CS4001NI Programming

30% Individual Coursework

2024 Spring

Student Name: Rohan Shrestha

London Met ID: 23056199

College ID: NP01NT4S240007

Group: N9

Assignment Due Date: Friday, May 10, 2024

Assignment Submission Date: Wednesday, May 8, 2024

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1. About the coursework

The aim of this assignment is to implement a real-world problem scenario using the Object-oriented concept of Java that includes creating a class to represent a Store, together with its two subclasses to represent a Department and a Retailer respectively. By creating classes with certain behaviours and attributes and reusing parent class functionality in their subclasses, this program illustrates Java's concepts of inheritance, encapsulation, and polymorphism.

1.2. Tools Used

BlueJ

BlueJ is an integrated development environment (IDE) specifically designed for teaching and learning object-oriented programming, primarily using the Java programming language. Developed by the University of Kent and Deakin University, it offers an easy-to-use interface featuring interactive objects, built-in editor, compiler, and a debugger. BlueJ simplifies the software development process, making it suitable for beginners while still offering advanced features for experienced programmers. (DevX, 2023)

Figure 1 - Logo of BlueJ

Microsoft Word

Microsoft Word is a popular word processing software developed by Microsoft Corporation, used for creating, editing, and formatting text documents. It offers numerous features such as spell checking, grammar checking, text formatting, and inserting images, tables, and other visual elements. Microsoft Word is an essential productivity tool across various industries and is available on multiple platforms, including Windows, macOS, iOS, and Android devices. (DevX, 2024)

Figure 2 - Logo of Microsoft Word

Draw.io

draw.io is a technology stack for building diagramming applications, and the world’s most widely used browser-based end-user diagramming software. (Draw.io, 2023)

Figure 3 - Logo of Draw.io

2. Class Diagram

A class diagram is a type of UML (Unified Modelling Language) diagram that visually represents the structure and relationships among classes within a software system. It depicts the static view of the system by illustrating its classes, attributes, methods, and the relationships between them such as inheritance, association, and aggregation. Class diagrams are commonly used in object-oriented design and analysis, serving as a blueprint for software development. (DevX, 2023)

Example of class diagram:

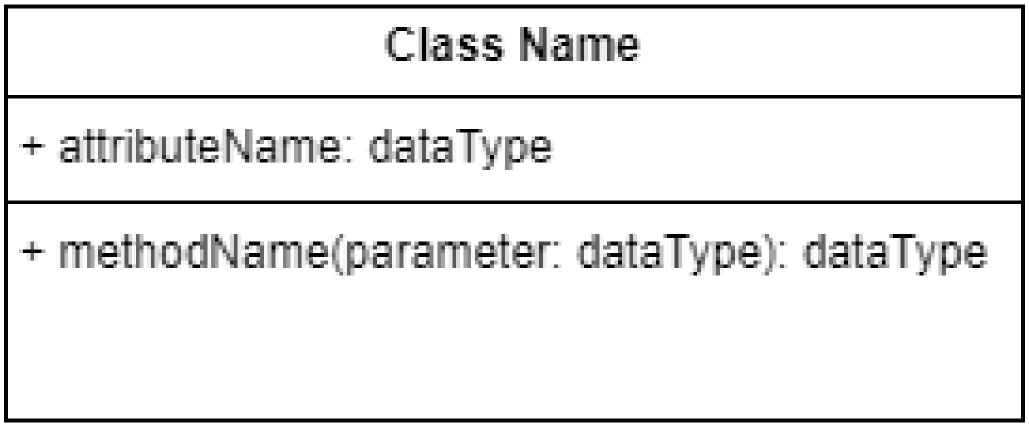

Class Name
+ attributeName: dataType
+ methodName(parameter: dataType): dataType

Figure 4 - Class diagram example

2.1. Class diagram of Store class

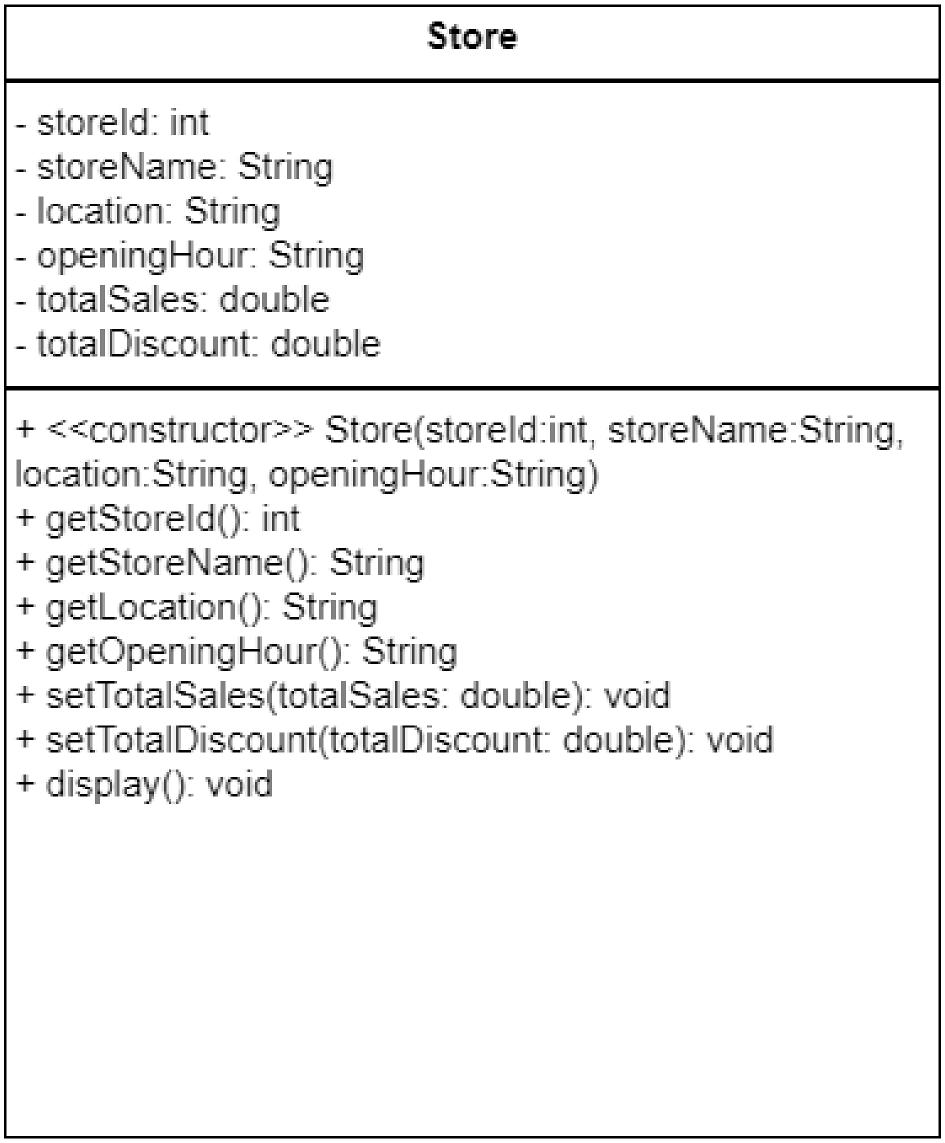

Store
- storeId: int
- storeName: String
- location: String
- openingHour: String
- totalSales: double
- totalDiscount: double
+ <<constructor>> Store(storeId:int, storeName:String, location:String, openingHour:String)
+ getStoreId(): int
+ getStoreName(): String
+ getLocation(): String
+ getOpeningHour(): String
+ setTotalSales(totalSales: double): void
+ setTotalDiscount(totalDiscount: double): void
+ display(): void

Figure 5 - Class diagram of Store

2.2. Class diagram of Department class

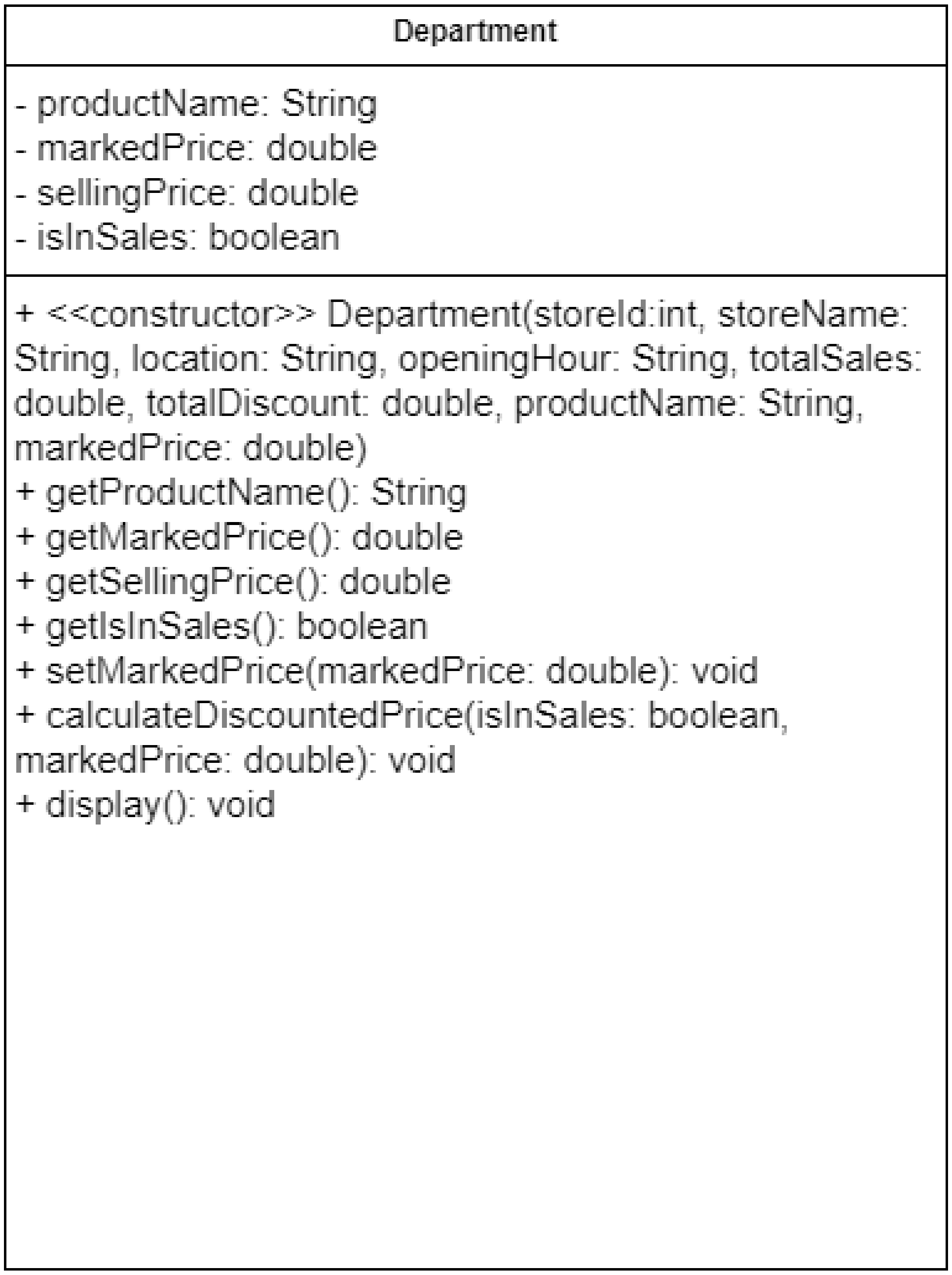
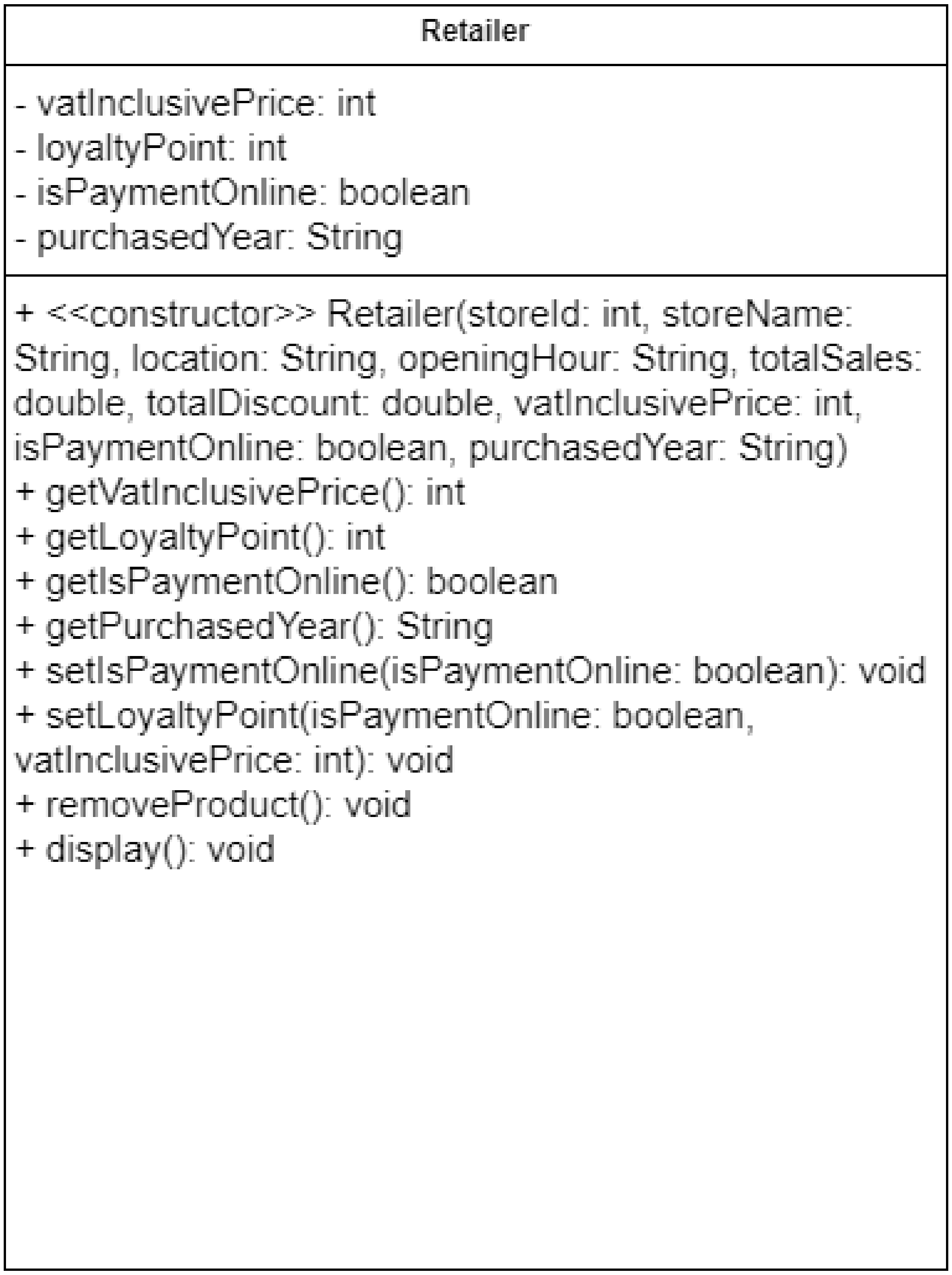

Department
- productName: String
- markedPrice: double
- sellingPrice: double
- isInSales: boolean
+ <<constructor>> Department(storeId:int, storeName: String, location: String, openingHour: String, totalSales: double, totalDiscount: double, productName: String, markedPrice: double)
+ getProductName(): String
+ getMarkedPrice(): double
+ getSellingPrice(): double
+ getIsInSales(): boolean
+ setMarkedPrice(markedPrice: double): void
+ calculateDiscountedPrice(isInSales: boolean, markedPrice: double): void
+ display(): void

Figure 6 - Class diagram of Department

2.3. Class diagram of Retailer class



- vatInclusivePrice: int
- loyaltyPoint: int
- isPaymentOnline: boolean
- purchasedYear: String

+ <<constructor>> Retailer(storeId: int, storeName: String, location: String, openingHour: String, totalSales: double, totalDiscount: double, vatInclusivePrice: int, isPaymentOnline: boolean, purchasedYear: String)
+ getVatInclusivePrice(): int
+ getLoyaltyPoint(): int
+ getIsPaymentOnline(): boolean
+ getPurchasedYear(): String
+ setIsPaymentOnline(isPaymentOnline: boolean): void
+ setLoyaltyPoint(isPaymentOnline: boolean, vatInclusivePrice: int): void
+ removeProduct(): void
+ display(): void

Figure 7 - Class diagram of Retailer class

2.4. Inheritance Diagram

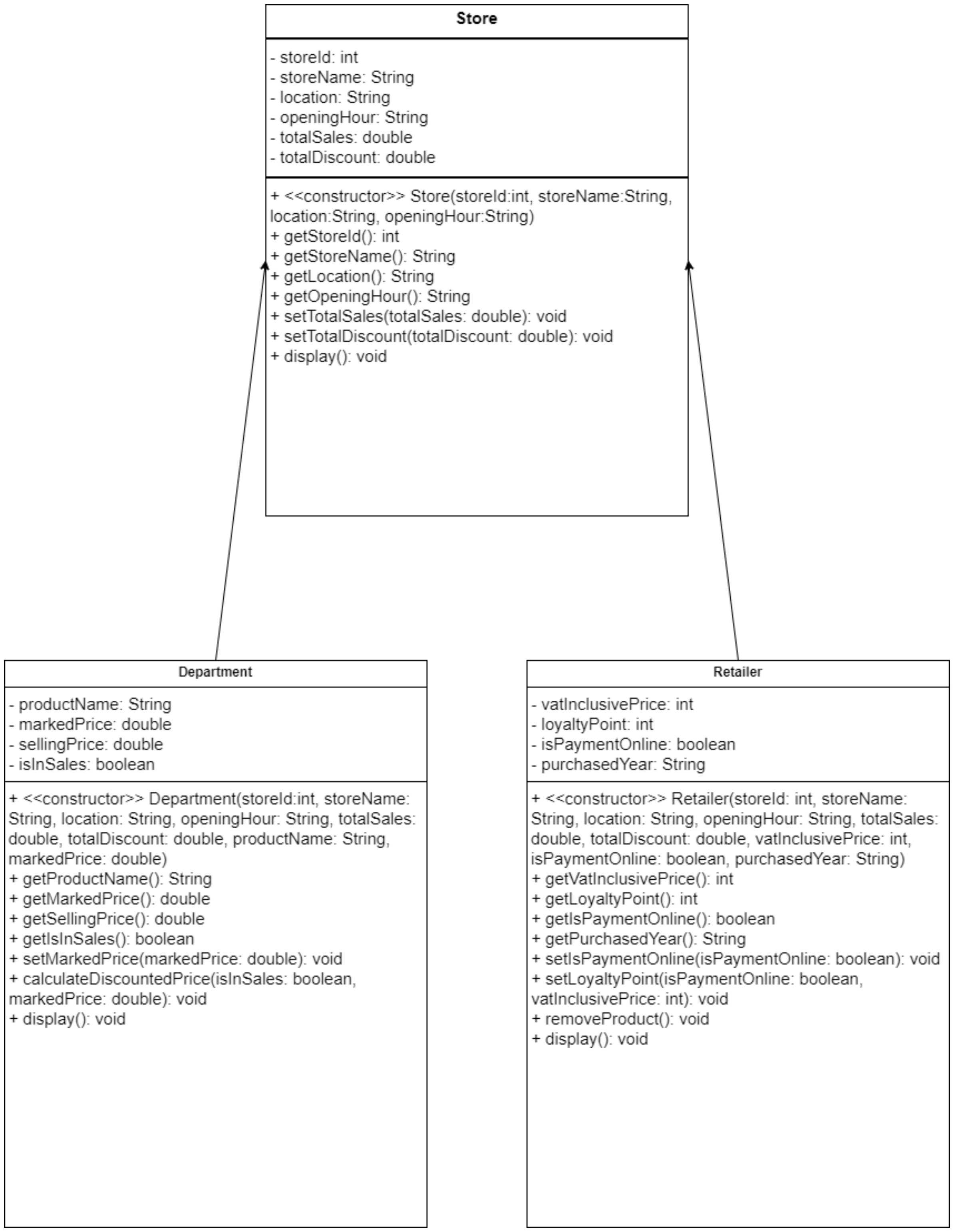

Store
- storeId: int
- storeName: String
- location: String
- openingHour: String
- totalSales: double
- totalDiscount: double
+ <<constructor>> Store(storeId:int, storeName:String, location:String, openingHour:String)
+ getStoreId(): int
+ getStoreName(): String
+ getLocation(): String
+ getOpeningHour(): String
+ setTotalSales(totalSales: double): void
+ setTotalDiscount(totalDiscount: double): void
+ display(): void
Department
- productName: String
- markedPrice: double
- sellingPrice: double
- isInSales: boolean
+ <<constructor>> Department(storeId:int, storeName: String, location: String, openingHour: String, totalSales: double, totalDiscount: double, productName: String, markedPrice: double)
+ getProductName(): String
+ getMarkedPrice(): double
+ getSellingPrice(): double
+ getIsInSales(): boolean
+ setMarkedPrice(markedPrice: double): void
+ calculateDiscountedPrice(isInSales: boolean, markedPrice: double): void
+ display(): void
Retailer
- vatInclusivePrice: int
- loyaltyPoint: int
- isPaymentOnline: boolean
- purchasedYear: String
+ <<constructor>> Retailer(storeId: int, storeName: String, location: String, openingHour: String, totalSales: double, totalDiscount: double, vatInclusivePrice: int, isPaymentOnline: boolean, purchasedYear: String)
+ getVatInclusivePrice(): int
+ getLoyaltyPoint(): int
+ getIsPaymentOnline(): boolean
+ getPurchasedYear(): String
+ setIsPaymentOnline(isPaymentOnline: boolean): void
+ setLoyaltyPoint(isPaymentOnline: boolean, vatInclusivePrice: int): void
+ removeProduct(): void
+ display(): void

Figure 8 - Inheritance diagram

3. Pseudocode

Pseudocode is a simplified high-level description of a computer program or algorithm, typically used for planning before the actual coding begins. It uses the structure and syntax of programming languages but is written in an informal, human-readable language. Pseudocode does not necessarily adhere to specific coding conventions or run on a computer, but serves as a conceptual tool to help developers organize and plan out their code. (DevX, 2023)

3.1. Pseudocode for Store Class

```
START

CREATE a parent class Store

DO

    DECLARE instance variable storeId as int using private access modifier

    DECLARE instance variable storeName as String using private access modifier

    DECLARE instance variable location as String using private access modifier

    DECLARE instance variable openingHour as String using private access modifier

    DECLARE instance variable totalSales as double using private access modifier

    DECLARE instance variable totalDiscount as double using private access
    modifier


    CREATE a constructor that accepts the following parameters: int storeId, String
    storeName, String location, String openingHour

    DO

        SET the value of storeId to parameter storeId

        SET the value of storeName to parameter storeName

        SET the value of location to parameter location

        SET the value of openingHour to parameter openingHour

        SET the value of totalSales to zero

        SET the value of totalDiscount to zero

    END DO
```

```
CREATE accessor methods for each attribute of Store class
DO
    CREATE a method getStoreId() with return type int
    DO
        RETURN storeId
    END DO

    CREATE a method getStoreName() with return type String
    DO
        RETURN storeName
    END DO

    CREATE a method getLocation() with return type String
    DO
        RETURN location
    END DO

    CREATE a method getOpeningHour() with return type String
    DO
        RETURN openingHour
    END DO
END DO
```

```
    CREATE a mutator method setTotalSales() with parameter totalSales of double
    data type
    DO
        ADD the value of the totalSales parameter to the totalSales attribute of
        Store class
    END DO

    CREATE a mutator method setTotalDiscount() with parameter totalDiscount of
    double data type
    DO
        ADD the value of the totalDiscount parameter to the totalDiscount attribute
        of Store class
    END DO

    CREATE a method display()
    DO
        PRINT the storeId
        PRINT the storeName
        PRINT the location
        PRINT the openingHour
        PRINT the totalSales

        IF totalSales and total discount is zero
        PRINT “No purchase is made.”
        END IF
    END DO
END DO
```

3.2. Pseudocode for Department Class

```
START

CREATE a subclass named Department that extends Store

DO

    DECLARE instance variable productName as String using private access
    modifier

    DECLARE instance variable markedPrice as double using private access
    modifier

    DECLARE instance variable sellingPrice as double using private access modifier

    DECLARE instance variable isInSales as Boolean using private access modifier


    CREATE a constructor that accepts the following parameters: int storeId, String
    storeName, String location, String openingHour, double totalSales, double
    totalDiscount, String productName, double markedPrice

    DO

        CALL super class constructor with parameters: int storeId, String
        storeName, String location, String openingHour

        CALL super class method setTotalSales with parameter totalSales

        CALL super class method setTotalDiscount with parameter totalDiscount

        SET isInSales to true

        SET sellingPrice to zero

        SET the value of productName to parameter productName

        SET the value of markedPrice to parameter markedPrice

    END DO
```

```
CREATE accessor methods for each attribute of Department class
DO
    CREATE a method getProductName() with return type String
    DO
        RETURN productName
    END DO

    CREATE a method getMarkedPrice() with return type double
    DO
        RETURN markedPrice
    END DO

    CREATE a method getSellingPrice() with return type double
    DO
        RETURN sellingPrice
    END DO

    CREATE a method getIsInSales() with return type boolean
    DO
        RETURN isInSales
    END DO
END DO
```

```
CREATE a mutator method setMarkedPrice() with parameter markedPrice of
double data type
DO
    SET the value of markedPrice to parameter markedPrice
END DO


CREATE a method calculateDiscountPrice() with parameter isInSales of boolean
data type and markedPrice of double data type
DO
    IF isInSales is true then
        IF markedPrice is greater than or equal to 5000
            SET sellingPrice to markedPrice subtracted by 20% of
            markedPrice
        ELSE IF markedPrice is greater than or equal to 3000
            SET sellingPrice to markedPrice subtracted by 10% of
            markedPrice
        ELSE IF markedPrice is greater than or equal to 1000
            SET sellingPrice to markedPrice subtracted by 5% of
            markedPrice
        ELSE
            SET sellingPrice to markedPrice
        END IF
    END IF
    CALL super class method setTotalDiscount and override parameter to
    markedPrice subtracted by sellingPrice
    CALL method setTotalSales and set parameter to sellingPrice
    SET isInSales to false
END DO
```

```
    OVERRIDE method display()
    DO
        CALL super class method display()
        IF isInSales is true then
            PRINT the productName
            PRINT the markedPrice
        ELSE
            PRINT the productName
            PRINT the sellingPrice
        END IF
    END DO
END DO
```

3.3. Pseudocode for Retailer Class

```
START
CREATE a subclass named Retailer that extends Store
DO
    DECLARE instance variable vatInclusivePrice as int using private access
    modifier
    DECLARE instance variable loyaltyPoint as int using private access modifier
    DECLARE instance variable isPaymentOnline as boolean using private access
    modifier
    DECLARE instance variable purchasedYear as String using private access
    modifier
```

```
CREATE a constructor that accepts the following parameters: int storeId, String
storeName, String location, String openingHour, double totalSales, double
totalDiscount, int vatInclusivePrice, boolean isPaymentOnline, String
purchasedYear

DO

    CALL super class constructor with parameters: int storeId, String
    storeName, String location, String openingHour

    CALL super class method setTotalSales with parameter totalSales

    CALL super class method setTotalDiscount with parameter totalDiscount

    SET the value of vatInclusivePrice to parameter vatInclusivePrice

    SET isPaymentOnline to false

    SET the value of purchasedYear to parameter purchasedYear

    SET loyaltyPoint to zero

END DO


CREATE accessor methods for each attribute of Retailer class

DO

    CREATE a method getVatInclusivePrice() with return type int

        DO

            RETURN vatInclusivePrice

        END DO


    CREATE a method getLoyaltyPoint() with return type int

    DO

        RETURN loyaltyPoint

    END DO
```

```
        CREATE a method getIsPaymentOnline() with return type boolean
        DO
            RETURN isPaymentOnline
        END DO

        CREATE a method getPurchasedYear() with return type String
        DO
            RETURN purchasedYear
        END DO
    END DO

    CREATE a mutator method setIsPaymentOnline() with parameter
    isPaymentOnline of boolean data type
    DO
        SET the value of isPaymentOnline to parameter isPaymentOnline
    END DO

    CREATE a method setLoyaltyPoint() with parameter isPaymentOnline of boolean
    data type and vatInclusivePrice of int data type
    DO
        IF isPaymentOnline is true then
            SET loyaltyPoint to 1% of vatInclusivePrice
        END IF
    END DO
    CREATE a method removeProduct()
    DO
        IF loyaltyPoint is zero and purchasedYear is equal to 2020 or 2021 or
        2022 then
            SET vatInclusivePrice to zero
            SET loyaltyPoint to zero
            SET isPaymentOnline to false
        END IF
    END DO

    OVERRIDE method display()
    DO
        IF loyaltyPoint is not zero and purchasedYear is not equal to 2020 or
        20201 or 2022 then
            CALL super class method display()
            PRINT vatInclusivePrice
            PRINT loyaltyPoint
            PRINT purchasedYear
        ELSE
            CALL super class method display()
            PRINT “Product has been removed.”
        END IF
    END DO
END DO
```

4. Method Description

4.1. Method description table for Store class

Method	Description
getStoreId()	Getter method of int data type that returns the attribute ‘storeId’.
getStoreName()	Getter method of String data type that returns the attribute ‘storeName’.
getLocation()	Getter method of String data type that returns the attribute ‘location’.
getOpeningHour()	Getter method of String data type that returns the attribute ‘openingHour’.
setTotalSales(double totalSales)	Setter method that sets the value of ‘totalSales’ attribute adding the previous total sales with new total sales using the parameter.
setTotalDiscount(double totalDiscount)	Setter method that sets the value of ‘totalDiscount’ attribute adding the previous total discount with new total discount using the parameter.
display()	Displays the details of Store (Store Id, Store name, Location, Opening hour, Total Sales).

Table 1 - Method description of Store class

4.2. Method description table for Department class

Method	Description
getProductName()	Getter method of String data type that returns the attribute ‘productName’.
getMarkedPrice()	Getter method of double data type that returns the attribute ‘markedPrice’.
getSellingPrice()	Getter method of double data type that returns the attribute ‘sellingPrice’.
getIsInSales()	Getter method of boolean data type that returns the attribute ‘isInSales’.
setMarkedPrice(double markedPrice)	Setter method that sets the value of attribute ‘markedPrice’ to the parameter of this method.
calculateDiscountPrice(Boolean isInSales, double markedPrice)	Setter method that sets total discount, total sales and isInSales using the given criteria.
display()	Displays the details of Store (Store Id, store name, location, opening hour, total sales) and displays details about Product (Product name, marked price/selling price).

Table 2 - Method description of Department class

4.3. Method for Retailer class

Method	Description
getVatInclusivePrice()	Getter method of int data type that returns attribute ‘vatInclusivePrice’.
getLoyaltyPoint()	Getter method of int data type that returns attribute ‘loyaltyPoint’.
getIsPaymentOnline()	Getter method of boolean data type that returns attribute ‘isPaymentOnline’.
getPurchasedYear()	Getter method of String data type that returns attribute ‘purchasedYear’.
setIsPaymentOnline()	Setter method that sets the value of attribute ‘isPaymentOnline’ to the parameter of this method.
setLoyaltyPoint()	Setter method that sets loyalty point using the given criteria.
removeProduct()	Can be called to remove a certain product if it fulfils a certain criteria.
display()	Displays the details of Store (Store Id, store name, location, opening hour, total sales) and displays the details of Retailer (Vat inclusive price, loyalty point, purchased year) or either if the product has been removed on the basis of the given criteria.

Table 3 - Method description of Retailer class

5. Testing

5.1. Test 1

Inspect Department class, calculate discount price and reinspect the Department class

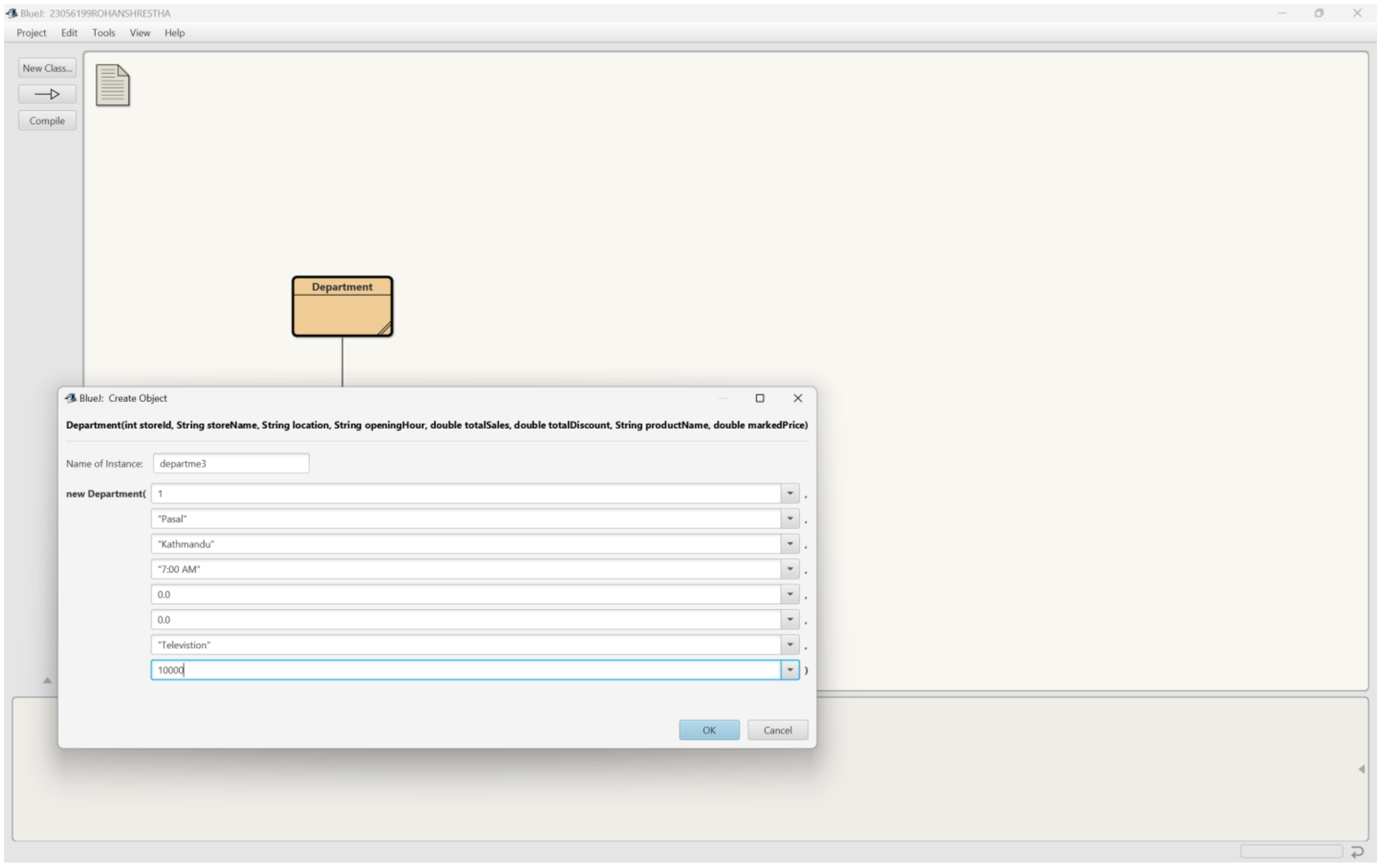
Test No:	
Objective:	To Inspect Department class, calculate discount price and reinspect the Department class.

Action:	→The Department is called with the following arguments:
	storeId = 1
	storeName = “Pasal”
	location = “Kathmandu”
	openingHour = “7:00 AM”
	totalSales = 0.0
	totalDiscount = 0.0
	productName = “Television”
	markedPrice = 10000
	→Inspection of Department class
	→void calculateDiscountPrice is called with following arguments:
	isInSales = true
	markedPrice = 10000
	→Re-inspection of Retailer class

Expected Result:	The selling price will be set according to the criteria given along with total sales and total discount.
Actual Result:	The selling price was set accordingly along with total sales and total discount.
Conclusion:	The test is successful.

Table 4 - Table of Test 1

Output Result:

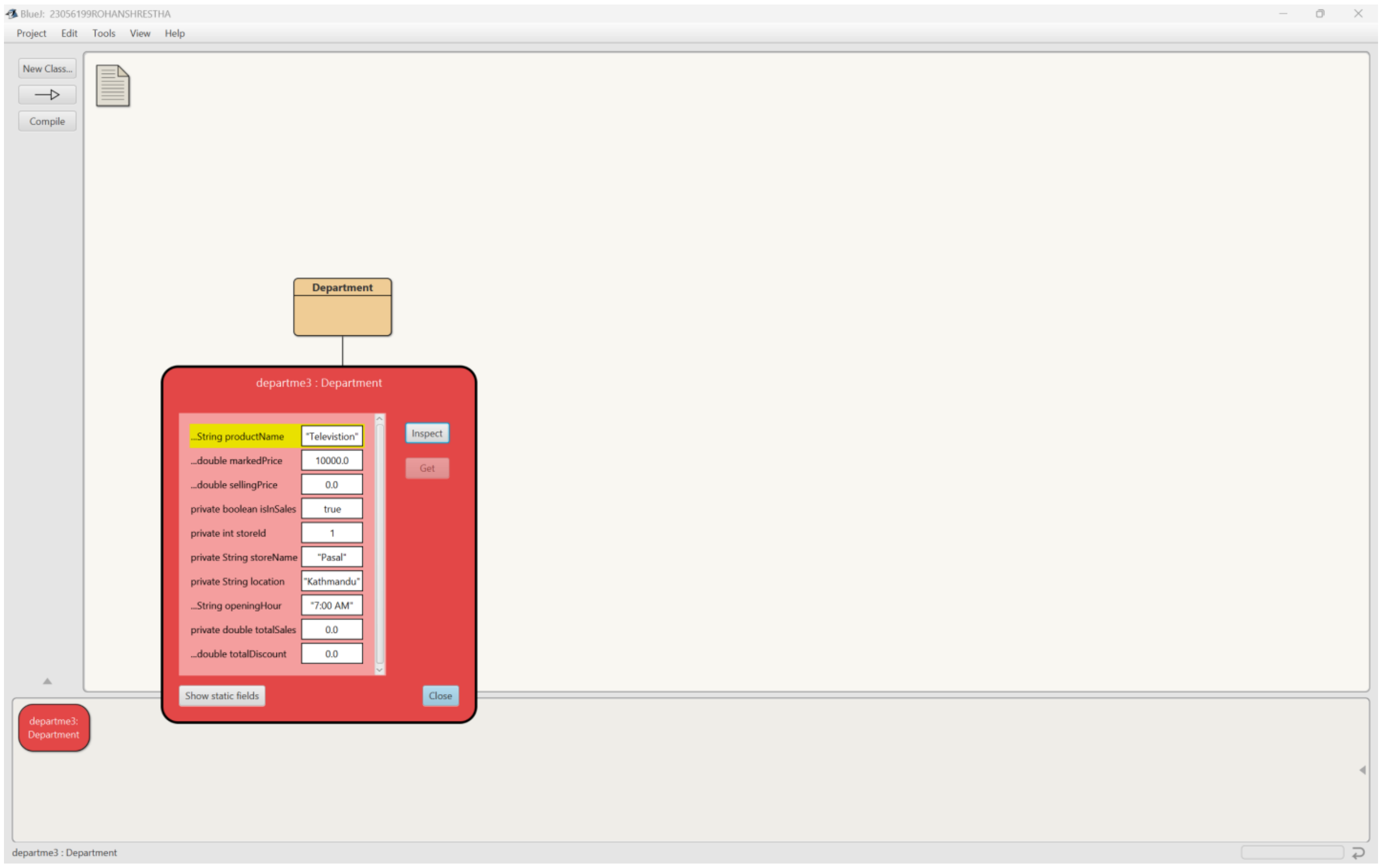


Department(int storeId, String storeName, String location, String openingHour, double totalSales, double totalDiscount, String productName, double markedPrice)

Name of Instance: departme3

new Department(1 ,
"Pasal" ,
"Kathmandu" ,
"7:00 AM" ,
0.0 ,
0.0 ,
"Televistion" ,
10000)

Figure 9 - Screenshot of assigning the values in Department class

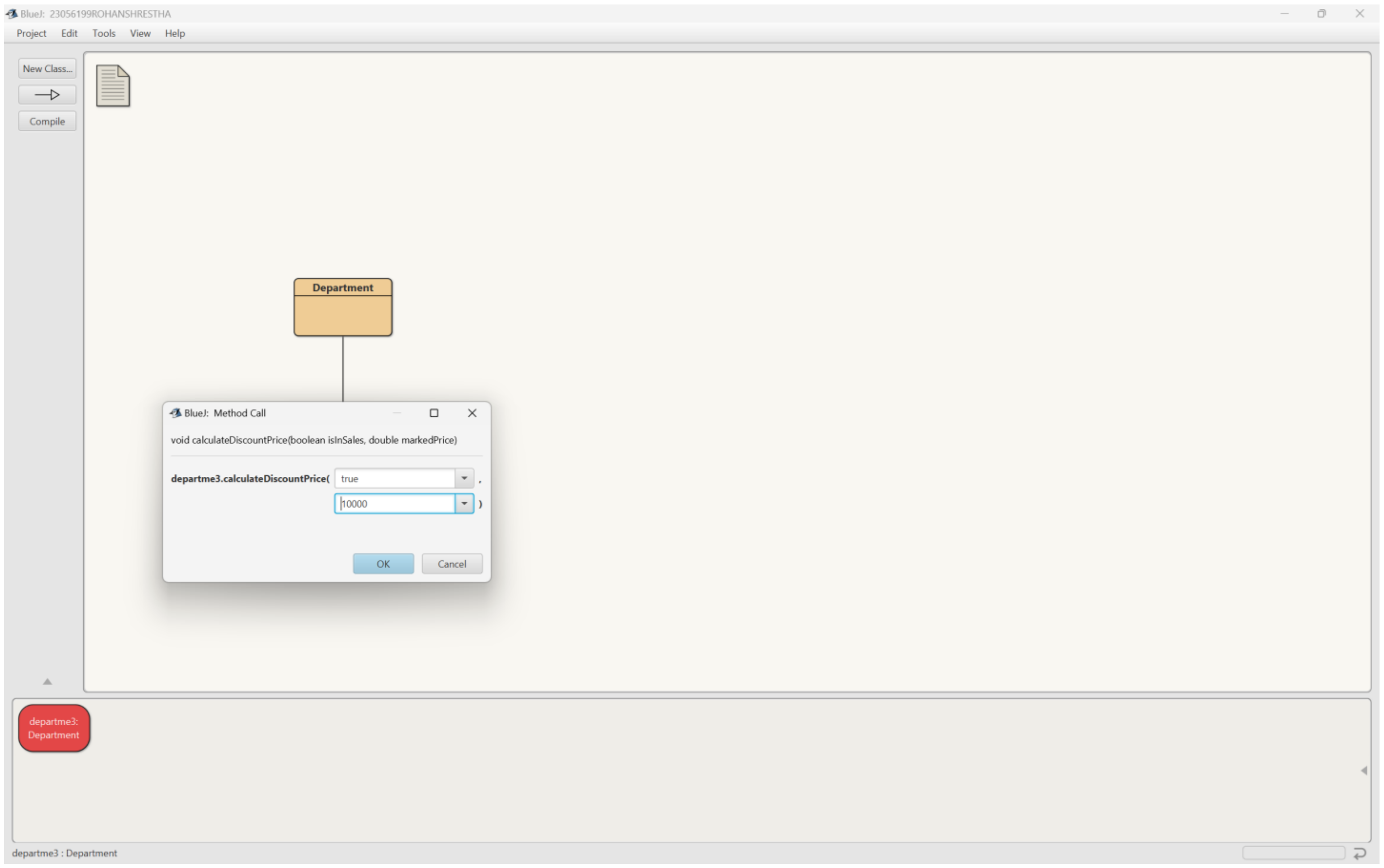


departme3 : Department

...String productName	"Televistion"
...double markedPrice	10000.0
...double sellingPrice	0.0
private boolean isInSales	true
private int storeId	1
private String storeName	"Pasal"
private String location	"Kathmandu"
...String openingHour	"7:00 AM"
private double totalSales	0.0
...double totalDiscount	0.0

departme3 : Department

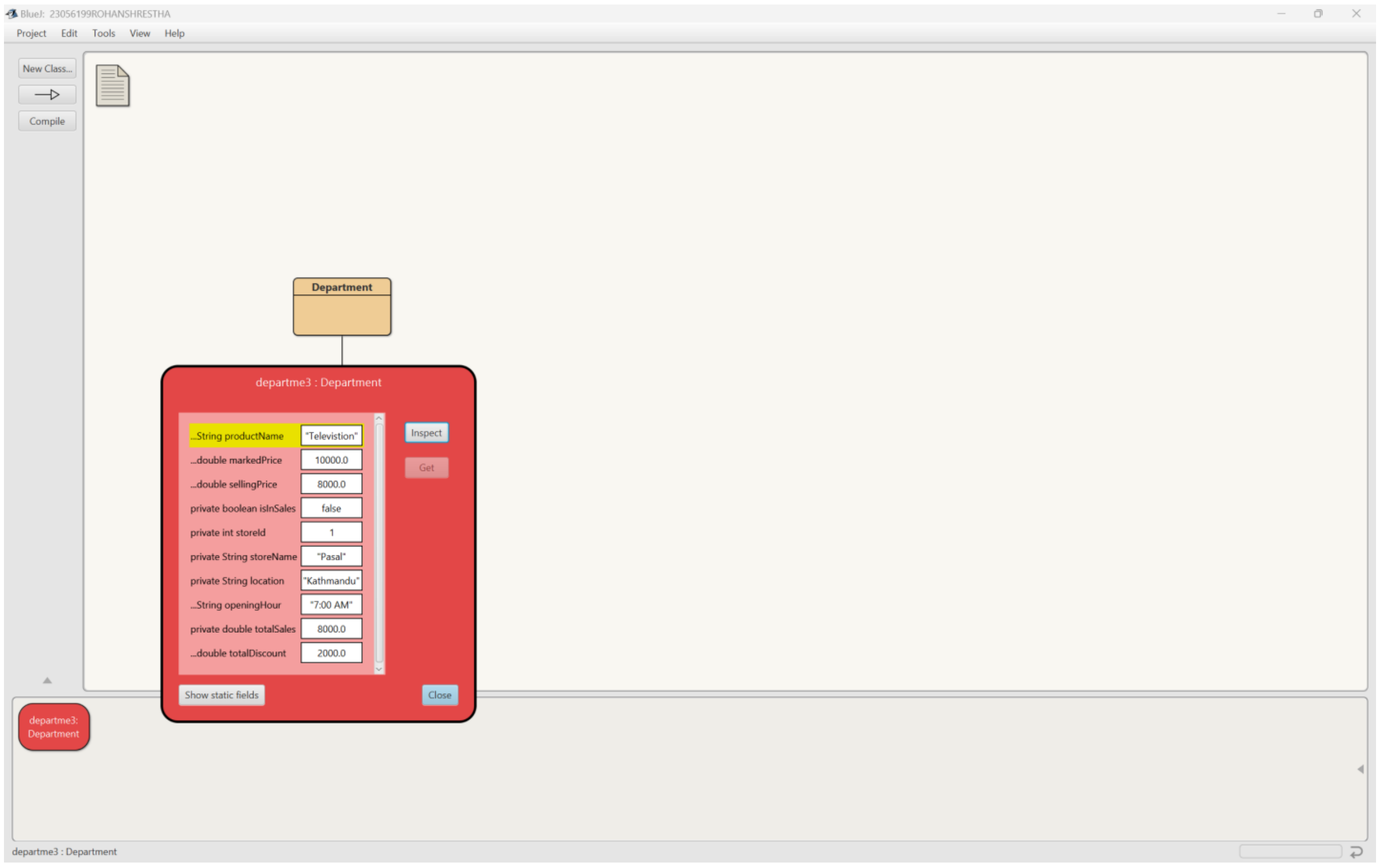
Figure 10 - Screenshot of inspection of Department class



void calculateDiscountPrice(boolean isInSales, double markedPrice)

departme3.calculateDiscountPrice(true , 10000)

Figure 11 - Screenshot of assigning the value in method calculateDiscountPrice()



departme3 : Department

...String productName	"Televistion"
...double markedPrice	10000.0
...double sellingPrice	8000.0
private boolean isInSales	false
private int storeId	1
private String storeName	"Pasal"
private String location	"Kathmandu"
...String openingHour	"7:00 AM"
private double totalSales	8000.0
...double totalDiscount	2000.0

departme3 : Department

Figure 12 - Screenshot of re-inspection of Department class

5.2. Test 2

Inspect Retailer class, set loyalty point and reinspect the Retailer class

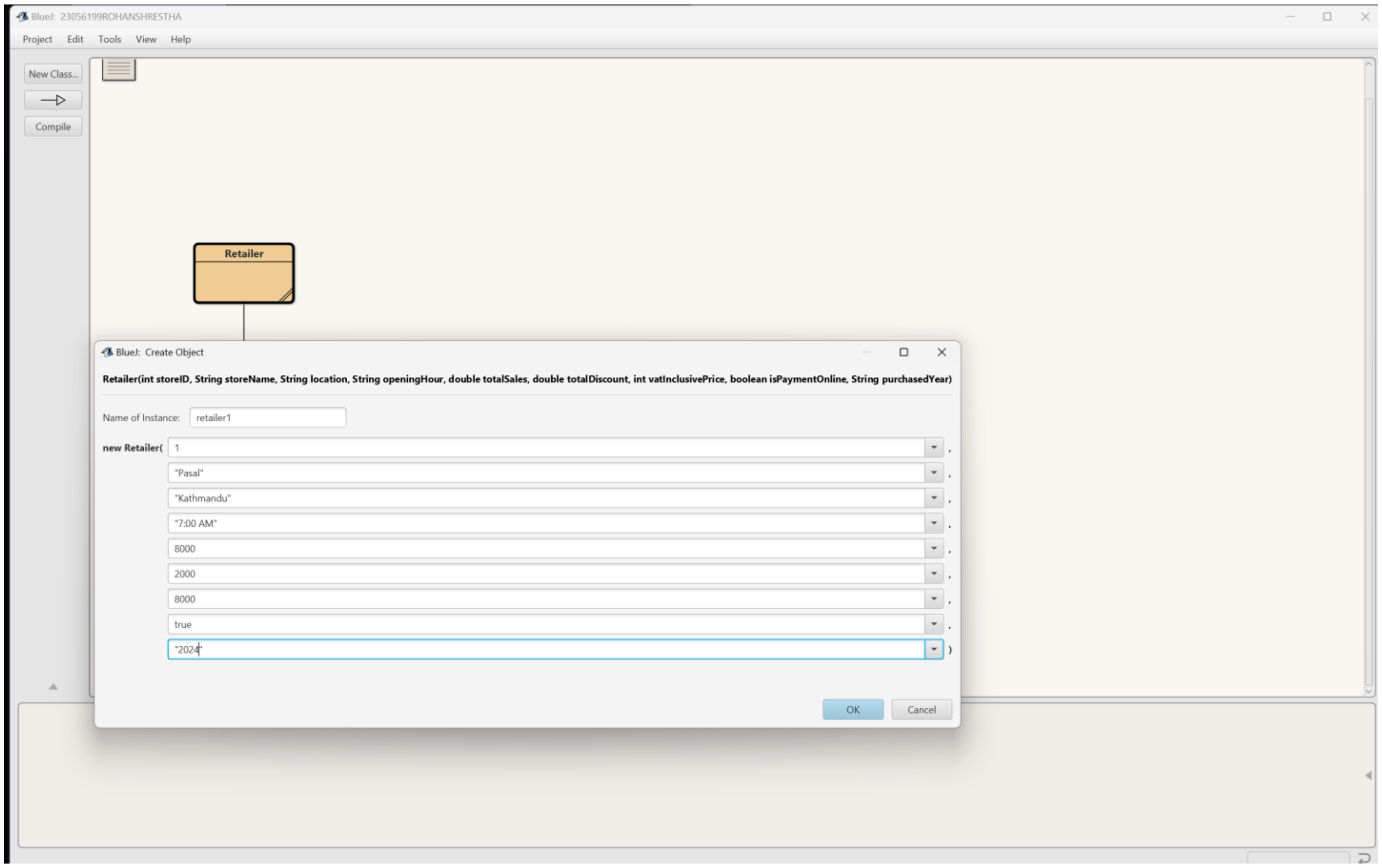
Test No:	**2**
Objective:	To Inspect Retailer class, set loyalty point and reinspect the Retailer class

Action:	→The Retailer class is called with the following arguments:
	storeId = 1
	storeName = “Pasal”
	location = “Kathmandu”
	openingHour = “7:00 AM”
	totalSales = 8000
	totalDiscount = 2000
	vatInclusivePrice = 8000
	isPaymentOnline = true
	purchasedYear = “2024”
	→Inspection of Retailer class
	→void setLoyaltyPoint is called with following arguments:
	isPaymentOnline = true
	vatInclusivePrice = 8000
	→Re-inspection of Retailer class

Expected Result:	The loyalty point will be set on the basis of vat inclusive price.
Actual Result:	The loyalty point was set on the basis of vat inclusive price.

Table 5 - Table of Test 2

Output Result:

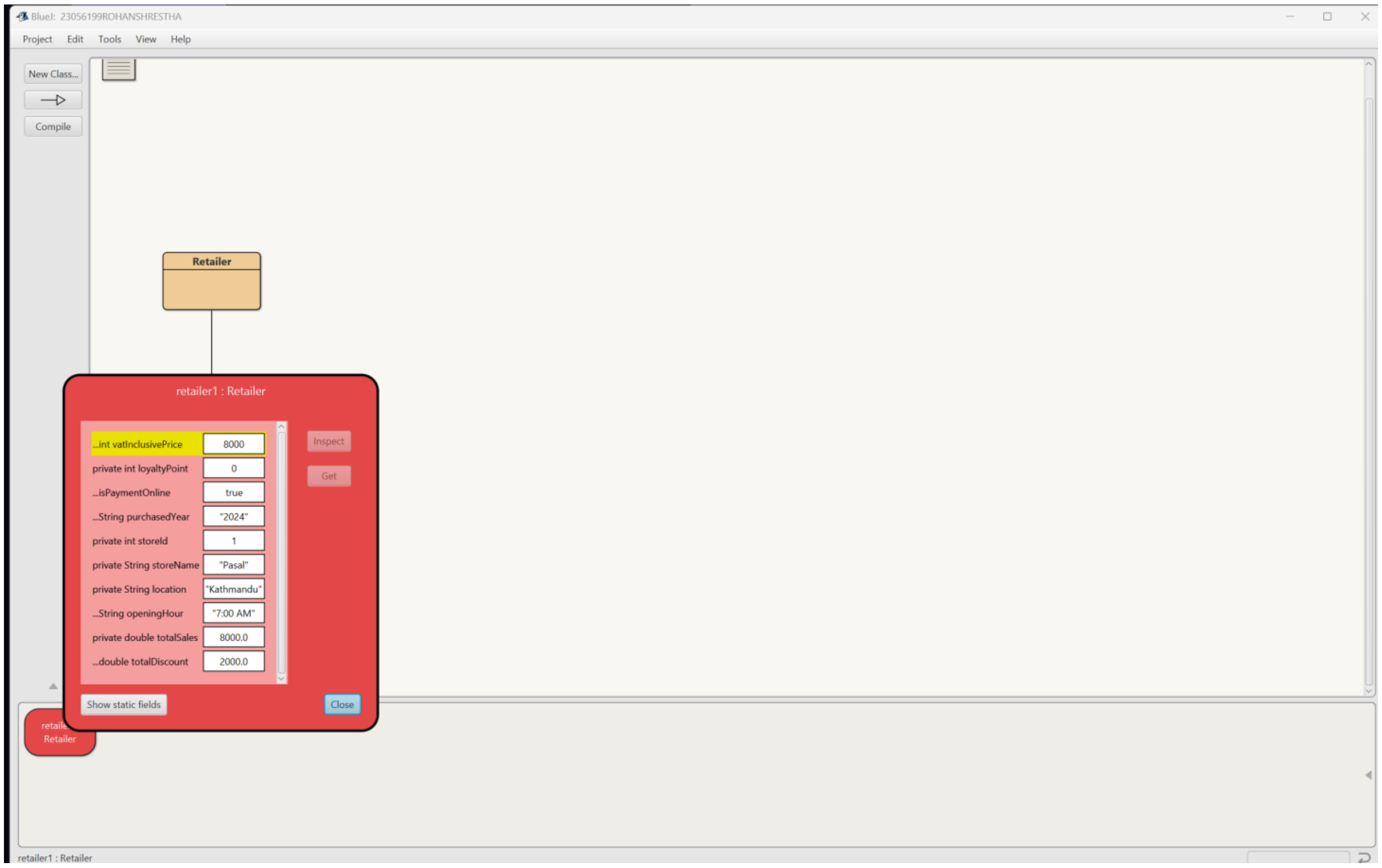


Retailer(int storeID, String storeName, String location, String openingHour, double totalSales, double totalDiscount, int vatInclusivePrice, boolean isPaymentOnline, String purchasedYear)

Name of Instance: retailer1

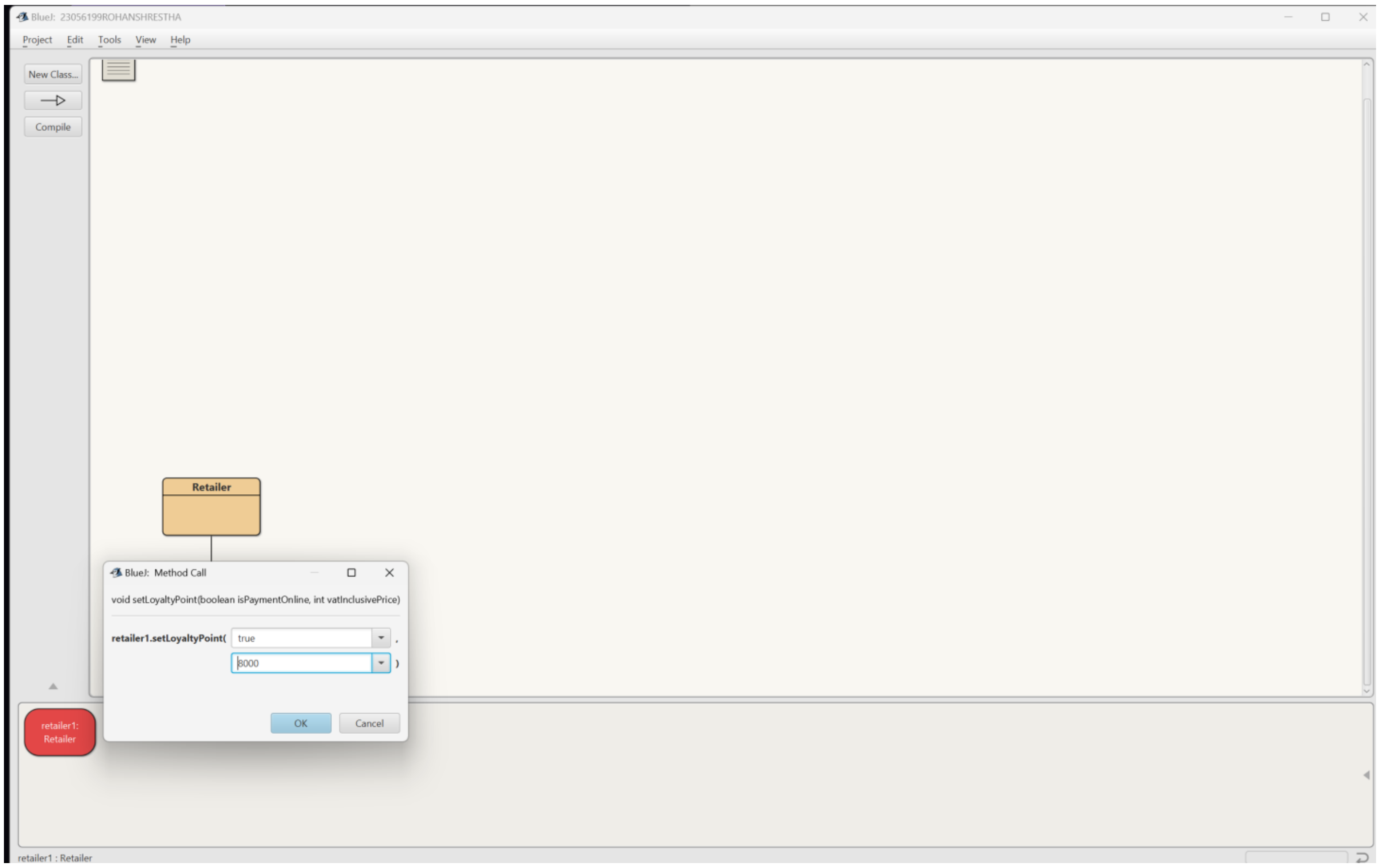
```
new Retailer( 1 ,
              "Pasal" ,
              "Kathmandu" ,
              "7:00 AM" ,
              8000 ,
              2000 ,
              8000 ,
              true ,
              "2024" )
```

Figure 13 - Screenshot of assigning the values in Retailer class



...int vatInclusivePrice	8000
private int loyaltyPoint	0
...isPaymentOnline	true
...String purchasedYear	"2024"
private int storeId	1
private String storeName	"Pasal"
private String location	"Kathmandu"
...String openingHour	"7:00 AM"
private double totalSales	8000.0
...double totalDiscount	2000.0

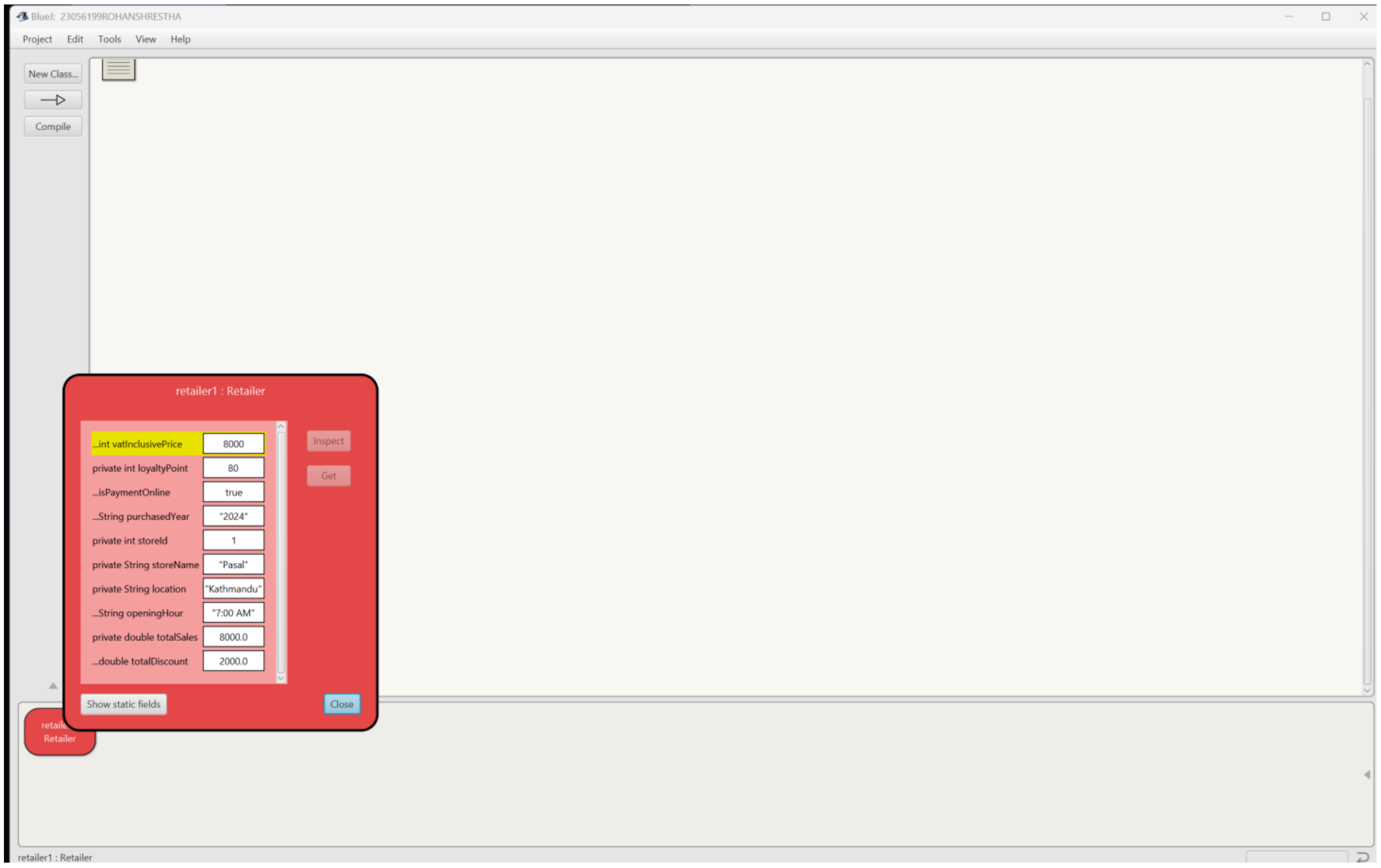
Figure 14 - Screenshot of inspection of Retailer class



void setLoyaltyPoint(boolean isPaymentOnline, int vatInclusivePrice)

retailer1.setLoyaltyPoint(true , 8000)

Figure 15 - Screenshot of assigning the value in method setLoyaltyPoint()



retailer1 : Retailer

...int vatInclusivePrice	8000
private int loyaltyPoint	80
...isPaymentOnline	true
...String purchasedYear	"2024"
private int storeId	1
private String storeName	"Pasal"
private String location	"Kathmandu"
...String openingHour	"7:00 AM"
private double totalSales	8000.0
...double totalDiscount	2000.0

retailer1 : Retailer

Figure 16 - Screenshot of re-inspection of Retailer class

5.3. Test 3

Inspect Retailer class again after removing the product.

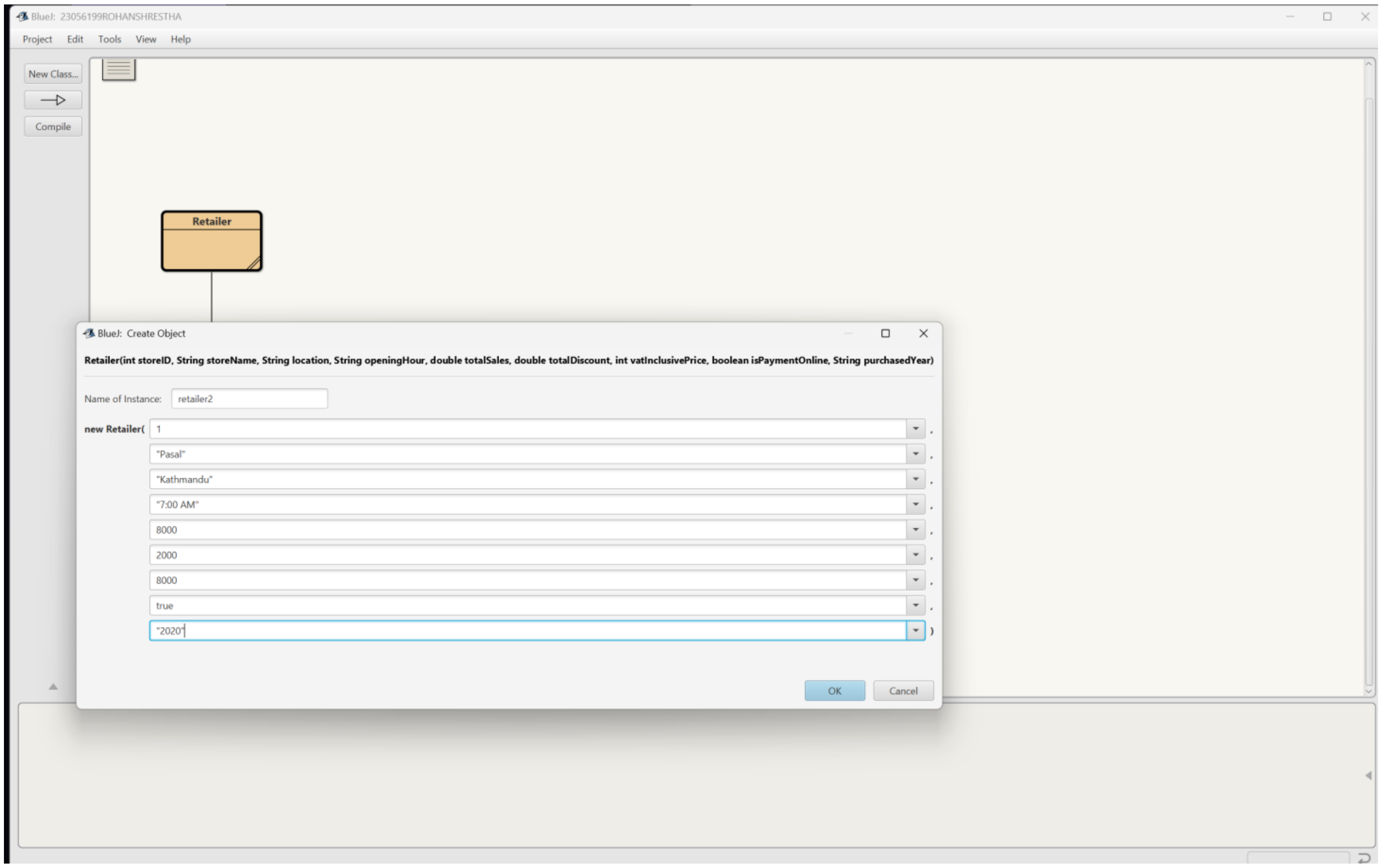
Test No:	**3**
Objective:	To Inspect Retailer class again after removing the product.

Action:	→The Retailer class is called with the following arguments:
	storeId = 1
	storeName = “Pasal”
	location = “Kathmandu”
	openingHour = “7:00 AM”
	totalSales = 8000
	totalDiscount = 2000
	vatInclusivePrice = 8000
	isPaymentOnline = true
	purchasedYear = “2020”
	→Inspection of Retailer class
	→void removeProduct is called
	→Re-inspection of Retailer class

Expected Result:	The vat inclusive price, loyalty point is set to zero and payment online is set to false since the product is removed.
Actual Result:	The vat inclusive price, loyalty point is set to zero and payment online is set to false.

Table 6 - Table for Test 3

Output Result:

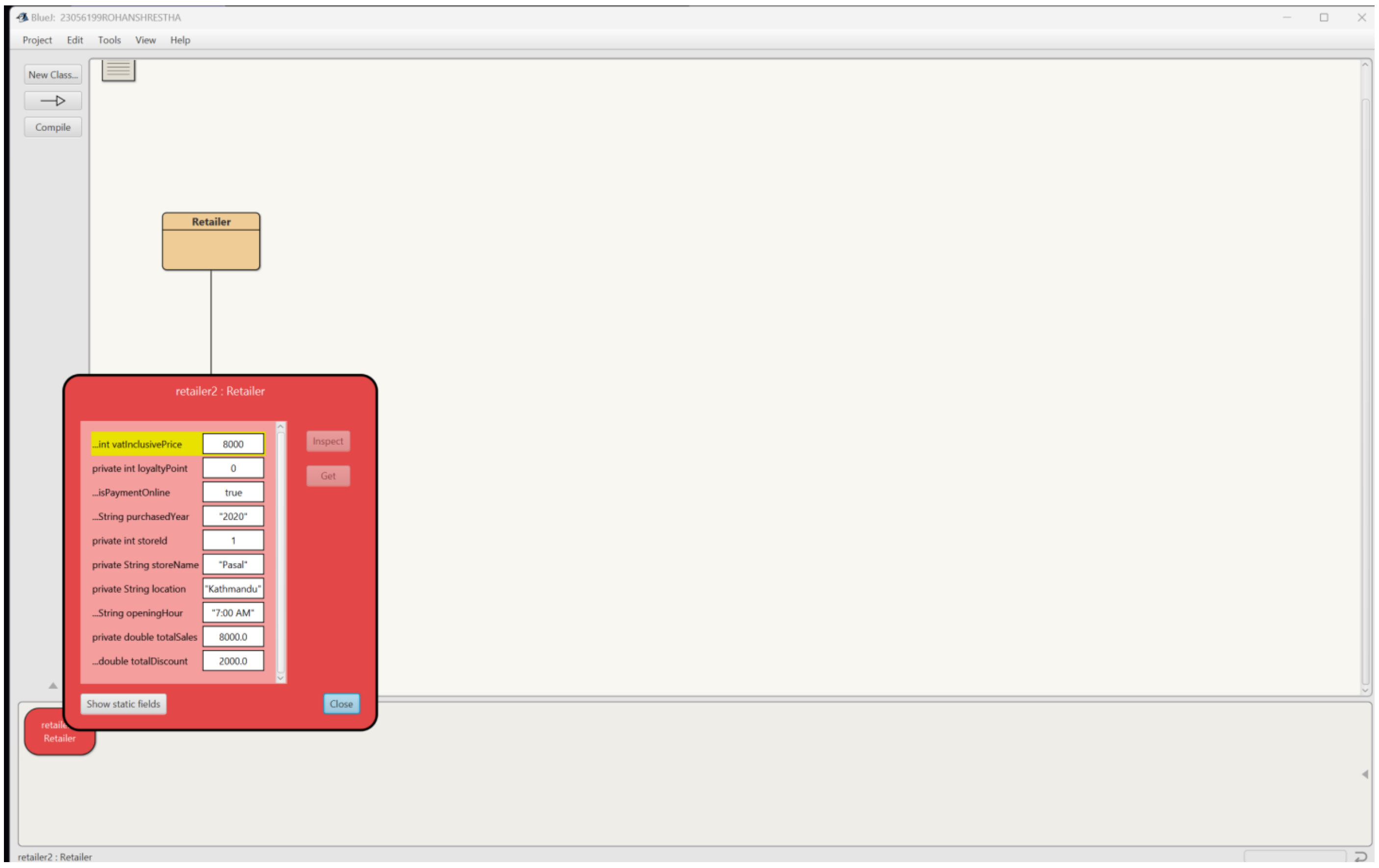


Retailer(int storeID, String storeName, String location, String openingHour, double totalSales, double totalDiscount, int vatInclusivePrice, boolean isPaymentOnline, String purchasedYear)

Name of Instance: retailer2

new Retailer(1,
"Pasal",
"Kathmandu",
"7:00 AM",
8000,
2000,
8000,
true,
"2020")

Figure 17 - Screenshot of assigning the value of Retailer class



retailer2 : Retailer

...int vatInclusivePrice	8000
private int loyaltyPoint	0
...isPaymentOnline	true
...String purchasedYear	"2020"
private int storeId	1
private String storeName	"Pasal"
private String location	"Kathmandu"
...String openingHour	"7:00 AM"
private double totalSales	8000.0
...double totalDiscount	2000.0

retailer2 : Retailer

Figure 18 - Screenshot of inspecting the Retailer class

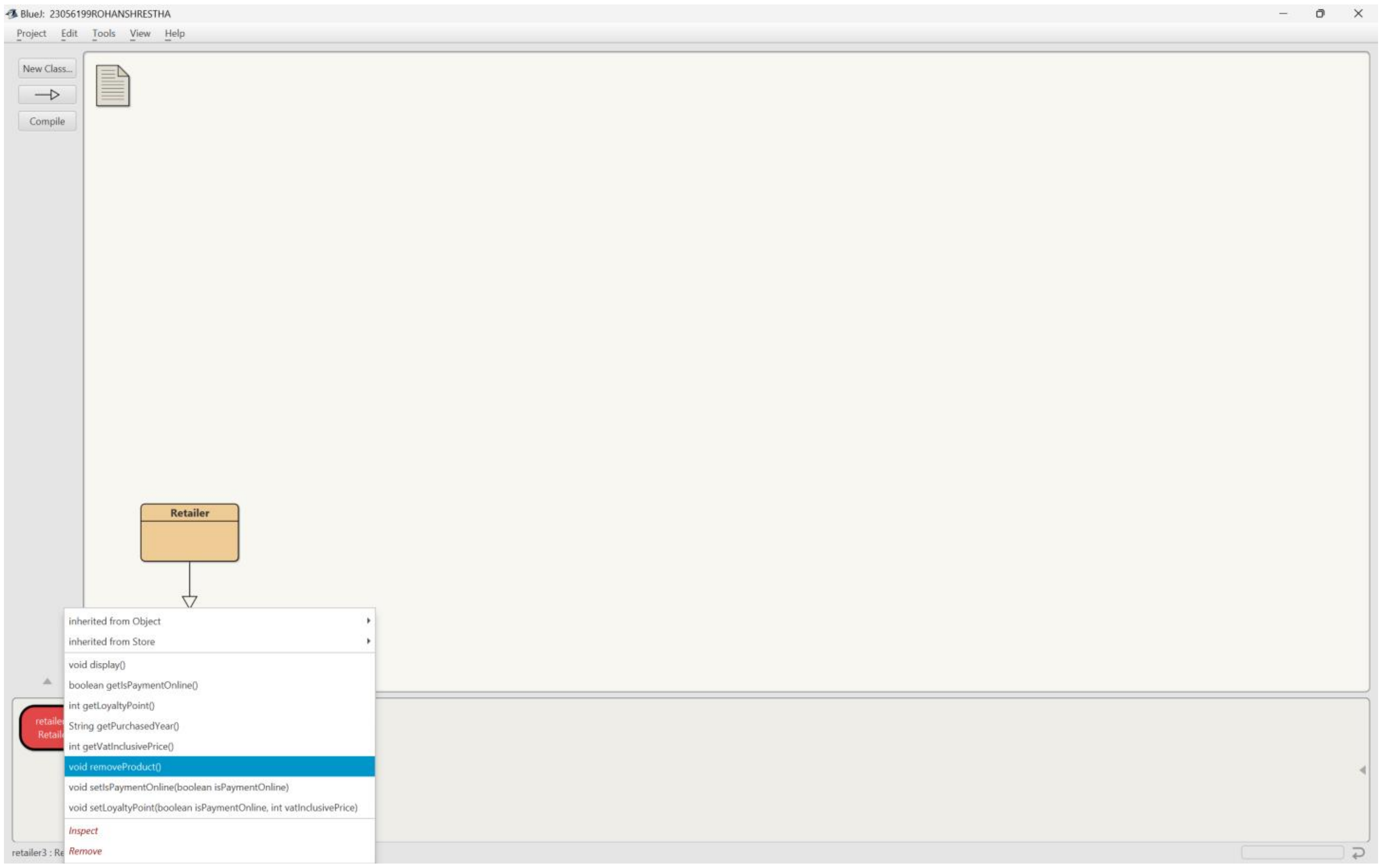
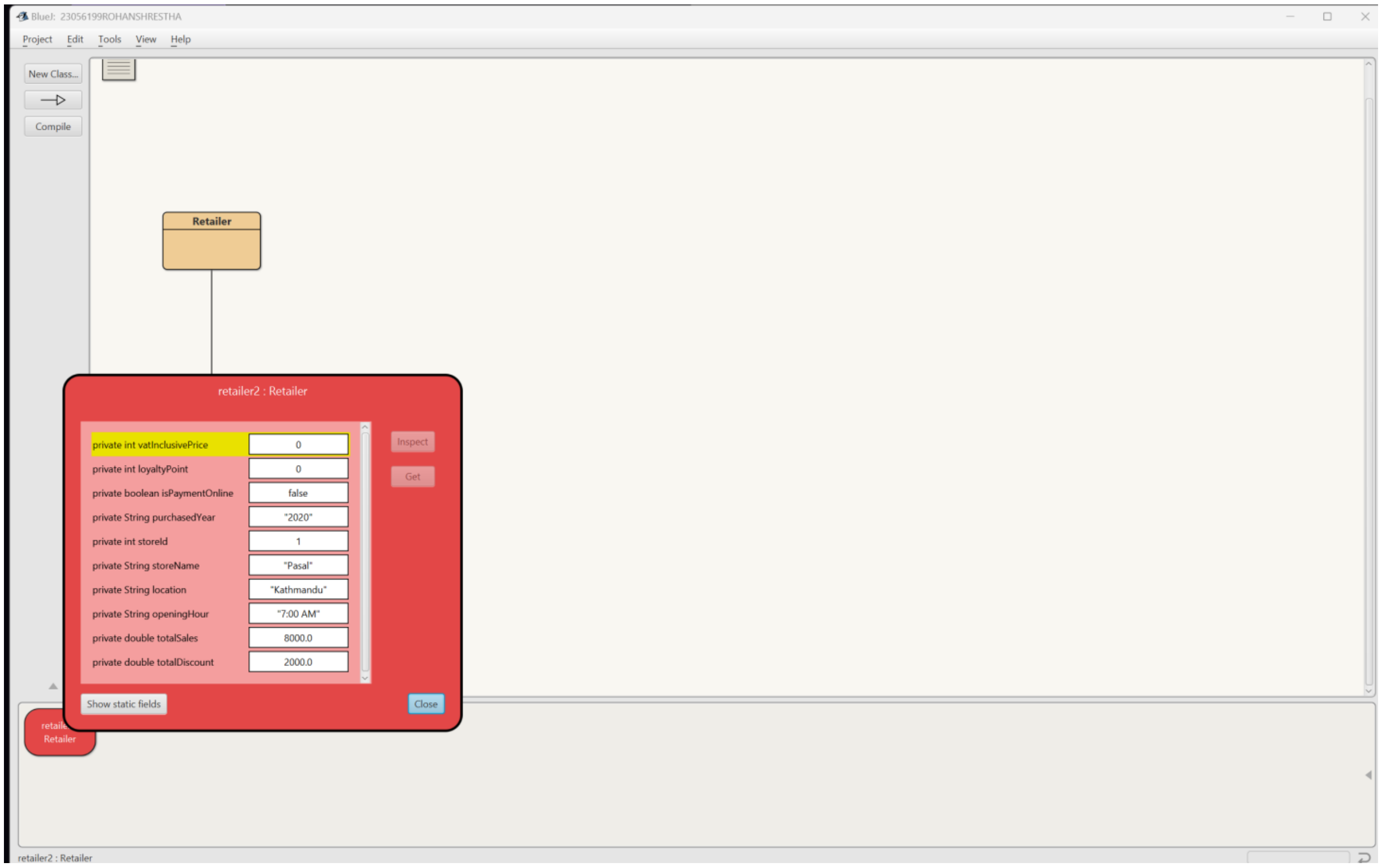

BlueJ: 23056199ROHANSHRESTHA
Project Edit Tools View Help
New Class...
Compile
Retailer
inherited from Object
inherited from Store
void display()
boolean getIsPaymentOnline()
int getLoyaltyPoint()
String getPurchasedYear()
int getVatInclusivePrice()
void removeProduct()
void setIsPaymentOnline(boolean isPaymentOnline)
void setLoyaltyPoint(boolean isPaymentOnline, int vatInclusivePrice)
Inspect
Remove
retailer3 : Re

Figure 19 - Screenshot of calling the method removeProduct()



retailer2 : Retailer

private int vatInclusivePrice	0
private int loyaltyPoint	0
private boolean isPaymentOnline	false
private String purchasedYear	"2020"
private int storeId	1
private String storeName	"Pasal"
private String location	"Kathmandu"
private String openingHour	"7:00 AM"
private double totalSales	8000.0
private double totalDiscount	2000.0

retailer2 : Retailer

Figure 20 - Screenshot of re-inspecting the Retailer class

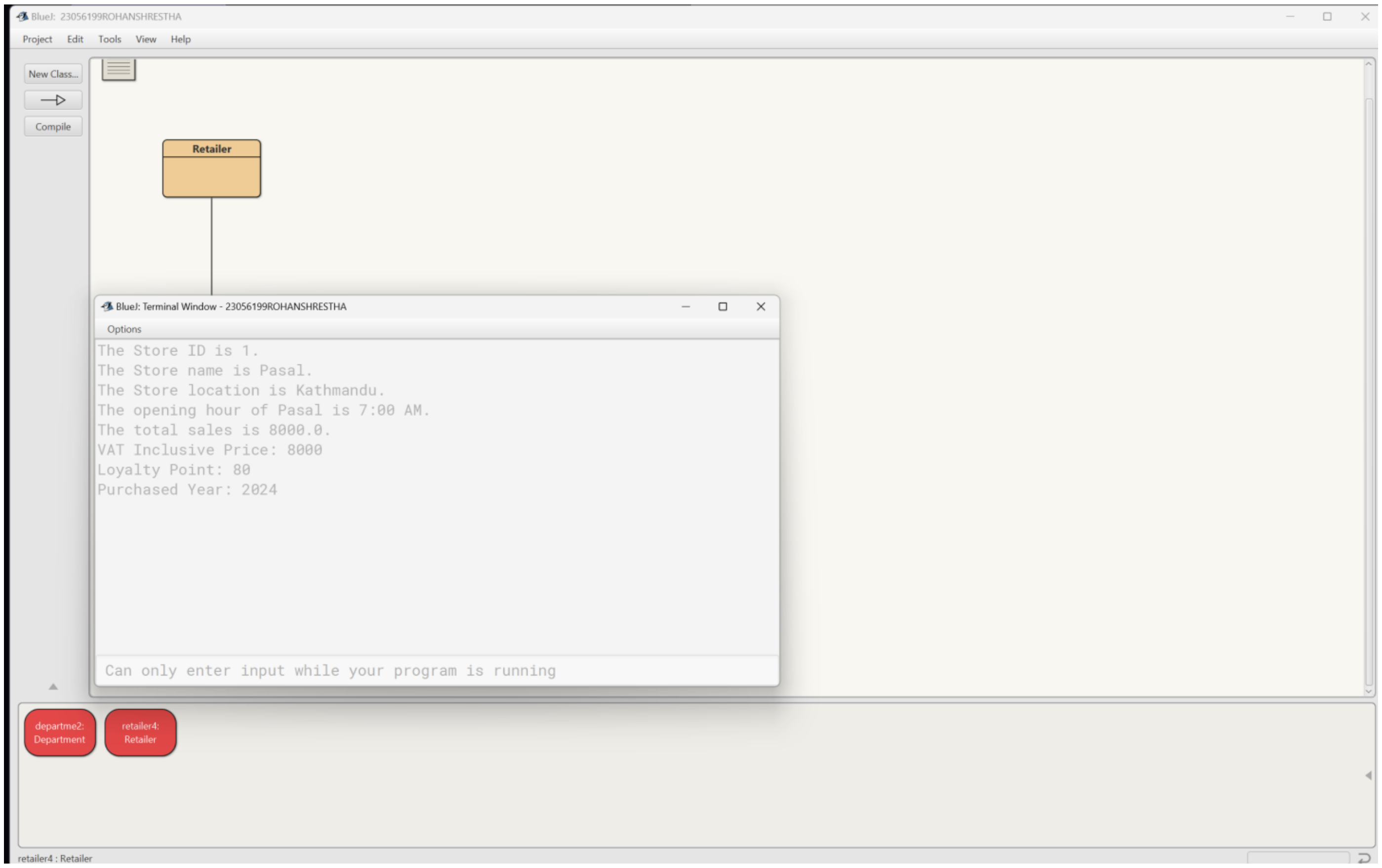
5.4. Test 4

Display the details of Department and Retailer classes.

Test No:	**4**
Objective:	To Display the details of Department and Retailer classes.
Action:	→void display is called in Department class →void display is called in Retailer class
Expected Result:	The selling price will be set according to the criteria given and details will be displayed in a new blueJ terminal window and loyalty points will also be set according to the criteria given and all the details will be displayed in a new blueJ terminal window.
Actual Result:	Everything is displayed accordingly.

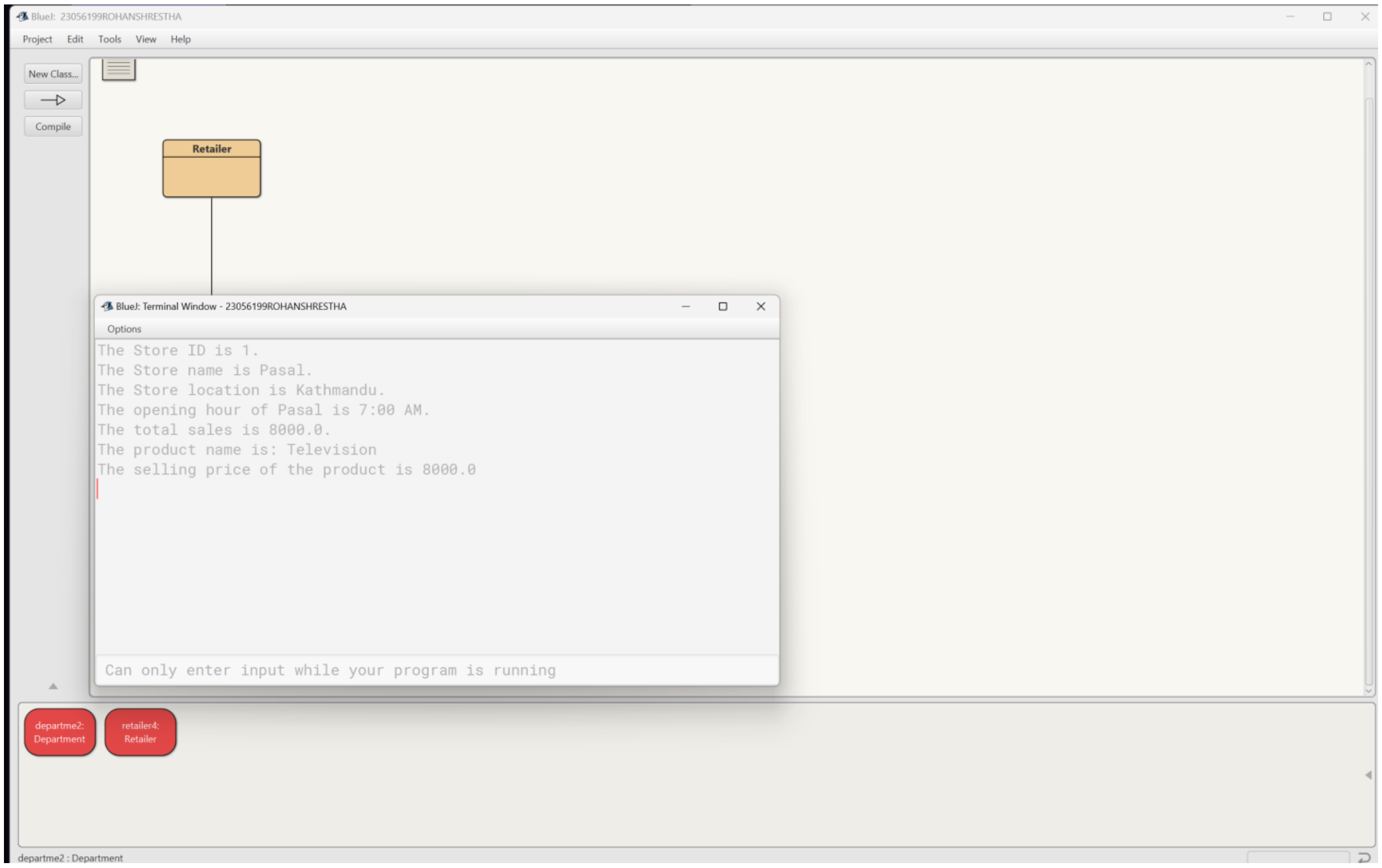
Table 7 - Table of Test 4

Output Result:


BlueJ: 23056199ROHANSHRESTHA
Project Edit Tools View Help
New Class...
Compile
Retailer
BlueJ: Terminal Window - 23056199ROHANSHRESTHA
Options
departme2: Department
retailer4: Retailer
retailer4 : Retailer

```
The Store ID is 1.
The Store name is Pasal.
The Store location is Kathmandu.
The opening hour of Pasal is 7:00 AM.
The total sales is 8000.0.
VAT Inclusive Price: 8000
Loyalty Point: 80
Purchased Year: 2024

Can only enter input while your program is running
```

Figure 21 - Screenshot after calling display() method in Department class


BlueJ: 23056199ROHANSHRESTHA
Project Edit Tools View Help
New Class...
Compile
Retailer
BlueJ: Terminal Window - 23056199ROHANSHRESTHA
Options
departme2: Department
retailer4: Retailer
departme2 : Department

```
The Store ID is 1.
The Store name is Pasal.
The Store location is Kathmandu.
The opening hour of Pasal is 7:00 AM.
The total sales is 8000.0.
The product name is: Television
The selling price of the product is 8000.0
```

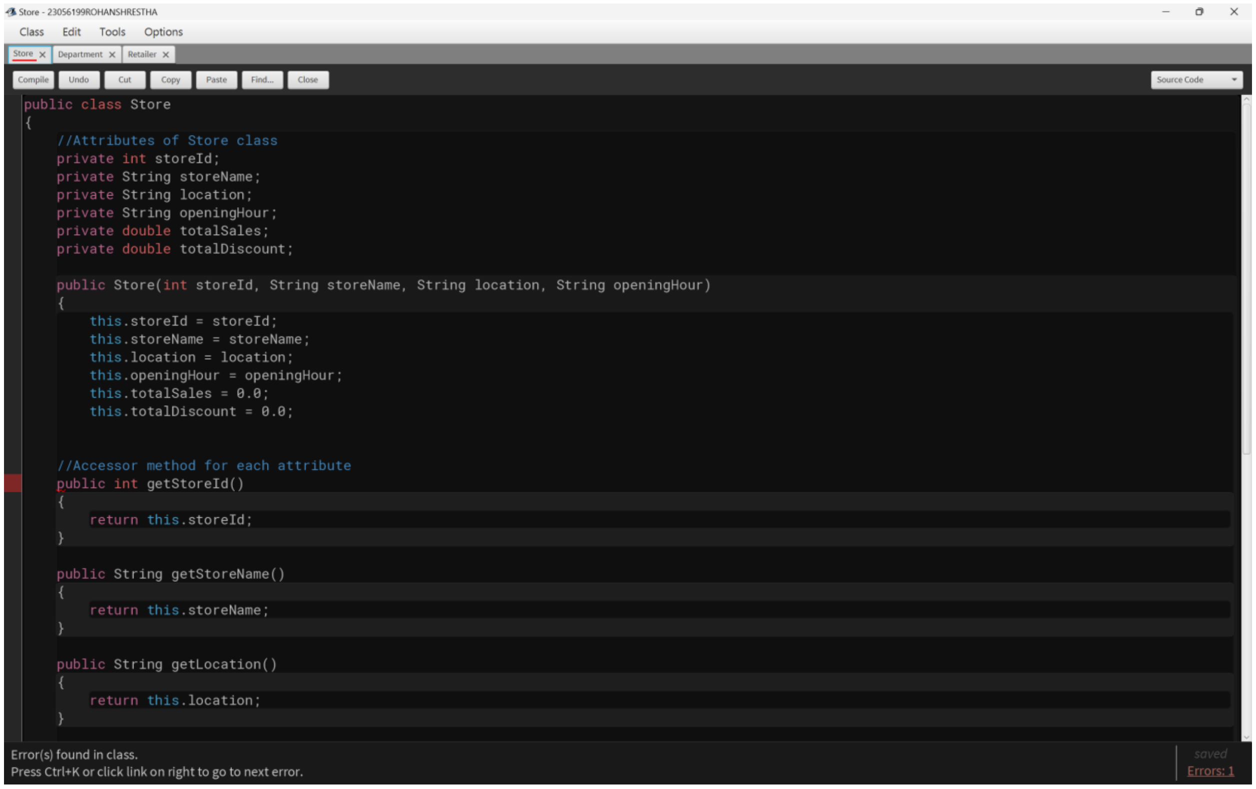
```
Can only enter input while your program is running
```

Figure 22 - Screenshot after calling display() method in Retailer class

6. Error Detection and Correction

6.1. Syntax Error

A syntax error in computer science is an error in the syntax of a coding or programming language, entered by a programmer. Syntax errors are caught by a software program called a compiler, and the programmer must fix them before the program is compiled and then run. (Technopedio, 2023)

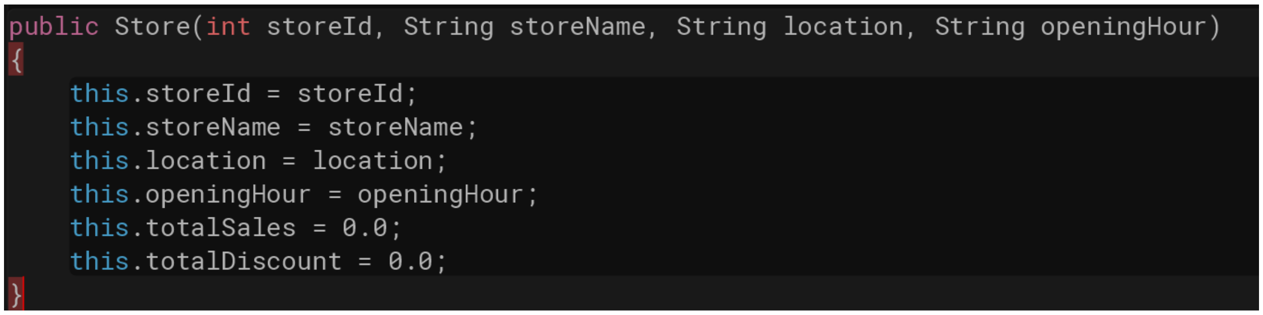

Store - 23056199ROHANSHRESTHA
Class Edit Tools Options
Store Department Retailer
Compile Undo Cut Copy Paste Find... Close
Source Code
saved
Errors: 1

```
public class Store
{
    //Attributes of Store class
    private int storeId;
    private String storeName;
    private String location;
    private String openingHour;
    private double totalSales;
    private double totalDiscount;

    public Store(int storeId, String storeName, String location, String openingHour)
    {
        this.storeId = storeId;
        this.storeName = storeName;
        this.location = location;
        this.openingHour = openingHour;
        this.totalSales = 0.0;
        this.totalDiscount = 0.0;


    //Accessor method for each attribute
    public int getStoreId()
    {
        return this.storeId;
    }

    public String getStoreName()
    {
        return this.storeName;
    }

    public String getLocation()
    {
        return this.location;
    }
```

```
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
```

Figure 23 - Screenshot of Syntax error

The error in the code above is caused by a missing brace which is a syntax error. The error is then corrected by adding the missing brace.

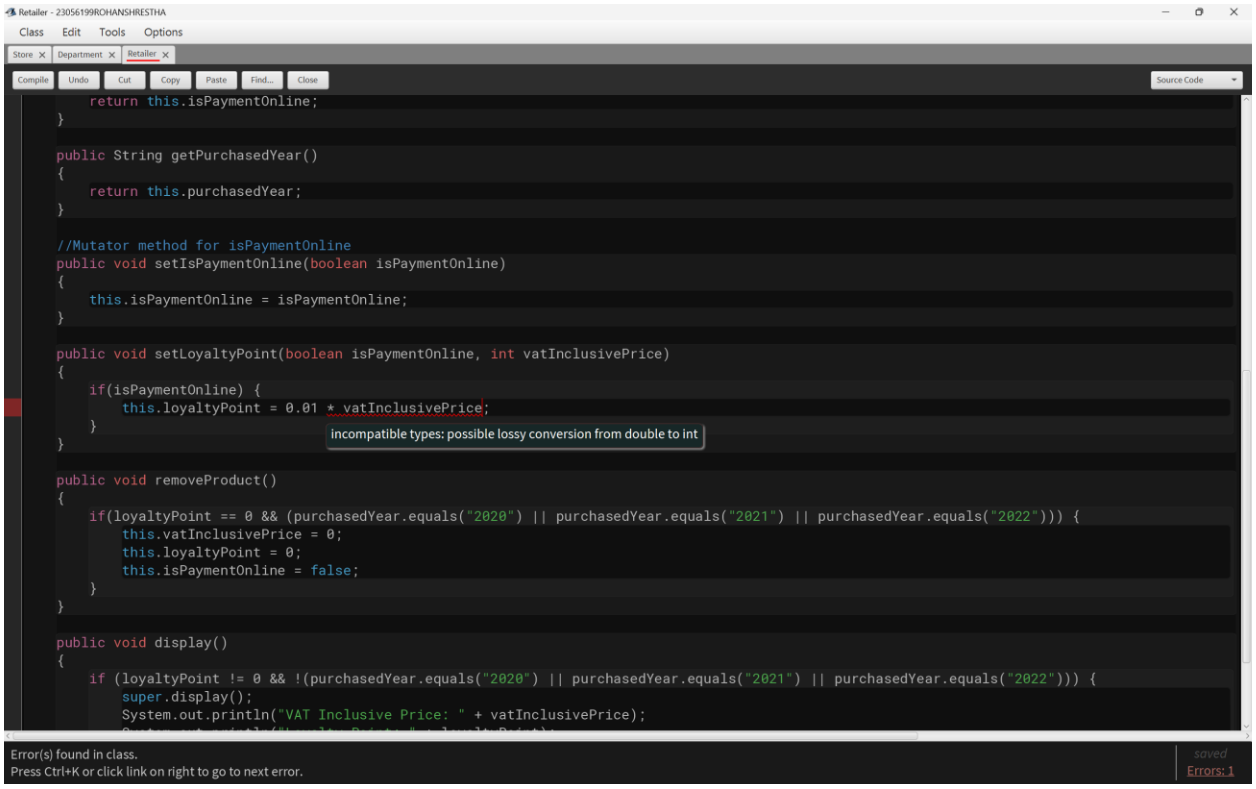


```
public Store(int storeId, String storeName, String location, String openingHour)
{
    this.storeId = storeId;
    this.storeName = storeName;
    this.location = location;
    this.openingHour = openingHour;
    this.totalSales = 0.0;
    this.totalDiscount = 0.0;
}
```

Figure 24 - Screenshot of fixing the Syntax error

6.2. Semantic Error

A semantic error is a problem in your code that prevents the interpreter from understanding it. There may not be anything wrong with the logic you've written, but it will cause the program to crash the way you've written. (Medium, 2023)



```
        return this.isPaymentOnline;
    }

    public String getPurchasedYear()
    {
        return this.purchasedYear;
    }

    //Mutator method for isPaymentOnline
    public void setIsPaymentOnline(boolean isPaymentOnline)
    {
        this.isPaymentOnline = isPaymentOnline;
    }

    public void setLoyaltyPoint(boolean isPaymentOnline, int vatInclusivePrice)
    {
        if(isPaymentOnline) {
            this.loyaltyPoint = 0.01 * vatInclusivePrice;
        }
    }

    public void removeProduct()
    {
        if(loyaltyPoint == 0 && (purchasedYear.equals("2020") || purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
            this.vatInclusivePrice = 0;
            this.loyaltyPoint = 0;
            this.isPaymentOnline = false;
        }
    }

    public void display()
    {
        if (loyaltyPoint != 0 && !(purchasedYear.equals("2020") || purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
            super.display();
            System.out.println("VAT Inclusive Price: " + vatInclusivePrice);
```

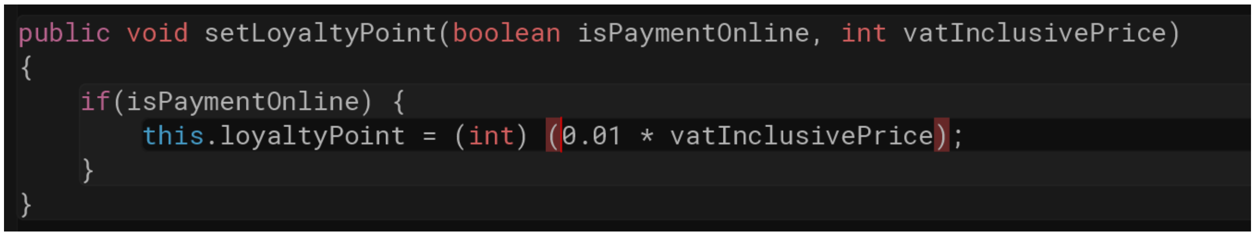
incompatible types: possible lossy conversion from double to int

Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.

saved
Errors: 1

Figure 25 - Screenshot of Semantic error

In the code above double value is being set in an int data type variable which is a semantic error. The error is then corrected by adding int keyword before the value being set.

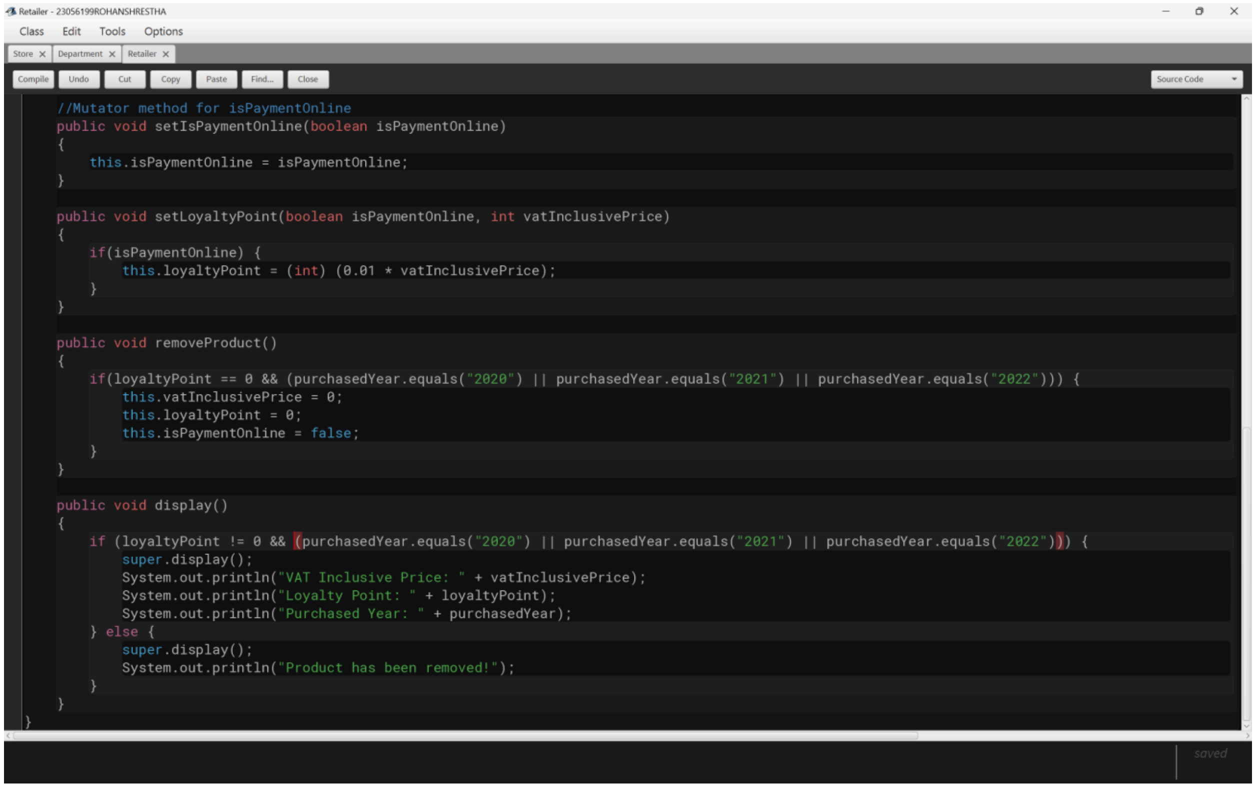


```
public void setLoyaltyPoint(boolean isPaymentOnline, int vatInclusivePrice)
{
    if(isPaymentOnline) {
        this.loyaltyPoint = (int) (0.01 * vatInclusivePrice);
    }
}
```

Figure 26 - Screenshot of fixing Semantic error

6.3. Logical Error

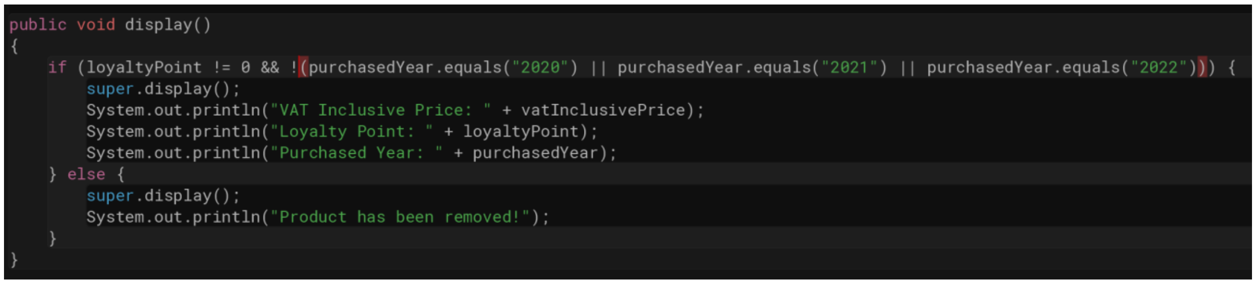
A logic error is an error in a program's source code that gives way to unanticipated and erroneous behavior. A logic error is classified as a type of runtime error that can result in a program producing an incorrect output. It can also cause the program to crash when running. (Technopedia, 2023)



```java
    //Mutator method for isPaymentOnline
    public void setIsPaymentOnline(boolean isPaymentOnline)
    {
        this.isPaymentOnline = isPaymentOnline;
    }

    public void setLoyaltyPoint(boolean isPaymentOnline, int vatInclusivePrice)
    {
        if(isPaymentOnline) {
            this.loyaltyPoint = (int) (0.01 * vatInclusivePrice);
        }
    }

    public void removeProduct()
    {
        if(loyaltyPoint == 0 && (purchasedYear.equals("2020") || purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
            this.vatInclusivePrice = 0;
            this.loyaltyPoint = 0;
            this.isPaymentOnline = false;
        }
    }

    public void display()
    {
        if (loyaltyPoint != 0 && (purchasedYear.equals("2020") || purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
            super.display();
            System.out.println("VAT Inclusive Price: " + vatInclusivePrice);
            System.out.println("Loyalty Point: " + loyaltyPoint);
            System.out.println("Purchased Year: " + purchasedYear);
        } else {
            super.display();
            System.out.println("Product has been removed!");
        }
    }
}
```

Figure 27 - Screenshot of Logical error

In the above code the logic does not make sense because it will always print the else part. The error is then corrected by putting suitable operator before the argument.



```java
public void display()
{
    if (loyaltyPoint != 0 && !(purchasedYear.equals("2020") || purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
        super.display();
        System.out.println("VAT Inclusive Price: " + vatInclusivePrice);
        System.out.println("Loyalty Point: " + loyaltyPoint);
        System.out.println("Purchased Year: " + purchasedYear);
    } else {
        super.display();
        System.out.println("Product has been removed!");
    }
}
```

Figure 28 - Screenshot of fixing Logical error

7. Conclusion

This coursework was very useful on giving us insight about the depth of Object-oriented programming. We gained a great deal of insight into the complexity of programming from the course work. We researched and learned a lot, particularly inheritance and polymorphism. We were able to learn multiple things at once from this coursework. Our knowledge of Java and program designs has improved.

Troubles faced:

Understanding criteria: At first, filtering through the criteria and breaking them down into smaller jobs was difficult.

Creating Class Connections: It was important to carefully consider the appropriate class layout between the Store, Department, and Retailer classes. It was essential to balance the use of inheritance and composition in organizing classes in a way that promotes code reusability, maintainability, and flexibility while ensuring that each class has a clear and specific responsibility within the overall system.

How the troubles were overcome:
Overcame challenges by conducting thorough analysis, making repeated revisions, and carrying out comprehensive testing.

8. References

DevX, 2023. *DevX.* [Online] *Definition of BlueJ*
Available at: https://www.devx.com/terms/bluej/
[Accessed 7 May 2024].

DevX, 2023. *DevX.* [Online] *Definition of class diagram*
Available at: https://www.devx.com/terms/class-diagram/
[Accessed 7 May 2024].

DevX, 2023. *DevX.* [Online] *Definition of pseudocode*
Available at: https://www.devx.com/terms/pseudocode/
[Accessed 8 May 2024].

DevX, 2024. *DevX.* [Online] *Definition of microsoft word*
Available at: https://www.devx.com/terms/microsoft-word/
[Accessed 7 May 2024].

Draw.io, 2023. *Draw.io.* [Online] *Definition on Draw.io*
Available at: https://www.drawio.com/about
[Accessed 7 May 2024].

Medium, 2023. *Medium.* [Online] *Definition of semantic error*
Available at: https://medium.com/thefreshwrites/java-semantic-errors-why-java-programmer-must-acknowledge-semantic-errors-6f3e63b411e6
[Accessed 7 May 2024].

Technopedia, 2023. *Technopedia.* [Online] *Definition of logical error*
Available at: https://www.techopedia.com/definition/8122/logic-error
[Accessed 8 May 2024].

Technopedio, 2023. *Technopedio.* [Online] *Definition of syntax error*
Available at: https://www.techopedia.com/definition/13391/syntax-error
[Accessed 8 May 2024].

9. Appendix

9.1 Code of Store.java

```
public class Store

{

    //Attributes of Store class

    private int storeId;

    private String storeName;

    private String location;

    private String openingHour;

    private double totalSales;

    private double totalDiscount;


    public Store(int storeId, String storeName, String location, String openingHour)

    {

        this.storeId = storeId;

        this.storeName = storeName;

        this.location = location;

        this.openingHour = openingHour;

        this.totalSales = 0.0;

        this.totalDiscount = 0.0;

    }


    //Accessor method for each attribute

    public int getStoreId()

    {

        return this.storeId;

    }
```

```
public String getStoreName()
{
    return this.storeName;
}

public String getLocation()
{
    return this.location;
}

public String getOpeningHour()
{
    return this.openingHour;
}

//Mutator method for Total sales and Total discount
public void setTotalSales(double totalSales)
{
    this.totalSales += totalSales;
}

public void setTotalDiscount(double totalDiscount)
{
    this.totalDiscount += totalDiscount;
}

//Method to display details of Store
public void display()
```

```
    {
        System.out.println("The Store ID is " + storeId + ".");
        System.out.println("The Store name is " + storeName + ".");
        System.out.println("The Store location is " + location + ".");
        System.out.println("The opening hour of " + storeName + " is " + openingHour +
".");
        System.out.println("The total sales is " + totalSales + ".");
        if(totalSales == 0 && totalDiscount == 0) {
            System.out.println("No purchase is made.");
        }
    }
}
```

9.2 Code of Retailer.java

```
public class Retailer extends Store
{
    //Attributes of Retailer class
    private int vatInclusivePrice;
    private int loyaltyPoint;
    private boolean isPaymentOnline;
    private String purchasedYear;


    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.isPaymentOnline = isPaymentOnline;
        this.purchasedYear = purchasedYear;
        this.loyaltyPoint = 0;
    }

    //Accessor method for each attribute
    public int getVatInclusivePrice()
    {
        return this.vatInclusivePrice;
    }

    public int getLoyaltyPoint()
    {
        return this.loyaltyPoint;
    }

    public boolean getIsPaymentOnline()
    {
        return this.isPaymentOnline;
    }

    public String getPurchasedYear()
    {
        return this.purchasedYear;
    }

    //Mutator method for isPaymentOnline and loyaltyPoint
    public void setIsPaymentOnline(boolean isPaymentOnline)
    {
        this.isPaymentOnline = isPaymentOnline;
    }

    public void setLoyaltyPoint(boolean isPaymentOnline, int vatInclusivePrice)
    {
        if(isPaymentOnline) {
            this.loyaltyPoint = (int) (0.01 * vatInclusivePrice);
        }
    }

    //Method to remove product
    public void removeProduct()
    {
        if(loyaltyPoint == 0 && (purchasedYear.equals("2020") ||
purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
            this.vatInclusivePrice = 0;
            this.loyaltyPoint = 0;
            this.isPaymentOnline = false;
        }
    }

    //Method to display details of Retailer and Store
    public void display()
    {
        if (loyaltyPoint != 0 && !(purchasedYear.equals("2020") ||
purchasedYear.equals("2021") || purchasedYear.equals("2022"))) {
            super.display();
            System.out.println("VAT Inclusive Price: " + vatInclusivePrice);
            System.out.println("Loyalty Point: " + loyaltyPoint);
            System.out.println("Purchased Year: " + purchasedYear);
        } else {
            super.display();
            System.out.println("Product has been removed!");
        }
    }
}
```

9.3 Code of Department.java

```
public class Department extends Store
{
    //Attributes of Department class
    private String productName;
    private double markedPrice;
    private double sellingPrice;
    private boolean isInSales;

    public Department(int storeId, String storeName, String location, String openingHour,
double totalSales, double totalDiscount, String productName, double markedPrice)
    {
        super(storeId, storeName, location, openingHour);
        super.setTotalSales(totalSales);
        super.setTotalDiscount(totalDiscount);
        this.isInSales = true;
        this.sellingPrice = 0.0;
        this.productName = productName;
        this.markedPrice = markedPrice;
    }
```

```
//Accessor method for each attribute
public String getProductName()
{
   return this.productName;
}

public double getMarkedPrice()
{
   return this.markedPrice;
}

public double getSellingPrice()
{
   return this.sellingPrice;
}

public boolean getIsInSales()
{
   return this.isInSales;
}

//Mutator method for Marked price
public void setMarkedPrice(double markedPrice)
{
   this.markedPrice = markedPrice;
}
```

```
//Method to calculate Selling price of the product
public void calculateDiscountPrice(boolean isInSales, double markedPrice)
{
    if(isInSales) {
      if(markedPrice >= 5000) {
          sellingPrice = markedPrice - (markedPrice * 0.2);
      }
      else if(markedPrice >= 3000) {
          sellingPrice = markedPrice - (markedPrice * 0.1);
      }
      else if(markedPrice >= 1000) {
          sellingPrice = markedPrice - (markedPrice * 0.05);
      }
      else {
          sellingPrice = markedPrice;
      }
    }
    setTotalDiscount(markedPrice-sellingPrice);
    setTotalSales(sellingPrice);
    this.isInSales = false;
}

//Method to display details of Department and Store
public void display()
{
    super.display();
    if(isInSales) {
        System.out.println("The product name is " + productName);
```

```
            System.out.println("The marked price of the product is " + markedPrice);
        }
        else {
            System.out.println("The product name is: " + productName);
            System.out.println("The selling price of the product is " + sellingPrice);
        }
    }
}
```

THE END