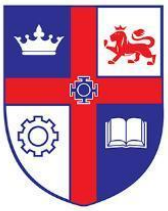
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2024 Spring CW-2

Student Name: Tapendra Singh
London Met ID: 23056247
College ID: Np01NT4S240057
Group: N10
Assignment Due Date: Friday, August 23, 2024
Assignment Submission Date: Friday, August 23, 2024

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Contents

Introduction

Java

Programming is writing a series of instructions to carry out a given activity. Java is one of many programming languages used today, each with its own set of features and capabilities available in various scales.

Programming involves instructing a machine to do predetermined actions. The process involves creating computer-readable commands. Java, formerly known as OAK, is a popular object-oriented programming language known for its stability, security, and speed. Java is a popular programming language for developing web applications. Java has been widely used by developers for almost 20 years, with millions of programs currently in use. Java is multi-platform, object-oriented, and network centric.

Programming aims to solve issues and automate tasks by creating software. The maincomponents of programming are:

Algorithm design involves breaking down an issue into a sequence of stages or algorithmsthat can be transformed into code.

Programming Languages: Using a programming language to write code for a computer

Can comprehend and execute. Examples of programming languages include Java.

Python, C++, JavaScript, and numerous others.

syntactic and Semantics: Writing according to a programming language's syntacticstandards.

Correct coding and comprehension of semantics, which defines the meaning of the code.

Coding is the process of translating algorithms into a specific programming language.

Description of BlueJ

BlueJ was developed specifically to help users learn about object-oriented programming. The interface provides visual views for classes and programmed objects. Visual representations of Java code can be ordered and organized to make computer languageseasier to understand.

Description of Coursework

London Metropolitan University has assigned us schoolwork geared at honing our skills and investigating ideas like class diagrams, pseudocode, and testing. This work is very important in our first semester programming module and requires a lot of time, effort, andresearch.

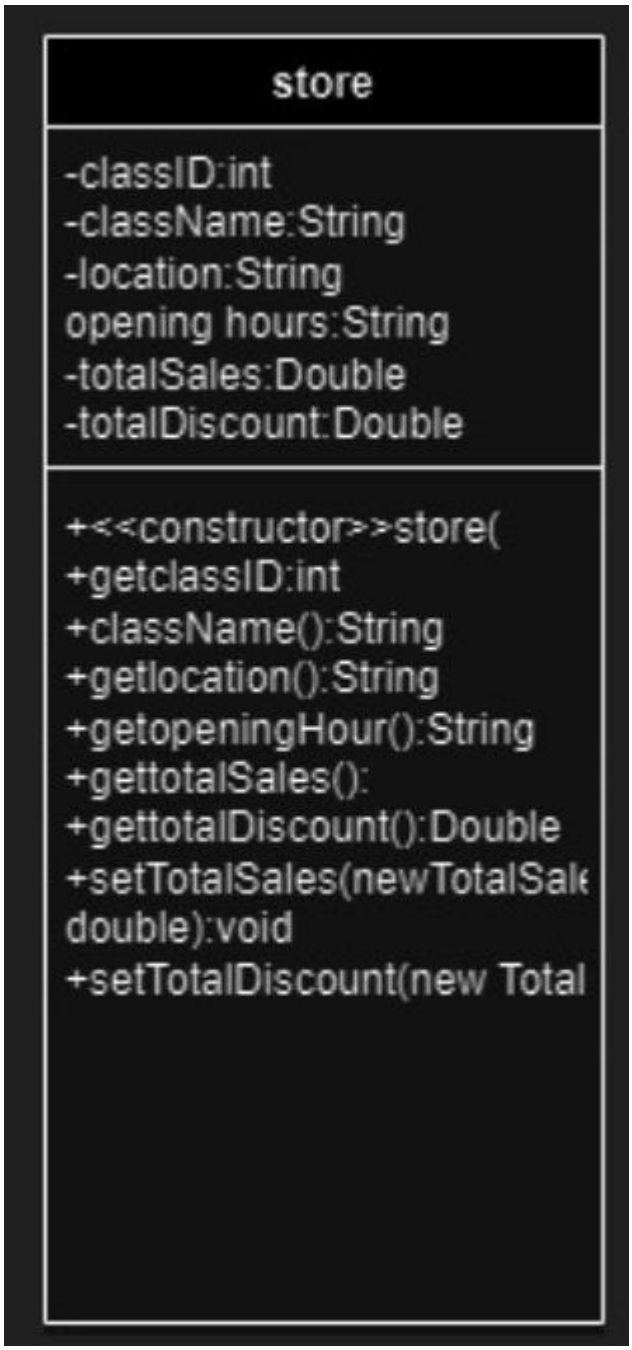
The basic goal of this course is to develop three classes, "store," "Department," and "Retailer," and then implement the necessary code. I created this report using BlueJ as a coding toTFor this report, I used BlueJ as a coding tool and Microsoft Word as the platform.

Class Diagram

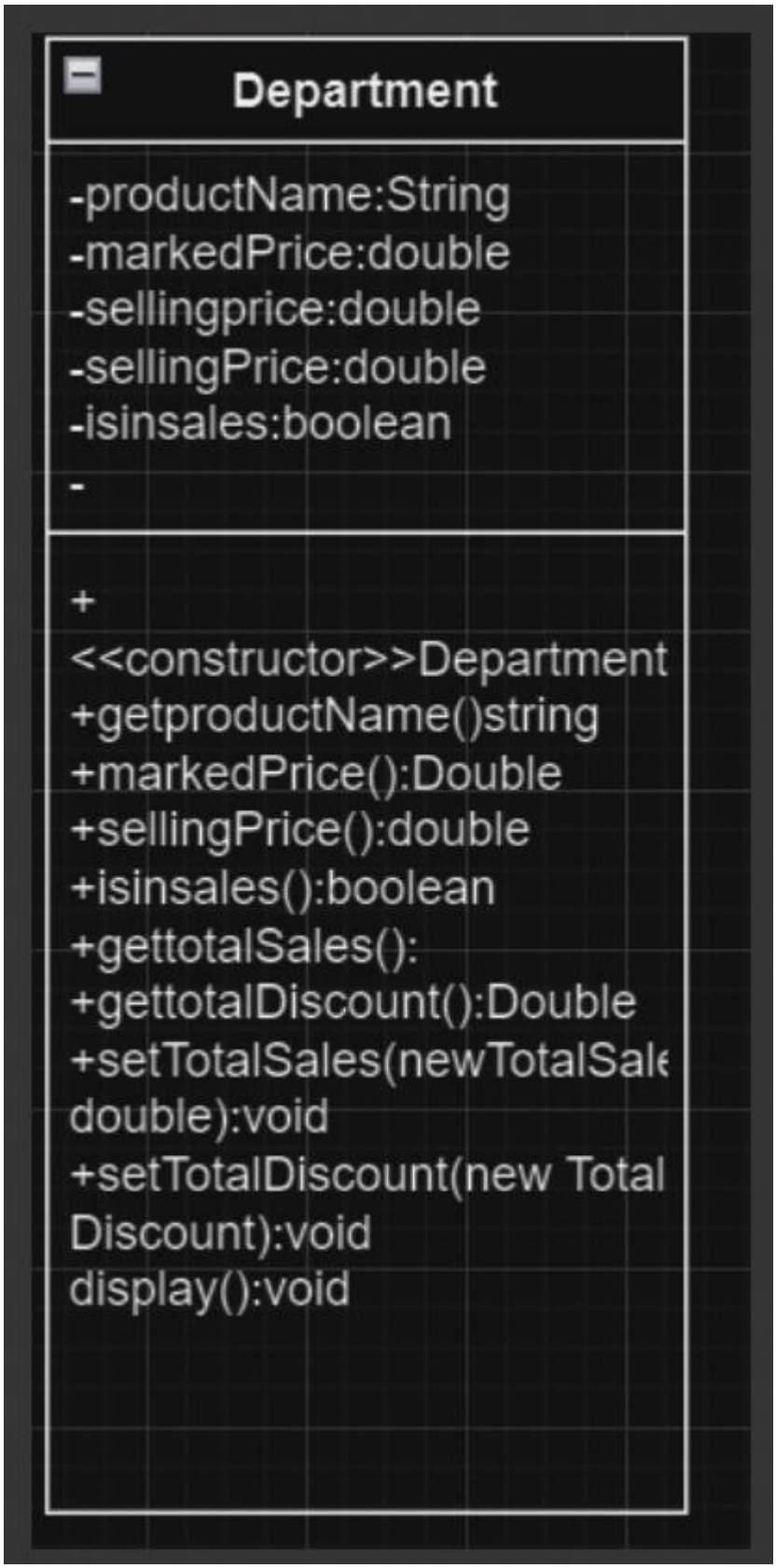
A class diagram is a visual representation that illustrates the static connections among various objects within a system. It utilizes elements like classes, attributes, relationships, and operations. This diagram is commonly acknowledged as a fundamental building block in the development of object-oriented programming languages.

A class diagram depicts static links between objects in a system. It uses elements like as classes, attributes, relationships, and operations. This diagram is widely recognized as a Fundamental building component in the evolution of object-oriented programming.

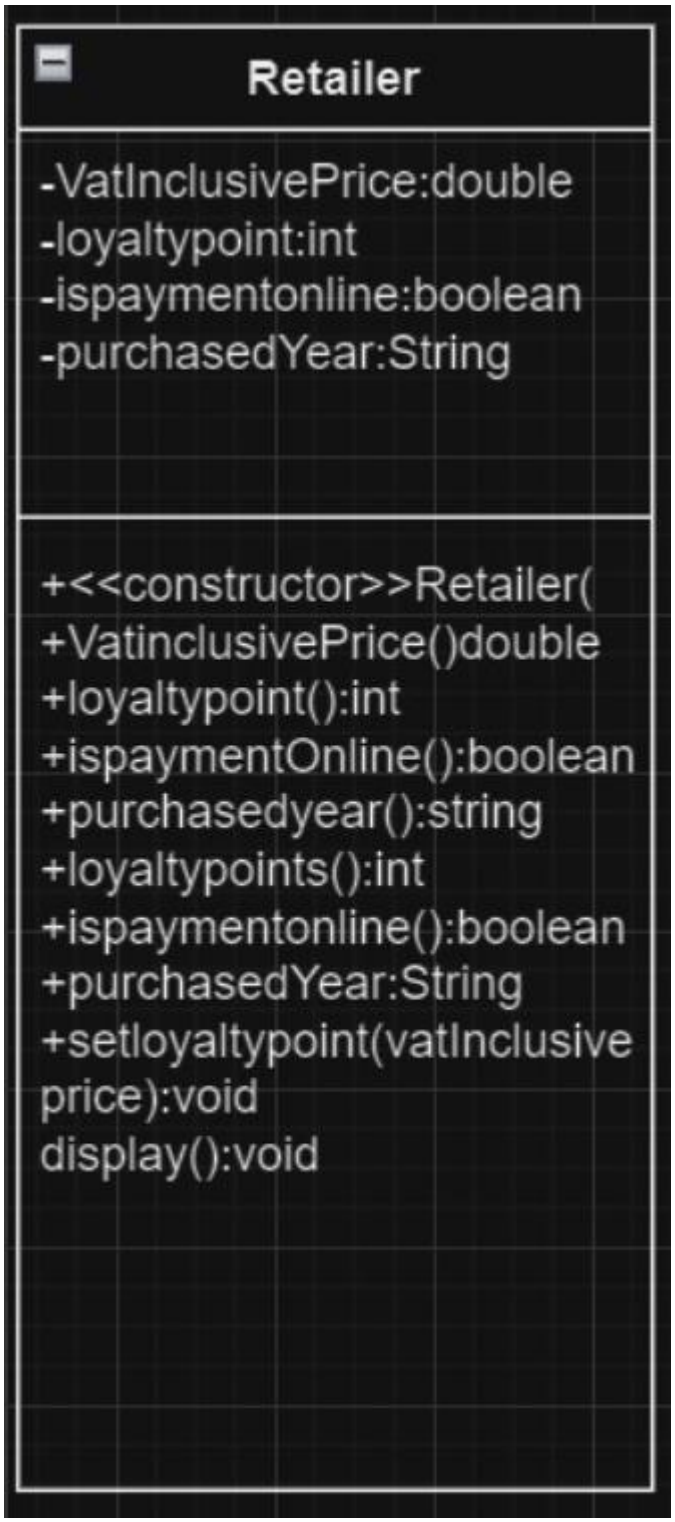
Class Diagram of Store



Class diagram of Department



Class Diagram of Retailer



Pseudocode

Programming languages are commonly used to construct programs. To ensure good program functionality, these languages must adhere to tight syntax. A basic representation of a programming language that contains the entire program's code. Programming languages are commonly used to construct programs. To ensure good program functionality, these languages must adhere to tight syntax. A basic representation of a programming language that contains the entire program's code. Pseudo code,which does not require Any specific syntax.The term pseudocode refers to a step-by-step description of algorithm. Pseudocode does not employ any programming language in its form. Instead it employs any programming language in its form. Instead,it employs plain english language text because it is intended for humans.

Pseudocode refers to the process of converting a concept into high-level language code.(Pseudo code: 2024)

Pseudocode of store

```
CLASS Store

    DECLARE PRIVATE ATTRIBUTES:
        storeID : Integer
        storeName : String
        location : String
        openingHours : String
        totalSales : Double
        totalDiscount : Double

    CONSTRUCTOR Store(storeID, storeName, location, openingHours)
        SET this.storeID = storeID
        SET this.storeName = storeName
        SET this.location = location
        SET this.openingHours = openingHours
        SET this.totalSales = 0.0
        SET this.totalDiscount = 0.0
    END CONSTRUCTOR
```

```
METHOD setTotalSales(totalSales)
   SET this.totalSales = totalSales
END METHOD

METHOD setTotalDiscount(totalDiscount)
   SET this.totalDiscount = totalDiscount
END METHOD

METHOD getStoreID()
   RETURN this.storeID
END METHOD

METHOD getStoreName()
   RETURN this.storeName
END METHOD

METHOD getLocation()
   RETURN this.location
END METHOD

METHOD getOpeningHours()
   RETURN this.openingHours
END METHOD

METHOD display()
   PRINT "Store ID: " + this.storeID
   PRINT "Store Name: " + this.storeName
   PRINT "Location: " + this.location
   PRINT "Opening Hours: " + this.openingHours

   IF this.totalSales EQUALS 0.0 AND this.totalDiscount EQUALS 0.0 THEN
      PRINT "No sales have been made."
   ELSE
      PRINT "Total sales have been made."
   END IF
END METHOD
```

END CLASS

Pseudocode of Department

```
CLASS Department EXTENDS Store

    DECLARE PRIVATE ATTRIBUTES:
        ProductName : String
        MarkedPrice : Double
        SellingPrice : Double
        isInSales : Boolean

    CONSTRUCTOR Department(storeID, storeName, location, openingHours, totalSales, totalDiscount, ProductName, MarkedPrice)
        CALL superclass constructor Store(storeID, storeName, location, openingHours)
        CALL super.setTotalSales(totalSales)
        CALL super.setTotalDiscount(totalDiscount)
        SET this.ProductName = ProductName
        SET this.MarkedPrice = MarkedPrice
        SET this.SellingPrice = 0
        SET this.isInSales = true
    END CONSTRUCTOR

    METHOD getProductName()
        RETURN this.ProductName
    END METHOD

    METHOD getMarkedPrice()
        RETURN this.MarkedPrice
    END METHOD

    METHOD getSellingPrice()
        RETURN this.SellingPrice
    END METHOD
```

```
    METHOD setMarkedPrice(MarkedPrice)
        SET this.MarkedPrice = MarkedPrice
    END METHOD

    METHOD calculateDiscountPrice(isInSales, MarkedPrice)
        DECLARE discount AS Double

        IF MarkedPrice >= 5000 THEN
            SET discount = 0.2
        ELSE IF MarkedPrice >= 3000 AND MarkedPrice < 5000 THEN
            SET discount = 0.1
        ELSE IF MarkedPrice >= 1000 AND MarkedPrice < 3000 THEN
            SET discount = 0.05
        END IF

        SET SellingPrice = MarkedPrice - (MarkedPrice * discount)
        CALL super.setTotalDiscount(MarkedPrice * discount)
        CALL super.setTotalSales(SellingPrice)
        SET this.isInSales = false
        SET this.MarkedPrice = MarkedPrice
    END METHOD

    METHOD display()
        CALL superclass method display()
        IF isInSales THEN
            PRINT "Product Name: " + ProductName
            PRINT "Selling Price: " + SellingPrice
        END IF
    END METHOD

END CLASS
```

Pseudocode of Retailer

```
CLASS Retailer EXTENDS Store

    DECLARE PRIVATE ATTRIBUTES:
        vatInclusivePrice : Integer
```

```
    loyaltyPoint : Integer
    isPaymentOnline : Boolean
    purchasedYear : String

  CONSTRUCTOR Retailer(storeID, storeName, location, openingHours, totalSales, totalDiscount, vatInclusivePrice, isPaymentOnline, purchasedYear)
    CALL superclass constructor Store(storeID, storeName, location, openingHours)
    CALL super.setTotalSales(totalSales)
    CALL super.setTotalDiscount(totalDiscount)
    SET this.vatInclusivePrice = vatInclusivePrice
    SET this.purchasedYear = purchasedYear
    SET this.loyaltyPoint = 0

    IF isPaymentOnline THEN
      CALL setLoyaltyPoint(vatInclusivePrice)
    END IF
  END CONSTRUCTOR

  METHOD getVatInclusivePrice()
    RETURN this.vatInclusivePrice
  END METHOD

  METHOD getLoyaltyPoint()
    RETURN this.loyaltyPoint
  END METHOD

  METHOD isPaymentOnline()
    RETURN this.isPaymentOnline
  END METHOD

  METHOD getPurchasedYear()
    RETURN this.purchasedYear
  END METHOD

  METHOD setIsPaymentOnline(isPaymentOnline)
    SET this.isPaymentOnline = isPaymentOnline
  END METHOD

  METHOD setLoyaltyPoint(vatInclusivePrice)
    IF isPaymentOnline THEN
      SET this.loyaltyPoint = (int)(vatInclusivePrice * 0.01)
    END IF
  END METHOD

  METHOD removeProduct()
    IF loyaltyPoint EQUALS 0 AND (purchasedYear EQUALS "2020" OR purchasedYear EQUALS "2021" OR purchasedYear EQUALS "2022") THEN
      SET vatInclusivePrice = 0
```

```
        SET loyaltyPoint = 0
        SET isPaymentOnline = false
    END IF
  END METHOD

END CLASS
```

Pseudocode of storeGUIClass Definition

CREATE class StoreGUI that implements ActionListener

Attributes:
DECLARE GUI components such as JFrame, JLabel, JTextField, JComboBox, JCheckBox,and JButton
DECLARE private attributes for store and retailer objects

Constructor:
CREATE constructor StoreGUI()

INITIALIZE GUI components and

layoutSETUP JFrame properties

INITIALIZE JLabel components for each attribute (e.g., storeID, storeName,

etc.) INITIALIZE JTextField components for input fields (e.g., storeIDT1,

storeNameT2, etc.)INITIALIZE JComboBox for purchasedYear

INITIALIZE JCheckBox for isPaymentOnline

INITIALIZE JButton components for various actions (retailerB1,

loyaltyB2, etc.)ADD components to the frame

ADD action listeners to

buttonsMethods:

CREATE method createLabel(text, font, x, y,
width, height)RETURN new JLabel

CREATE method createTextField(x, y,
width, height)RETURN new JTextField

CREATE method createButton(text, x, y, width, height)RETURN new JButton

CREATE method addComponentsToFrame(frame, components...)ADD components to JFrame

Action Listener Implementation:

IF action source is retailerB1

THEN CREATE retailer object

with input values

DISPLAY message "Retailer is added

successfully!" ELSE IF action source is

loyaltyB2 THEN

CHECK if currentRetailer is not

null CALL setloyaltypoint on

currentRetailer

DISPLAY message "Loyalty points updated

successfully"ELSE

DISPLAY message "No retailer is

available!" ELSE IF action source is

removeProductB3 THEN CHECK if

currentRetailer is not null

CALL removeproduct on currentRetailer

DISPLAY message "Product removed

successfully!"ELSE

DISPLAY message "No retailer

available!" ELSE IF action source is

displayB4 THEN CHECK if

currentRetailer is not null

CALL display on

currentRetailerELSE

DISPLAY message "No retailer is

available!" ELSE IF action source is

clearB5 THEN

CLEAR all input fields and reset JCheckBox and

JComboBox ELSE IF action source is retailerBackB6

THEN

HIDE and DISPOSE current

frameMain Method:

CREATE method main(args)

INSTANTIATE StoreGUI

Description of method

Methodology explanation In Java, a method is a piece of code that, when called, carries out the defined duties given in it. The method will carry out written instructions, for instance, if it is to draw a straight line. Procedures that incorporate values or parameters will only ever be used in order to carry them out. Java has numerous advantages, including increases readability of code, permits code reuse, and _most importantly_divides complex program into a chunk of code that is easier.

- **Constructor**: StoreGUI()
    - **Description**: Initializes the StoreGUI object. Sets the title, size, default close operation, and layout manager for the JFrame. Creates labels, text fields, combo boxes, checkboxes, and buttons for user input and actions. Adds these components to the frame and sets up action listeners for eachbutton.

- **Method**: createLabel()
  - **Description**: Creates a JLabel with specified text, font, and position on the
    JFrame. Returns the configured JLabel object.
- **Method**: createTextField()
  - **Description**: Creates a JTextField with specified size and position on the
    JFrame. Returns the configured JTextField object.
- **Method**: createButton()
  - **Description**: Creates a JButton with specified text and position on the JFrame. Sets the button's background color and returns the configuredJButtonobject.
- **Method**: addComponentsToFrame()

  - **Description**: Adds multiple components to the JFrame. This method simplifies the addition of various components by iterating through an array ofcomponents and adding each to the frame.
- **Method**: actionPerformed()
  - **Description**: Handles the action events for various buttons in the GUI.
    - **Case**: retailerB1(Add to Retailer)
      - **Description**: Handles the addition of a retailer. Parses input from text fields to create a new retailer object with the provided information. Displays a success message if the addition is successful.
    - **Case**: loyaltyB2(Loyalty Point)
      - **Description**: Updates the loyalty points of the current retailer. Calls the setloyaltypoint method on the currentRetailer object if available and shows a success message. Displays an error message if no retailer is available.
    - **Case**: removeProductB3(Remove Product)
      - **Description**: Removes the product associated with the current retailer. Calls the removeproduct method on the currentRetailer object if available. Displays an errormessage if no retailer is available.
    - **Case**: displayB4(Display)
      - **Description**: Displays the information of the current retailer. Calls the display method on the currentRetailer

object if available. Displays an error message if no retailer is available.

        - **Case**: clearB5(Clear)
            - **Description**: Clears all input fields and resets the checkboxes and combo boxes. Sets the text fields to empty strings, unchecks the checkbox, and resets the combo box to its default value.
        - **Case**: retailerBackB6(Back to Main Page)
            - **Description**: Closes the current JFrame and returns to the main page. Sets the visibility of the frame to false and disposesof it.
- **Method**: main()
    - **Description**: Entry point of the application. Creates a new instance of StoreGUI, which initializes the GUI components and makes the framevisible.

Conclusion

Through the development of the StoreGUI application, I have enhanced my understanding of Java Swing for creating graphical user interfaces. By implementing various components such as JFrame, JLabel, JTextField, JComboBox, JCheckBox, and JButton, I learned how to design and layout a complex GUI effectively. Each element was carefully configured and added to ensure a user-friendly interface for managing store data.

One of the challenges I encountered was managing the interaction between different GUI components and ensuring that user actions were properly handled. This required a solid grasp of event handling in Java, particularly using the ActionListener interface to respond to button clicks. I had to meticulously implement and test these event-driven behaviors to ensure they worked as intended.

To overcome these challenges, I employed a systematic approach to testing and debugging. By simulating various user interactions and examining the GUI's response, I identified and resolved potential issues. This process not only improved the functionality of the application but also deepened my

understanding of how different GUI components interact within the Java Swing framework.

In conclusion, working on this project provided valuable experience in developing a GUI- based application in Java. It reinforced my knowledge of object-oriented programming principles, particularly encapsulation and inheritance, and improved my ability to design intuitive user interfaces. This project has bolstered my confidence in using Java for real- world applications and furthered my skills in software development.

References

https://www.w3schools.com I used this website in coding.
https://www.geeksforgeeks.org/ I used this wesite for Pseudocode definition.
https://www.geeksforgeeks.org/ I used this website for class diagram.
https://www.w3schools.com/java/java_intro.asp i used this website for java definition
https://www.techopedia.com/definition/29530/bluej I used this website for bluej definition

Appendix

Code of store

```
public class store  //store class is parents class
{
  //setting value in attributes
  private int storeID;
  private String
  storeName; private
  String location; private
  String openinghours;
  private double
  TotalSales; private
  double totaldiscount;
  //constructor for corresponding values
  public store(int storeID,String storeName, String location,String
    openinghour){this.storeID = storeID;
    this.storeName=storeName;
    this.location=location;
    this.openinghours=openinghours;
    this.TotalSales=0.0;
    this.totaldiscount=0.0;

  }
  //using setter method
   public void setTotalSales(double TotalSales){
     this.TotalSales=TotalSales;
   }
   public void settotaldiscount(double totaldiscount){
     this.totaldiscount=totaldiscount;
   }
   //using accessor method

  public int
    getstoreID(){
    return
    this.storeID;
```

```
}
public                    String
  getstoreName(){    return
  this.storeName;
}
public              String
  getlocation(){   return
  this.location;
}
public                    String
  getopeninghours(){  return
  this.openinghours;
}

//method        for
display      public
void display(){
System.out.println("storeID:"+              storeID);
System.out.println("storeName:"+   storeName);
System.out.println("location:"+            location);
System.out.println("openinghours:"+
openinghours);    if(TotalSales    ==    0.0    &&
totaldiscount == 0.0){
  System.out.println("No sales have been made");
}else {
  System.out.println("total sales have been made");
}

 }
}
```

Code of Department

```
public class Department extends
  store{        private        String
  productName;
  private             double
  markedPrice;     private
```

```
  double sellingPrice;
  private boolean
  isInSales;
  //Constructor method of department subclasss
  public Department(int storeId, String storeName, String location,
String openingHour, double TotalSales, double totalDiscount, String
prouductName,double markedPrice){
    super(storeId, storeName,location,
    openingHour);
    super.setTotalSales(TotalSales);
    super.settotaldiscount(totalDiscount);
    this.productName=productName;
    this.markedPrice=markedPrice;
    this.sellingPrice=0.0;
    this.isInSales=true;
    calculateDiscountPrice(
    );
  }//Accessor methods of department
  subclasspublic String getproductName(){
    return productName;

  }
  public double
    getMarkedPrice(){ return
    markedPrice;
  }



  //mutator method of marked price
  public void setMarkedPrice(double MarkedPrice){
   this.markedPrice=MarkedPrice;


  }//method forcalculating selling price based on
  discountsprivate void calculateDiscountPrice(){
    double discount=0.0;
    if(markedPrice>=5000){
```

```
        discount=0.2;
      }else                    if(markedPrice>=3000&&
        markedPrice<5000){discount=0.1;
      }else  if  (markedPrice>=1000&&  markedPrice<3000){
        discount=0.05;
      }sellingPrice=markedPrice-(markedPrice*discount);
      settotaldiscount(markedPrice*discount);
      setTotalSales(sellingPrice);
      this.isInSales=false;
      this.markedPrice=markedPrice;
      }
      //Display  method  of  department
      subclasspublic void display(){
        super.display();
        if(isInSales){
          System.out.println("ProductName:"+productName);
          System.out.println("MarkedPrice:"+markedPrice);


        }else{
          System.out.println("Sellingprice:"+sellingPrice);



      }
    }

}
```

code of Retailer

```
public  class  retailer  extends
  store{       private       int
  vatInclusiveprice;  private  int
  loyaltypoint;
  private                  boolean
  Ispaymentonline;  private  String
  purchasedyear;
```

```
//constructor for corresponding value
 public retailer(int storeID,String storename,String location, String
openinghour,double TotalSales,double totaldiscount,
          int vatInclusiveprice,boolean Ispaymentonline,String
    purchasedyear){ super(storeID, storename,location,openinghour);
    super.setTotalSales(TotalSales);
    super.settotaldiscount(totaldiscount);
    this.vatInclusiveprice = vatInclusiveprice;
    this.purchasedyear = purchasedyear;
    this.loyaltypoint = 0;
    if(Ispaymentonline) {
     setloyaltypoint(vatInclusiveprice);
    }
  }
  //using Accessor methods
  public int
  getvatInclusiveprice(){
    return this.vatInclusiveprice;
  }
    public int
      getloyaltypoint(){
      return
      this.loyaltypoint;
}
public boolean Ispaymentonline()
  {return this.Ispaymentonline;
}
public String
  getpurchasedyear() { return
  this.purchasedyear;
}
 //using setter method
  public void setsIspaymentonline(boolean Ispaymentonline) {
   this.Ispaymentonline = Ispaymentonline;
  }
  //setting loyaltypoints to vatInclusiveprice
```

```
  public void setloyaltypoint(int vatInclusiveprice) {
    if(Ispaymentonline){
    this.loyaltypoint = (int) (vatInclusiveprice * 0.01);
  }
}
  //method for removing
  product public void
  removeproduct() {
    if (loyaltypoint ==0 && (purchasedyear.equals("2020") ||
purchasedyear.equals("2021") ||
      purchasedyear.equals("2022"))){vatInclusiveprice = 0;
      loyaltypoint = 0;
      Ispaymentonline =
      false;
    }
  }
  //using display
  method public void
  display(){
    if (loyaltypoint !=0 && !(purchasedyear.equals("2020") ||
purchasedyear.equals("2021") || purchasedyear.equals("2022"))){
      super.display();
      System.out.println("VatInclusiveprice is:"+ vatInclusiveprice);
      System.out.println("Loyalty point is:"+ loyaltypoint);
      System.out.println("Purchased year is:"+ purchasedyear);
    }else{
      System.out.println("Product has been removed");
    }
  }
}
```

Code of storeGUI

5.2 import
javax.swing.*;
import java.awt.*;
import
java.awt.event.ActionEvent;

```
import
java.awt.event.ActionListener;

public class StoreGUI implements ActionListener {
  // DECLARING INSTANCE VARIABLES
  JFrame retailerFrame;
  JLabel titleRetailer, storeID, storeName, location, openingHour,
totalSales, totalDiscount, vatInclusivePrice, loyaltyPoint,
isPaymentOnline, purchasedYear;
  JTextField storeIDT1, storeNameT2, locationT3, openingHourT4,
totalSalesT5,totalDiscountT6, vatInclusivePriceT7, loyaltyPointT8;
  JComboBox<String>  purchasedYearCB;
  JCheckBox checkBox1;
  JButton retailerB1, loyaltyB2, removeProductB3, displayB4, clearB5,
retailerBackB6;

  // Store and Retailer
  objects private store
  currentStore; private
  retailer currentRetailer;

  public StoreGUI() {
    // Creating new Retailer frame
    retailerFrame = new JFrame("Retailer Frame");
    retailerFrame.setResizable(false);
    retailerFrame.setLayout(null);
    retailerFrame.setSize(900, 800);
    retailerFrame.getContentPane().setBackground(Color.decode("#D3
    E5FF"));

    // Creating and configuring JLabels
    titleRetailer = createLabel("Retailer Frame", new Font("plain",
Font.BOLD, 26),300, 20, 300, 40);
    storeID = createLabel("Store ID:", null, 50, 80, 150, 30);
    storeName = createLabel("Store Name:", null, 50, 130, 150, 30);
    location = createLabel("Location:", null, 50, 180, 150, 30);
    openingHour = createLabel("Opening Hour:", null, 50, 230, 150, 30);
    totalSales = createLabel("Total Sales:", null, 50, 280, 150, 30);
```

```
totalDiscount = createLabel("Total Discount:", null, 50, 330, 150, 30);
vatInclusivePrice = createLabel("VAT Inclusive Price:", null, 50, 380, 200, 30);
loyaltyPoint = createLabel("Loyalty Point:", null, 50, 430, 150, 30);
purchasedYear = createLabel("Purchased Year:", null, 50, 480, 150, 30);
isPaymentOnline = createLabel("Is Payment Online:", null, 620, 80, 200, 30);

// Creating and configuring JTextFields
storeIDT1 = createTextField(250, 80,
200, 30);
storeNameT2 = createTextField(250, 130, 200, 30);
locationT3 = createTextField(250, 180, 200, 30);
openingHourT4 = createTextField(250, 230, 200, 30);
totalSalesT5 = createTextField(250, 280, 200, 30);
totalDiscountT6 = createTextField(250, 330, 200, 30);
vatInclusivePriceT7 = createTextField(250, 380, 200, 30);
loyaltyPointT8 = createTextField(250, 430, 200, 30);

// Creating JComboBox for purchased
year String[] years = {"2006", "2004",
"2008", "2010"};
purchasedYearCB = new JComboBox<>(years);
purchasedYearCB.setBounds(250, 480, 150, 30);

// Creating JCheckBox for
isPaymentOnline checkBox1 = new
JCheckBox();
checkBox1.setBounds(750, 80, 20, 30);
checkBox1.setBackground(Color.decode("#BBCED2"));
// Creating JButtons and setting their bounds
retailerB1 = createButton("Add to Retailer", 600, 150, 250, 40);
loyaltyB2 = createButton("Loyalty Point", 600, 230, 250, 40);
removeProductB3 = createButton("Remove Product", 600, 310, 250, 40);
displayB4 = createButton("Display", 600, 390, 250, 40);
clearB5 = createButton("Clear", 600, 470, 250, 40);
```

```
    retailerBackB6 = createButton("Back to Main Page", 600, 550, 250,
    40);

    // Adding all components to the frame
    addComponentsToFrame(retailerFrame, titleRetailer, storeID,
    storeName,
location, openingHour, totalSales, totalDiscount, vatInclusivePrice,
loyaltyPoint, purchasedYear, isPaymentOnline, storeIDT1, storeNameT2,
locationT3, openingHourT4, totalSalesT5, totalDiscountT6,
vatInclusivePriceT7, loyaltyPointT8, purchasedYearCB, checkBox1,
retailerB1, loyaltyB2, removeProductB3, displayB4, clearB5,
retailerBackB6);

    // Add action listeners
    retailerB1.addActionListener(this);
    loyaltyB2.addActionListener(this);
    removeProductB3.addActionListene
    r(this);
    displayB4.addActionListener(this);
    clearB5.addActionListener(this);
    retailerBackB6.addActionListener(t
    his);

    retailerFrame.setVisible(true);
  }

  private JLabel createLabel(String text, Font font, int x, int y, int width, int
    height) {JLabel label = new JLabel(text);
    if (font != null) {
      label.setFont(font);
    }
    label.setBounds(x, y, width,
    height);return label;
  }

  private JTextField createTextField(int x, int y, int width, int
    height) {JTextField textField = new JTextField();
```

```
    textField.setBounds(x,  y,  width,
    height);return textField;
  }

  private JButton createButton(String text, int x, int y, int width, int
    height) {JButton button = new JButton(text);
    button.setBounds(x,  y,  width,  height);
    button.setBackground(Color.decode("#4CAF5
    0")); return button;
  }

  private void addComponentsToFrame(JFrame frame, Component...
  components)
{
    for  (Component  component  :  components)  {
      frame.add(component);
    }
  }

  @Override
  public                                  void
    actionPerformed(ActionEvent  e)  {  if
    (e.getSource() == retailerB1) {
      // Creating a retailer based on user input
      int                  storeID                  =
      Integer.parseInt(storeIDT1.getText());  String
      storeName     =     storeNameT2.getText();
      String location = locationT3.getText();
      String openingHour = openingHourT4.getText();
      double                totalSales                =
      Double.parseDouble(totalSalesT5.getText());          double
      totalDiscount                                     =
      Double.parseDouble(totalDiscountT6.getText());          int
      vatInclusivePrice                                 =
      Integer.parseInt(vatInclusivePriceT7.getText());     boolean
      isPaymentOnline = checkBox1.isSelected();
      String          purchasedYear          =          (String)
      purchasedYearCB.getSelectedItem();
```

```
        currentRetailer = new retailer(storeID, storeName, location,
openingHour, totalSales, totalDiscount, vatInclusivePrice,
isPaymentOnline, purchasedYear);
        JOptionPane.showMessageDialog(retailerFrame, "Retailer is
addedsuccessfully!");
    } else if (e.getSource() ==
      loyaltyB2) { if (currentRetailer !=
      null) {
        currentRetailer.setloyaltypoint(currentRetailer.getvatInclusivepri
        ce());
        JOptionPane.showMessageDialog(retailerFrame, "Loyalty points
is updatedsuccessfully");
      } else {
        JOptionPane.showMessageDialog(retailerFrame, "No retailer is
        available!");
      }
    } else if (e.getSource() ==
      removeProductB3) { if (currentRetailer
      != null) {
        currentRetailer.removeproduct();
      } else {
        JOptionPane.showMessageDialog(retailerFrame, "No retailer
        available!");
      }
    } else if (e.getSource() ==
      displayB4) { if (currentRetailer
      != null) {
        currentRetailer.display();
      } else {
        JOptionPane.showMessageDialog(retailerFrame, "No retailer is
        available!");
      }
    } else if (e.getSource() == clearB5) {
      // Clearing all input fields
      storeIDT1.setText("");
      storeNameT2.setText("");
      locationT3.setText("");
```

```
        openingHourT4.setText("");
        totalSalesT5.setText("");
        totalDiscountT6.setText("");
        vatInclusivePriceT7.setText("");
        loyaltyPointT8.setText("");
        checkBox1.setSelected(false);
        purchasedYearCB.setSelectedIndex(0)
        ;
      } else if (e.getSource() == retailerBackB6)
        {          retailerFrame.setVisible(false);
        retailerFrame.dispose();
      }
    }

    public  static  void  main(String[]
      args) {new StoreGUI();
    }
}
```