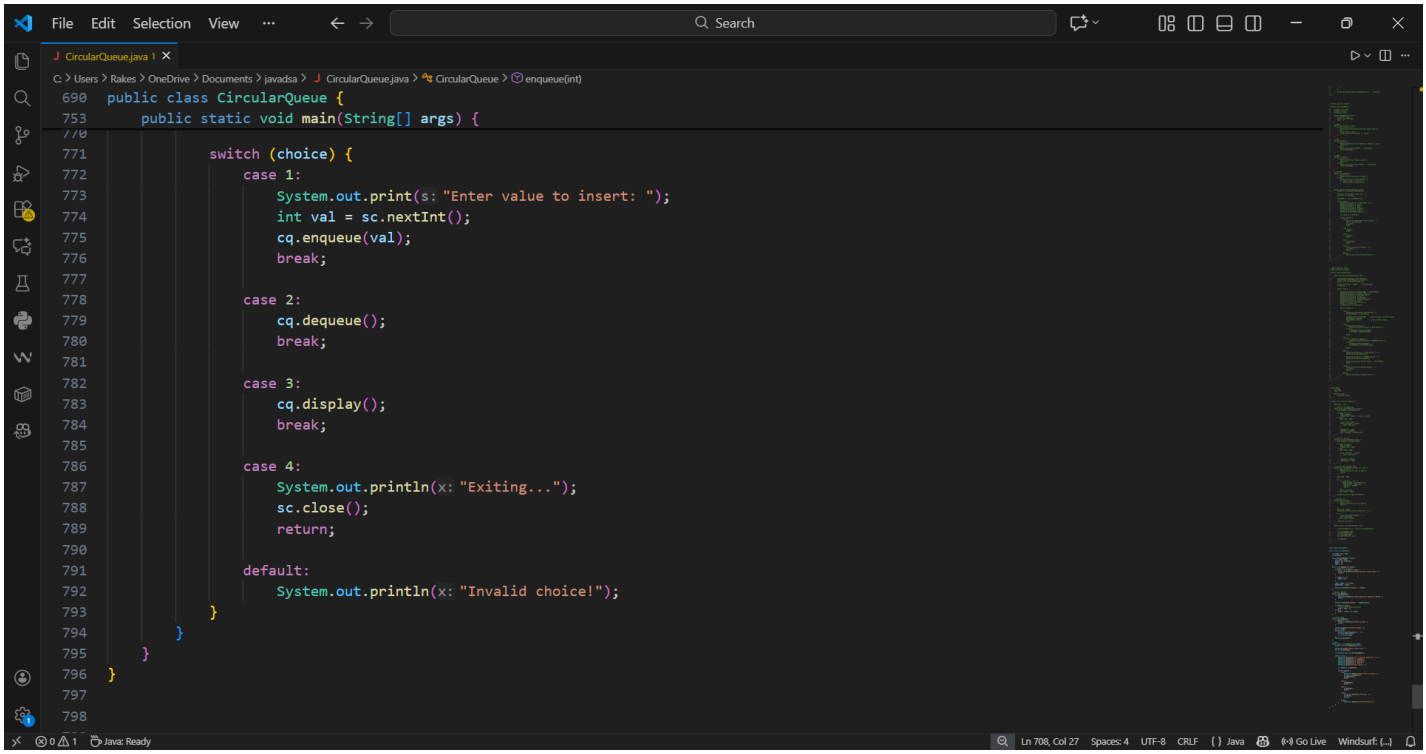
```
import java.util.Scanner;

Windsurf: Refactor | Explain
public class CircularQueue {

    int front, rear, size;
    int[] queue;

    public CircularQueue(int size) {
        this.size = size;
        queue = new int[size];
        front = -1;
        rear = -1;
    }
    Windsurf: Refactor | Explain | Generate Javadoc |
    public void enqueue(int value) {
        // Queue full condition
        if ((rear + 1) % size == front) {
            System.out.println(x: "Queue Overflow! Cannot insert.");
            return;
        }

        if (front == -1) {
            front = 0;
        }

        rear = (rear + 1) % size;
        queue[rear] = value;

        System.out.println("Inserted: " + value);
    }
```



```
public class CircularQueue {
    Windsurf: Refactor | Explain |
    public void dequeue() {
        if (front == -1) {
            System.out.println(x: "Queue Underflow! Nothing to delete.");
            return;
        }

        System.out.println("Deleted: " + queue[front]);

        if (front == rear) {
            // Only one element was present
            front = rear = -1;
        } else {
            front = (front + 1) % size;
        }
    }

    // Display queue
    Windsurf: Refactor | Explain |
    public void display() {
        if (front == -1) {
            System.out.println(x: "Queue is empty.");
            return;
        }

        System.out.print(s: "Circular Queue: ");
        int i = front;
        while (true) {
            System.out.print(queue[i] + " ");
            if (i == rear) break;
```

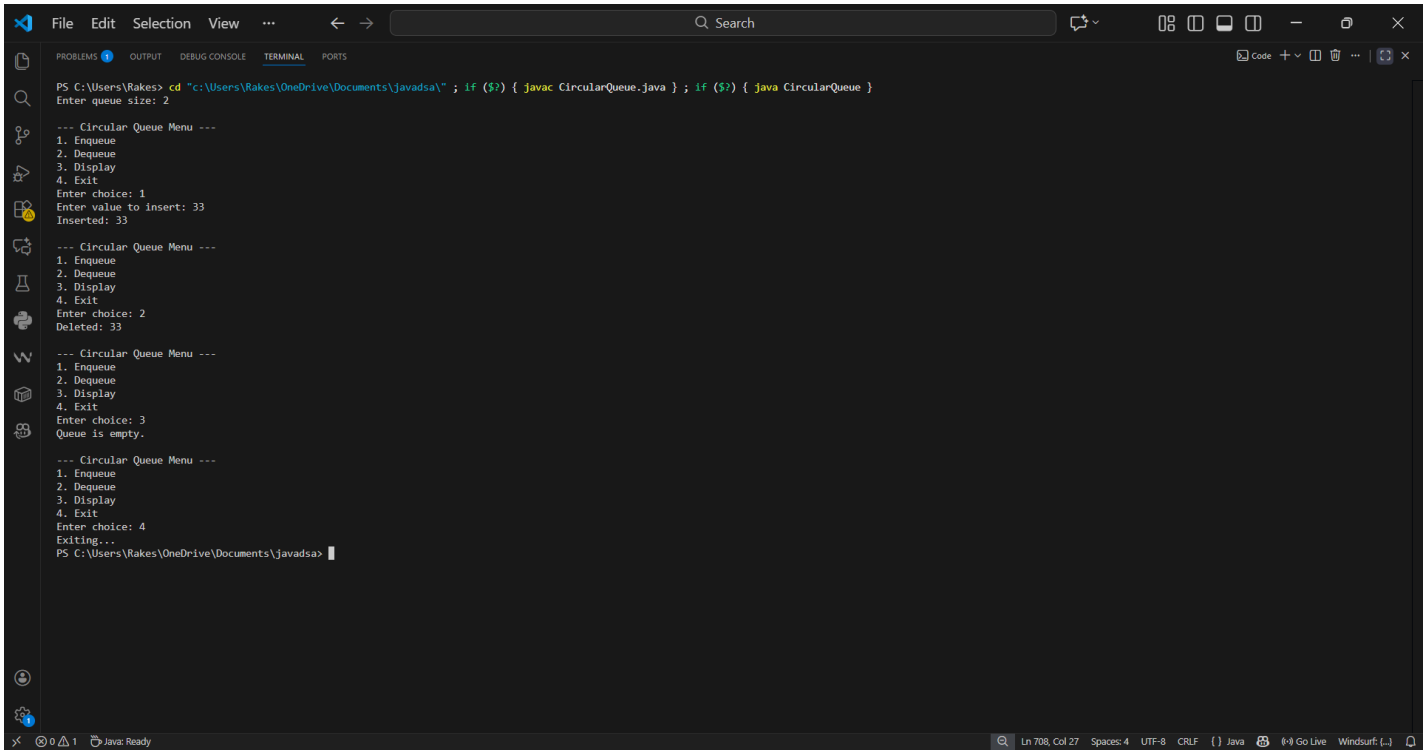


```
public class CircularQueue {

    // MAIN
    Run | Debug | Windsurf: Refactor | Explain | X
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print(s: "Enter queue size: ");
        int n = sc.nextInt();

        CircularQueue cq = new CircularQueue(n);

        while (true) {
            System.out.println(x: "\n--- Circular Queue Menu ---");
            System.out.println(x: "1. Enqueue");
            System.out.println(x: "2. Dequeue");
            System.out.println(x: "3. Display");
            System.out.println(x: "4. Exit");
            System.out.print(s: "Enter choice: ");

            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.print(s: "Enter value to insert: ");
                    int val = sc.nextInt();
                    cq.enqueue(val);
                    break;

                case 2:
                    cq.dequeue();
                    break;
```



```
public class CircularQueue {
    public static void main(String[] args) {

            switch (choice) {
                case 1:
                    System.out.print(s: "Enter value to insert: ");
                    int val = sc.nextInt();
                    cq.enqueue(val);
                    break;

                case 2:
                    cq.dequeue();
                    break;

                case 3:
                    cq.display();
                    break;

                case 4:
                    System.out.println(x: "Exiting...");
                    sc.close();
                    return;

                default:
                    System.out.println(x: "Invalid choice!");
            }
        }
    }
}
```

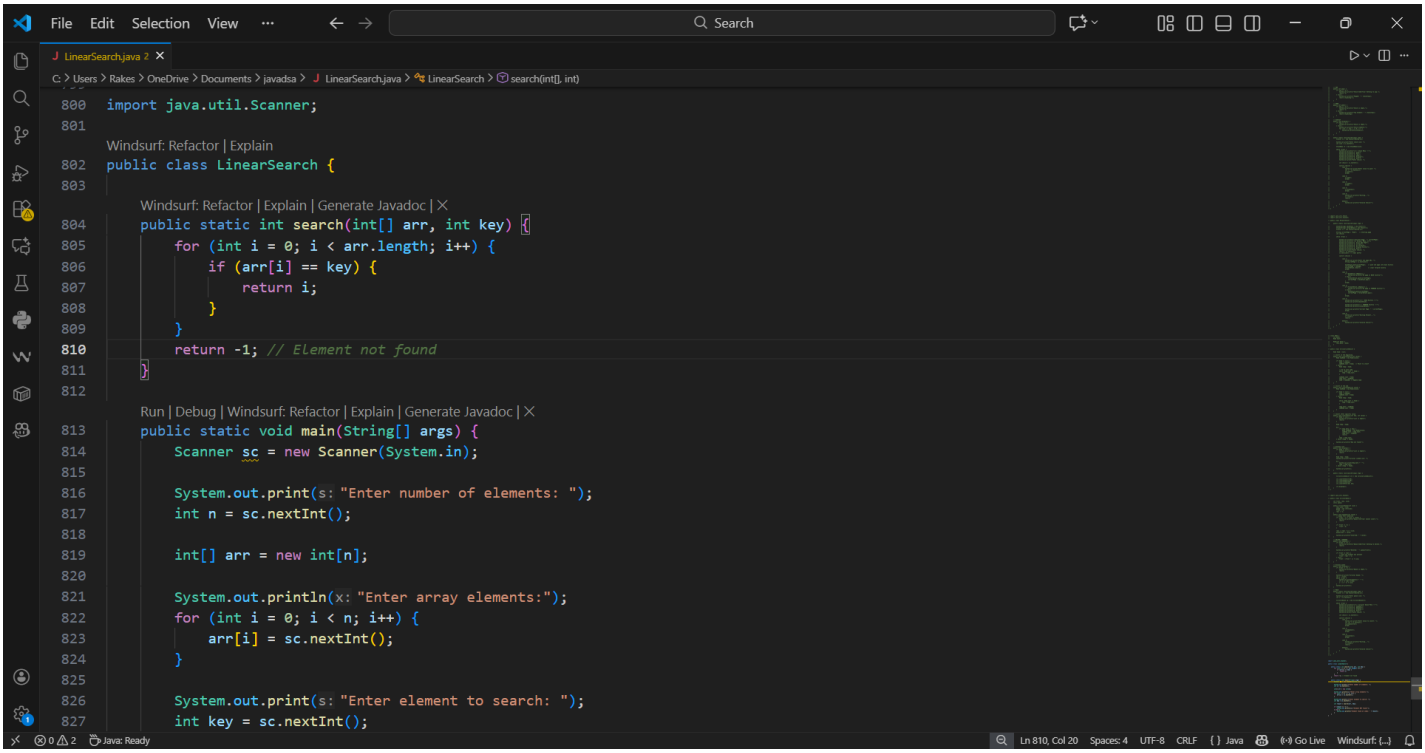
OUTPUT:


File Edit Selection View
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac CircularQueue.java } ; if ($?) { java CircularQueue }
Enter queue size: 2
--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 33
Inserted: 33
--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 2
Deleted: 33
--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 3
Queue is empty.
--- Circular Queue Menu ---
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter choice: 4
Exiting...
PS C:\Users\Rakes\OneDrive\Documents\javadsa>
Java: Ready
Ln 708, Col 27 Spaces: 4 UTF-8 CRLF { } Java Go Live Windsurf: (...)

TASK-9:

LINEAR SEARCH

CODE:



```java
import java.util.Scanner;

Windsurf: Refactor | Explain
public class LinearSearch {

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static int search(int[] arr, int key) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                return i;
            }
        }
        return -1; // Element not found
    }

    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print(s: "Enter number of elements: ");
        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println(x: "Enter array elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print(s: "Enter element to search: ");
        int key = sc.nextInt();
```

OUTPUT:

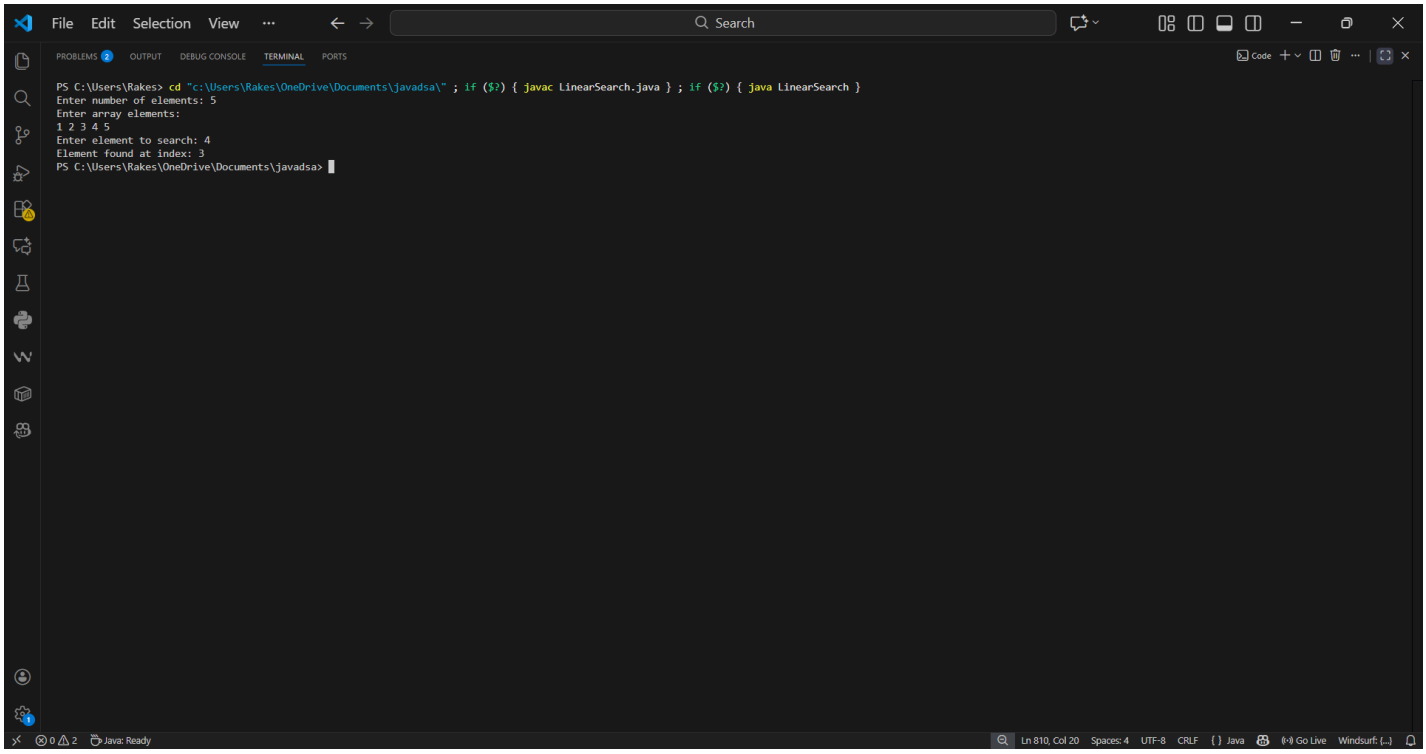
TASK-10:

BINARY SEARCH:

CODE:

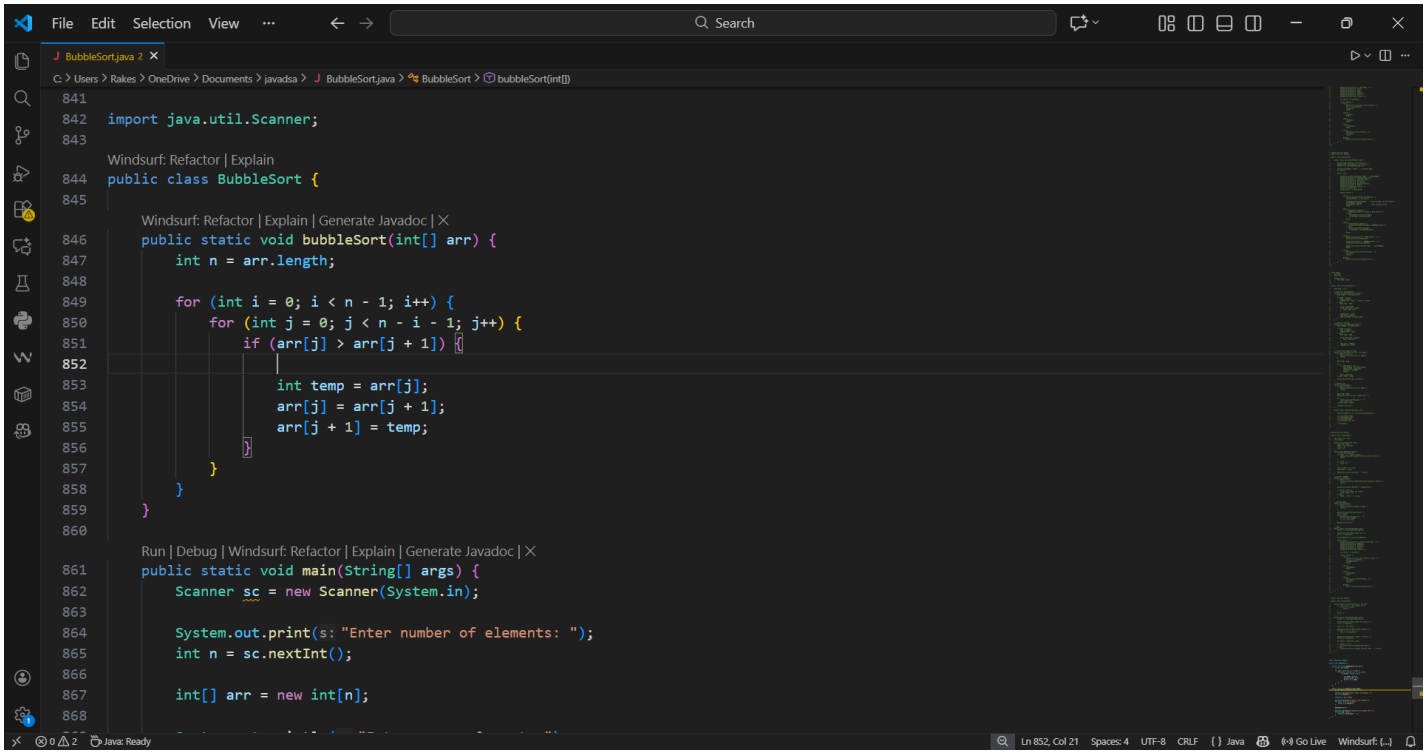
```python
import random

values = sorted(random.randint(1, 100) for _ in range(10))
print("Array:", values)

key = int(input("Enter element to search (binary): "))

low = 0
high = len(values) - 1
found_flag = False

while low <= high:
    mid = (low + high) // 2
    if values[mid] == key:
        print("Element found at index", mid)
        found_flag = True
        break
    elif values[mid] < key:
        low = mid + 1
    else:
        high = mid - 1

if not found_flag:
    print("Element not found")
```

OUTPUT:

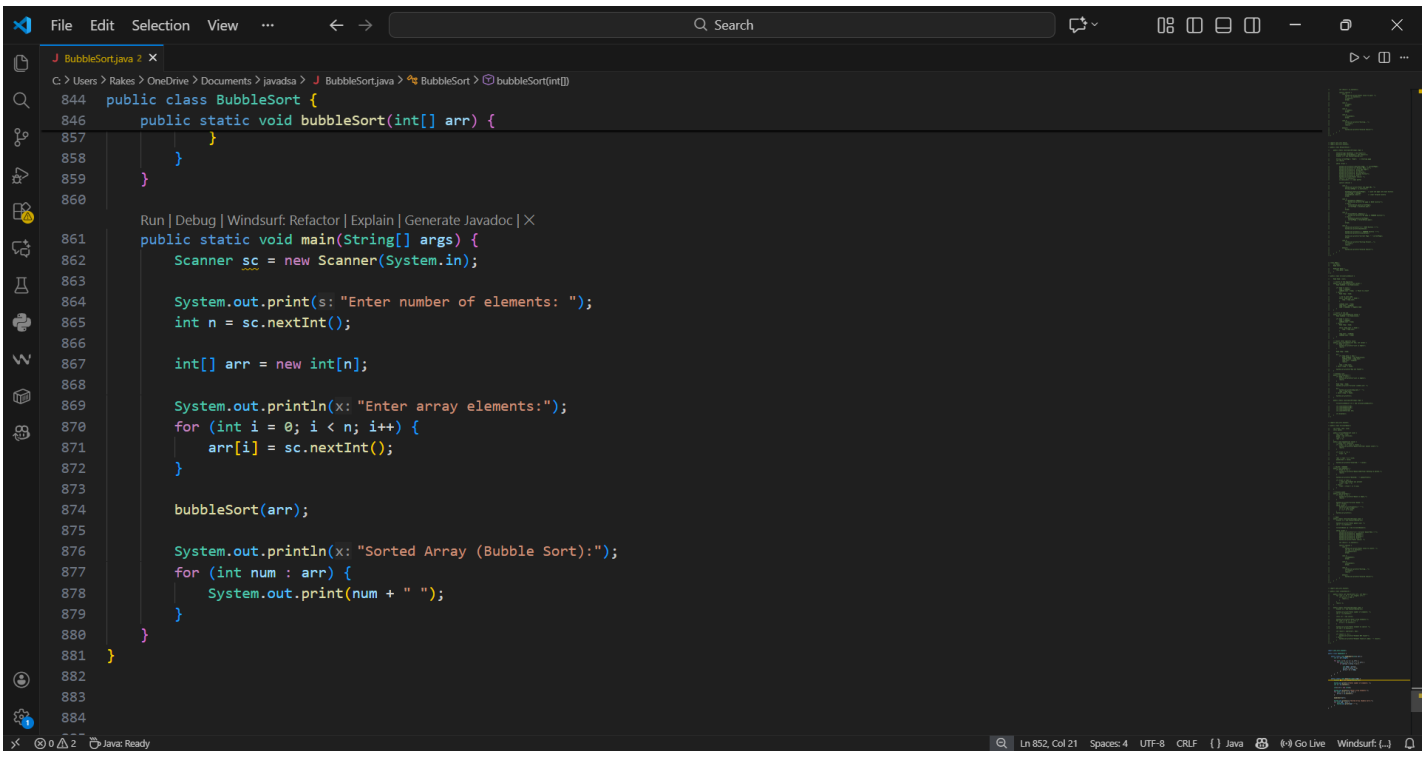

File Edit Selection View
Search
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac LinearSearch.java } ; if ($?) { java LinearSearch }
Enter number of elements: 5
Enter array elements:
1 2 3 4 5
Enter element to search: 4
Element found at index: 3
PS C:\Users\Rakes\OneDrive\Documents\javadsa>
Java: Ready
Ln 810, Col 20 Spaces: 4 UTF-8 CRLF Java Go Live

TASK 11:BUBBLE SORT

CODE:

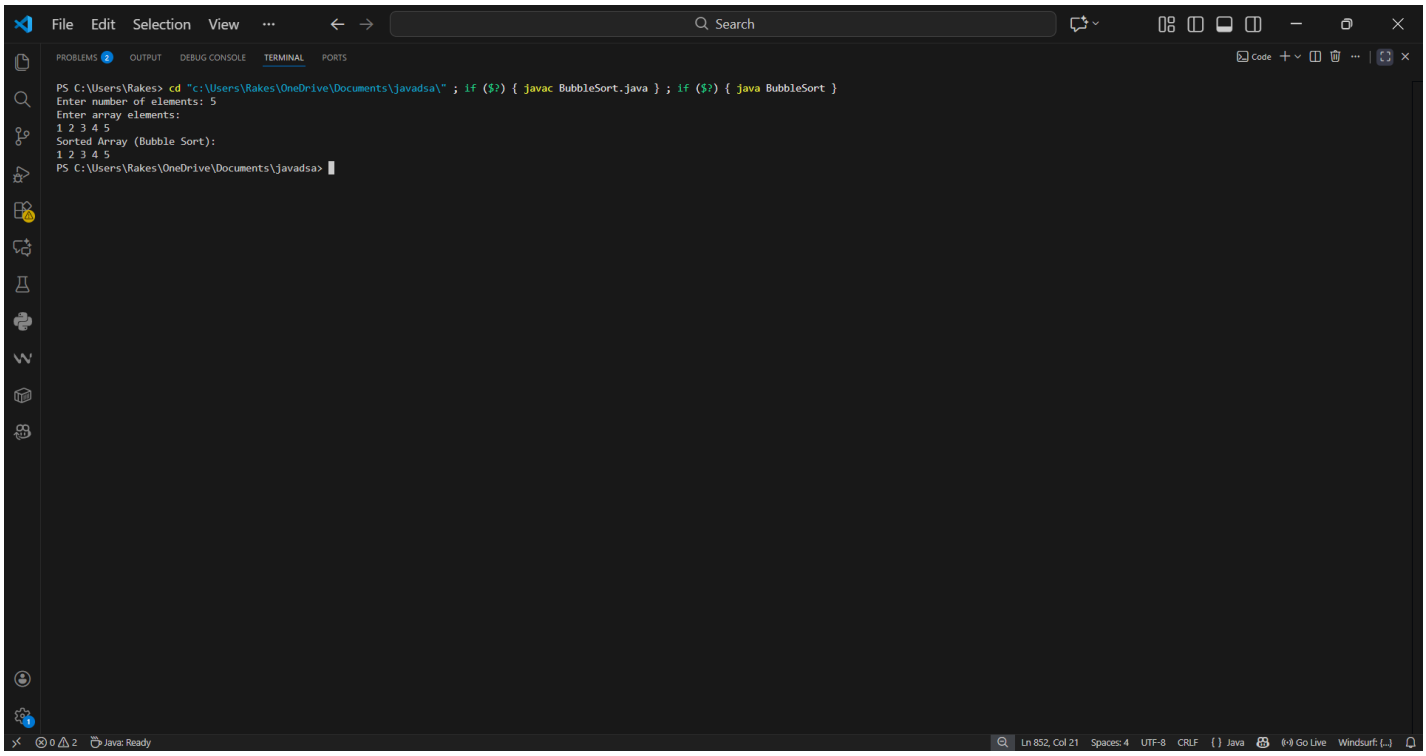


```
841
842  import java.util.Scanner;
843
     Windsurf: Refactor | Explain
844  public class BubbleSort {
845
         Windsurf: Refactor | Explain | Generate Javadoc |
846      public static void bubbleSort(int[] arr) {
847          int n = arr.length;
848
849          for (int i = 0; i < n - 1; i++) {
850              for (int j = 0; j < n - i - 1; j++) {
851                  if (arr[j] > arr[j + 1]) {
852
853                      int temp = arr[j];
854                      arr[j] = arr[j + 1];
855                      arr[j + 1] = temp;
856                  }
857              }
858          }
859      }
860
         Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
861      public static void main(String[] args) {
862          Scanner sc = new Scanner(System.in);
863
864          System.out.print(s: "Enter number of elements: ");
865          int n = sc.nextInt();
866
867          int[] arr = new int[n];
868
```



```
844  public class BubbleSort {
846      public static void bubbleSort(int[] arr) {
857          }
858      }
859  }
860
         Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc |
861      public static void main(String[] args) {
862          Scanner sc = new Scanner(System.in);
863
864          System.out.print(s: "Enter number of elements: ");
865          int n = sc.nextInt();
866
867          int[] arr = new int[n];
868
869          System.out.println(x: "Enter array elements:");
870          for (int i = 0; i < n; i++) {
871              arr[i] = sc.nextInt();
872          }
873
874          bubbleSort(arr);
875
876          System.out.println(x: "Sorted Array (Bubble Sort):");
877          for (int num : arr) {
878              System.out.print(num + " ");
879          }
880      }
881  }
882
883
884
```

OUTPUT:


File Edit Selection View
Search
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Rakes> cd "c:\Users\Rakes\OneDrive\Documents\javadsa\" ; if ($?) { javac BubbleSort.java } ; if ($?) { java BubbleSort }
Enter number of elements: 5
Enter array elements:
1 2 3 4 5
Sorted Array (Bubble Sort):
1 2 3 4 5
PS C:\Users\Rakes\OneDrive\Documents\javadsa>
Java: Ready
Ln 852, Col 21 Spaces: 4 UTF-8 CRLF { } Java Go Live