LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4001NI Programming

Assessment Weightage & Type

30% Individual Coursework 2

Year and Semester

2021- 22 Spring - 1

Student Name: Anukul Karki

London Met ID: 21049482

College ID: NP01CP4A210134

Assignment Due Date: 5th August 2022

Assignment Submission Date: 5th August 2022

I confirm that I understand my coursework needs to be submitted online via Google Classroom 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.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 About Coursework

In the given coursework project to develop software for the musical store named as Sarangi Sansar was assigned which was to be done in Java programming language using the IDE BlueJ as instructed. In this coursework projects were assigned in such a way that it gives the knowledge about the understanding of the programming language. The main objective of the given coursework is to write a code to make GUI for the musical store. As the musical store Sarangi Sansar provides different services that allows customers to buy different instruments, renting the instruments. As, doing all these work manually might be hectic. So, the main aim for this code is to provide the software that can handle all the functions like storing the data of those people who have rented the instrument and buyer of the instrument that makes daily operations easy for the Sarangi Sansar.

Here, while developing code four different classes were made where one was parent class and others two were child class and the final class was to make the GUI. Parent class name is Instrument whereas child class names are InstrumentToSell and InstrumentToRent, the class name for the GUI section was SarangiSansar. In the class InstrumentToSell helps to handle the data about selling the instrument in the store and InstrumentToRent helps handling the data about the renting the musical instruments. The class SarangiSansar handles most of the part, as it contains the GUI section. Also, the class access all classes that are Instrument, InstrumentToRent and InstrumentToSell. This class helps in storing instruments that is to be rented or sell. It helps checking if the instrument is available or not, if all the conditions satisfies it allows method from other classes to be run. The code contains different swing components, awt components in order to complete the GUI section. Different methods for handling the events were also created. For the Other classes, they contain many setters, getters method and many more. In the overall code different static, local, instance variable was also used, different access modifier was also used in the process.

Not only, the codes but also the report in the coursework is also mandatory as it contains the Class diagram the made from the code, the Pseudocode. It also contains the testing of the code by inserting different values in the GUI section in order to check if it is fully working or not. Report also contains the error that was created in the code while developing it.

1.2 Tools Used

While completing the coursework different tools were used to make the report and project complete and they are as follow

- BlueJ
- Ms-Word
- Moqups
- Draw.io

a) BlueJ

BlueJ is the IDE for the Java programming language which provides the platform for java to be run. The main function of the BlueJ is that the codes can still be compiled even without using main method. As it has different functions so the codes were to be written in the BlueJ as it was the criteria for the coursework too. It best suited for the coursework as it is free for all users. BlueJ is user friendly too, so anyone can use this IDE which make it easy to complete the project. (Techopedia, 2020)

(Png, 2021)

Figure 1 BlueJ

b) Ms-word

Ms-word is a product of Microsoft office package that helps in creating different projects. It is best suited for the coursework as it helps in making the report for the code that was made for the SarangiSansar. As the report contains different Class Diagram, Pseudocode, Testing and it contains different images as well as tables which can be easily written in the MS-word. The functionality that helps to auto generate the table of figures, table of content, table of table provided lots of help and the function that helps to insert citation was also very useful while creating the report.

(Png, 2021)

Figure 2 MS - Word

c) Moqups

Moqups is a web app that help in making wire frame for the GUI section of the musical store named SarangiSansar. The main objective of making the wireframe was to get the overall view of how will the GUI section will be displayed. Also using the moqups also helped in getting the coordinates that are used in the setBounds function of the code. It also helped in placing the right color to make GUI attractive. It helped in placement of the Labels, TextFields, Buttons in order to make the GUI easy to use to all the users.


(crunchbase, 2015)

Figure 3 Moqups

d) Draw.io

Draw.io is a web app that is used to create class diagram of the classes Instrument, InstrumentToSell, InstrumentToRent, SarangiSansar. It contains the different variables and methods that are used in the code. It is user friendly and can be easily used to make class diagrams. So, it is also used to while completing the report.

 (crunchbase, 2015)

Figure 4 Draw.io

2. Class Diagram

Class Diagram is a type of diagram that defines and provides the structure of a code that contains the variable and its datatype as well as the method and its return type. The figures below represent the class diagram of the Instrument class, InstrumentToRent and InstrumentToSell class and SarangiSansar

a) Class Diagram of Instrument Class.

Here is the class diagram of the Instrument class that contains all the methods, instance variable, return type, access modifier.

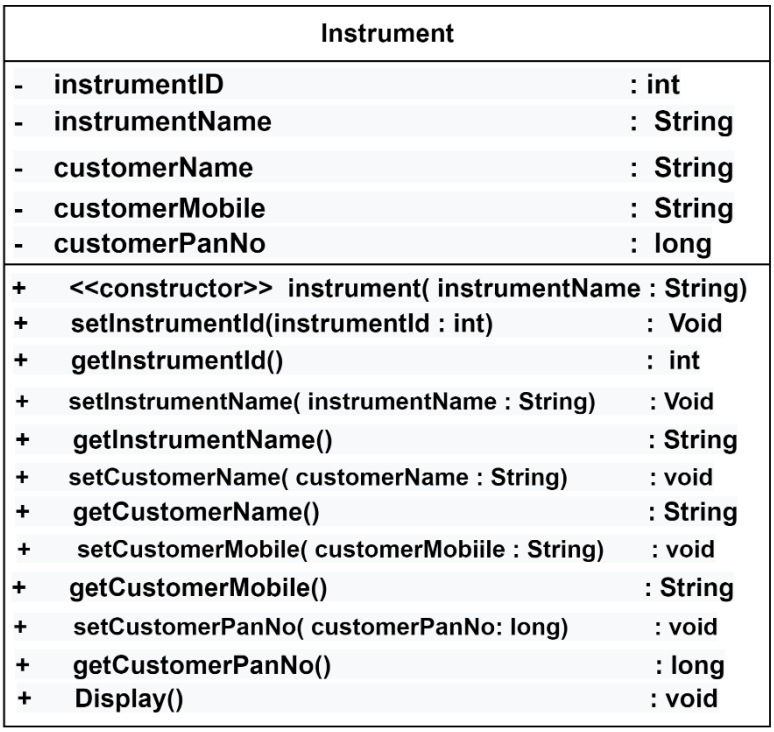

Figure 5 Class Diagram Of Instrument

b) Class Diagram of InstrumentToRent class

Here is the class diagram of the InstrumentToRent class that contains all the methods, instance variable, return type, access modifier.

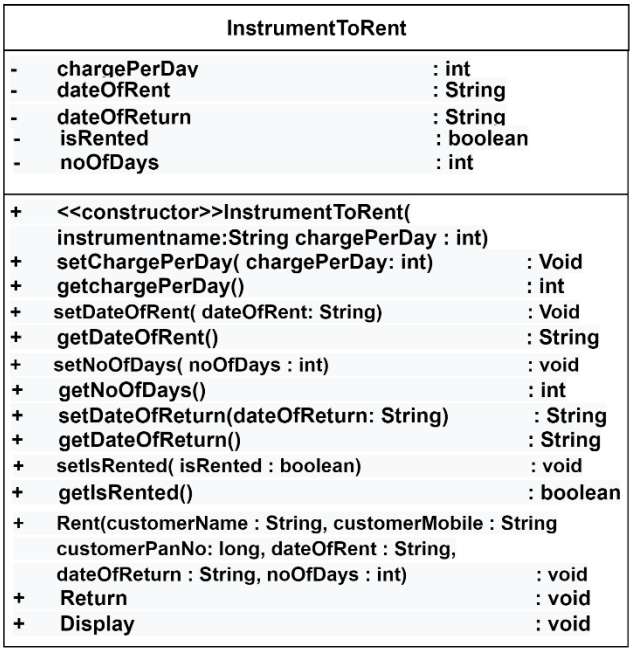

Figure 6 Class Diagram of InstrumentToRent

c) Class Diagram of InstrumentToSell class

Here is the class diagram of the instrumentToSell class that contains all the methods, instance variable, return type, access modifier.

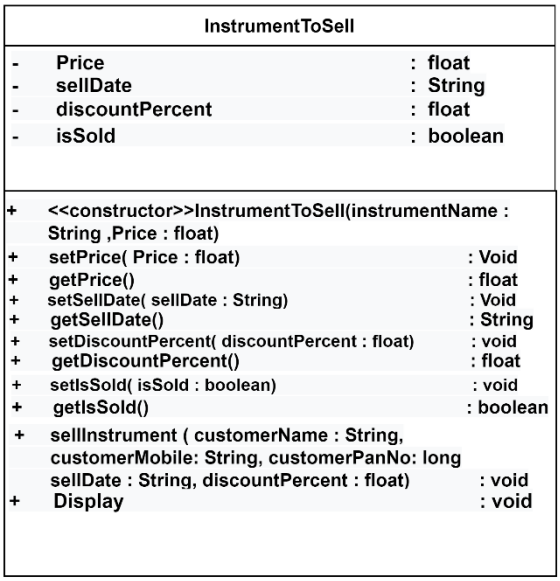

Figure 7 Class Diagram of InsrumentToSell

d) Class Digram of SarangiSansar class

Here is the class diagram of the SarangiSansar class that contains all the methods, instance variable, return type, access modifier.

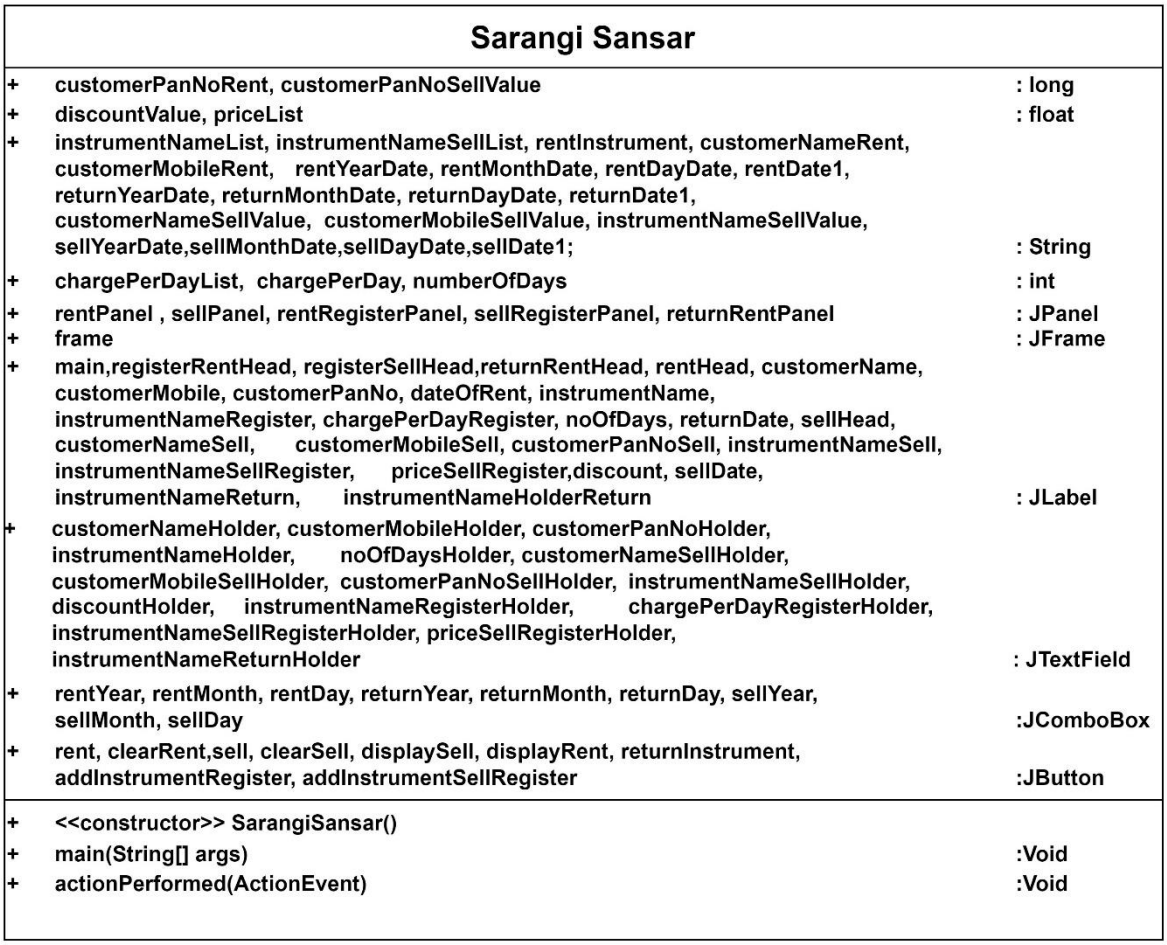

Figure 8 Class Diagram of Sarangi Sansar

e) Connected Class Diagram

Here in the figure it has shown how those class diagram was connected to each other.

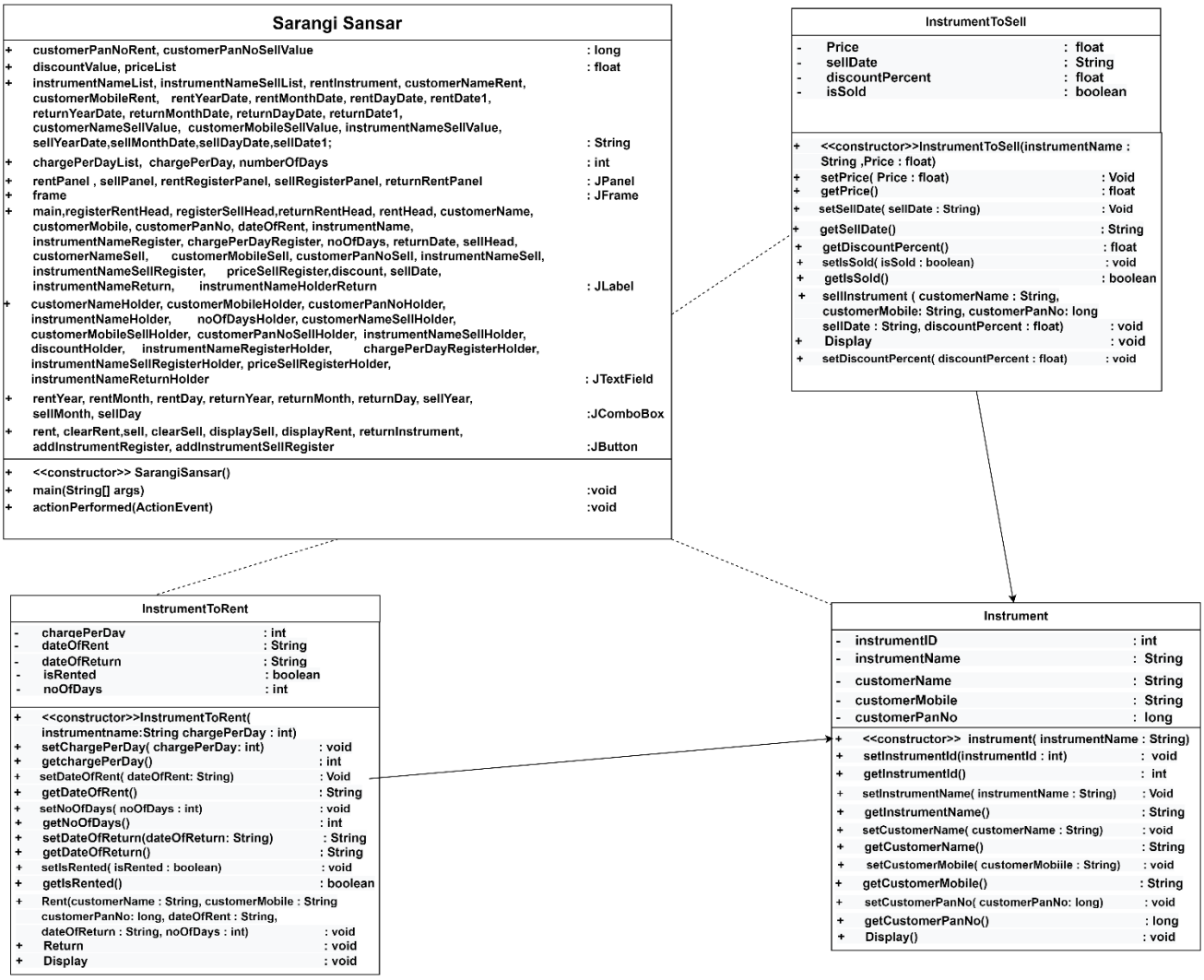

Figure 9 Class Diagram of All Class

3. Pseudocode

Pseudocode is a way of writing code in the simple language as possible which uses short phrases to write codes. It is written before actually writing in a specific language. It is used to help the developer for making the algorithm.

3.1 Pseudocode of Sarangi Sansar

Pseudocode of the class SarangiSansar is given below

START

IMPORT swing component from javax
IMPORT awt components from java
IMPORT event from java awt
IMPORT ArrayList from java util package

CREATE class SarangiSansar **IMPLEMENTS** ActionListnener
DO
DECLARE an instance variable of long data type
DECLARE an instance variable of float data type
DECLARE an instance variable of String data type
DECLARE an instance variable of int data type
DECLARE JPanel
DECLARE JFrame
DECLARE JLabel
DECLARE JTextField
DECLARE JComboBox
DECLARE JButton
DECLARE ArrayList
END DO

CREATE a constructor named as SarangiSansar
DO

CREATE new JFrame named as SarangiSansar

CREATE multiple JPanels for adding all the respective components in it
CREATE multiple JLabels to display all the respective text in the Frame

CREATE different arrays to store multiple data at ones

CREATE multiple JComboBox to add dates for renting and selling purposes
CREATE multiple Jbuttons to do all the respective function assigned
SET addActionListener method to all JButtons for event handiling

CREATE multiple fonts in the variable to change the respective designs
SET all the components to its preferred size and positions
ASSIGN multiple fonts to the preferred components of the frame
ASSIGN multiple backgrounds to the preferred components
ADD multiple components to the preferred JPanels
SET multiple borders to the preferred JPanels
SET the layout of the multiple JPanels null
SET the layout of the JFrame to null
ADD multiple components to the preferred JFrame
SET the visible of layout of JFrame to true
SET the visible of layout of JFrame to true
SET the size of the JFrame frame

END DO
CREATE an instance method actionPerformed having ActionEvent as parameter
DO

IF event source is equal to addInstrumentRegister **THEN**
DO
SET the value of exist to false
IF instrumentName is empty **OR** chargePerDay is empty **THEN**
DO
GENERATE an error message for empty field
END IF
ELSE
DO
TRY
DO
GET the value of JTextField
CONVERT the value of string to int from JTextField

IF list size is equal to zero **THEN**
DO
CREATE object of InstrumetnToRent
ADD object into ArrayList
GENERATE an information message for instrument added
END IF
ELSE
DO
CREATE loop for arraylist
DO
IF instrumentname is equal and object instance of Instrument To Rent **THEN**
DO
SET value of exist to true
BREAK the loop

END IF

END DO

IF exist is equals to false **THEN**

DO

CREATE object of InstrumetnToRent

ADD object into ArrayList

GENERATE an information message for instrument added

ENDIF

ELSE

DO

GENERATE an error message as already exist

END DO

END DO

SET the value of two JtextFields to empty

CATCH number format exception

DO

GENERATE the error message for wrong data type

SET the value of two JtextFields to empty

END DO

END DO

END IF

IF event source is equal to addInstrumentSellRegister **THEN**

DO

SET the value of sellexist to false

IF instrumentName is empty **OR** price is empty **THEN**

DO

GENERATE an error message for empty field

END IF

ELSE

DO

TRY

DO

GET the value of JTextField

CONVERT the value of string to int from JTextField

IF list size is equal to zero **THEN**

DO

CREATE object of InstrumetnToSell

ADD object into ArrayList

GENERATE an information message for instrument added

END IF

ELSE

DO

CREATE loop for arraylist

DO

IF instrumentname matched and object instance of Instrument To Sell

DO

SET value of sellExist to true

BREAK the loop

END IF

END DO

IF sellExist is equals to false **THEN**

DO

CREATE object of InstrumetnToSell

ADD object into ArrayList

GENERATE an information message for instrument added

ENDIF

ELSE

DO

GENERATE an error message as already exist

END DO

END DO

SET the value of two JtextFields to empty

CATCH number format exception

DO

GENERATE the error message for wrong data type

SET the value of two JtextFields to empty

END DO

END DO

END IF

IF event source is equal to rent **THEN**

DO

IF JtextFields are empty for rent all **THEN**

DO

GENERATE an error message for empty

END IF

ELSE

DO

TRY

DO

GET the value of instrument name

SET the value of string to Integer

GET the value of customer name

GET the value of mobile

GET the value of customer pan no

GET the value of rent year

GET the value of rent month

GET the value of rent day

ADD the three value of rent date
GET the value of return year
GET the value of return month
GET the value of return day
ADD the three value of return date
SET the value of rentNumber to 0
CREATE loop for arraylist
DO
IF instrument name matches **THEN**
DO
IF instrument instance of InstrumentToRent **THEN**
DO
CREATE an object by doing downcasting
IF instrument is rented **THEN**
DO
CREATE loop for arraylist
DO
If instrument name match AND instrument rent is rented and instrument instance instrument to rent **THEN**
DO
GENERATE the appropriate message
BREAK the loop
END IF
END DO
BREAK the loop
END IF
ELSE
DO

GENERATE the appropriate information message
SET all the text fields of rent to empty
BREAK the loop
END DO
END IF
END IF
ADD the value of rent number by 1

END DO
IF rent number equal size of list
DO
GENERATE the appropriate message
END IF
END DO
CATCH
DO
GENERATE the appropriate message
END DO
END IF
IF event source is equal to sell **THEN**
DO
IF JtextFields are empty for rent all **THEN**
DO
GENERATE an error message for empty
END IF
ELSE
DO
TRY
DO
GET the value of customer name
GET the value of customer mobile
GET the value of instrumentName
SET the value of string to Float of discount

SET the value of string to long of customer pan no
GET the value of sell year
GET the value of sell month
GET the value of sell day
ADD the three value of sell date
SET the value of rentNumber to 0
CREATE loop for arraylist
DO
IF instrument name matches **THEN**
DO
IF instrument instance of InstrumentToSell **THEN**
DO
CREATE an object by doing downcasting
IF instrument is sold **THEN**
DO
CREATE loop for arraylist
DO
If instrument name match AND instrument instance instrument to sell **THEN**
DO
GENERATE the appropriate message
BREAK the loop
END IF
END DO
BREAK the loop
END IF
ELSE
DO
GENERATE the appropriate information message

SET all the text fields of rent to empty

BREAK the loop

END DO

END IF

END IF

ADD the value of sellNumber by 1

END DO

IF rent number equal size of list

DO

GENERATE the appropriate message

END IF

END DO

CATCH

DO

GENERATE the appropriate message

END DO

END IF

IF event source is equal to clearRent **THEN**

DO

SET the value of instrumentNameRegisterHolder to empty

SET the value of chargePerDayRegisterHolder to empty

SET the value of customerNameHolder to empty

SET the value of customerMobileHolder to empty

SET the value of customerPanNoHolder to empty

SET the value of instrumentNameHolder to empty

SET the value of noOfDaysHolder to empty

SET the value of instrumentNameReturnHolder to empty

SET the index of rentYear to 0

SET the index of rentMonth to 0

SET the index of rentDay to 0

SET the index of returnYear to 0

SET the index of returnMonth to 0

SET the index of returnDay to 0

END IF

IF event source is equal to clearRent **THEN**

DO

SET the value of instrumentNameSellRegisterHolder to empty

SET the value of priceSellRegisterHolderto empty

SET the value of customerNameSellHolderto empty

SET the value of customerMobileSellHolderto empty

SET the value of customerPanNoSellHolderto empty

SET the value of instrumentNameSellHolderto empty

SET the value of discountHolderto empty

SET the index of sellYearto 0

SET the index of sellMonthto 0

SET the index of sellDayto 0

END IF

IF event source is equal to returnInstrument **THEN**

DO

IF instrumentName is empty **THEN**

DO

GENERATE the appropriate message for empty field

END IF

ELSE

DO

GET the value of instrument name

SET the value of itemCheck to false

CREATE the loop for array list

DO

IF instrument name matches **THEN**

DO

IF instrument instance of Instrument To Rent

DO

CREATE an object by doing downcasting
SET the value of itemCheck to true
IF instrument is rented **THEN**
DO
GENERATE the approprate message
CALL the return method
END IF
ELSE
DO
GENERATE the appropriate message
END DO
BREAK the loop
END IF
END IF
END DO
IF itemCheck is equal to false **THEN**
DO
GENERATE the appropriate message
END IF
END DO
END IF
IF event source is equal to displayRent **THEN**
DO
SET the value of counter to 0
SET the value of rentCounter to 0
IF instrument instance of instrumentToRent **THEN**
DO
CREATE an object by doing downcasting
CALL the display method
ADD the value of counter by 1

IF instrument is not rented **THEN**
DO
ADD the value of counter by 1
END IF
ELSE
DO
GENERATE the appropriate message
END DO
END IF
END DO
IF counter equals to rentCounter
DO
GENERATE the appropriate message
END DO
END IF
IF event source is equal to displaySell **THEN**
DO
SET the value of sellCounter to 0
SET the value of sellNumber to 0
IF instrument instance of instrumentToSell **THEN**
DO
CREATE an object by doing downcasting
CALL the display method
ADD the value of sellNumber by 1
IF instrument is not rented **THEN**
DO
ADD the value of sellCustomer by 1
END IF
ELSE
DO
GENERATE the appropriate message
END DO
END IF
END DO

IF sellCustomer equals to sellNumber

DO

GENERATE the appropriate message

END DO

END IF

END DO

CREATE a main method of void return type

DO

CALL the constructor

END DO

END DO

4. Method Description

Method Description contains all the methods that are present in the class SarangiSansar. It explains how each of the function works, but in this course work there are only few methods so according to the course work criteria description of button will be added. In the description what and how the button work will be explained, as there is only one class so the buttons are explained under one sub heading and the methods are explained under another method

4.1 Method Description: Sarangi Sansar

The methods that are present in this code are

a) public void actionPerfored(ActionEvent event)

It is the instance method whose return type is void. It also accepts the parameter of the events that is fired while pressing the button. All of the event handling that was completed in the course work was done in this method. Different methods like getSource() inside this method helped in completing this event handling

b) public static void main(String[] args)

It is the main method whose return type is void. This method was used in the coursework in order to call the constructor where the GUI section was done.

4.2 Button Description: SarangiSansar

Here, in the button description, the functionality of buttons like what does it do and what functions are added in the buttons are explained

Button	Button Description
addInstrumentRegister	Here, In, this button the main functionality was to add the instrument that can be rented in the near future in the arraylist. This button takes two values instrument name and charge per day in order to make an object of instrumentToRent class, which was then added to the arraylist. It also checks if the instrument is already in the list or not. If the instrument does not exist in the arraylist then the object is created and then it is added. And if the instrument does exist it display the appropriate error message to make user know that the instrument is already added. It also checks if the data field is empty or not. If the data is empty, then it will let the user know about the problem. It also checks if the charge per day is not written in the number format, if it isn't in the number format then the appropriate error message is shown to the user. At last after completing the process the text field will automatically get empty making place to add the new instrument.

addInstrumentSellRegister	Here, In, this button the main functionality was to add the instrument that can be sell in the near future in the arraylist. This button takes two values instrument name and price in order to make an object of instrumentToSell class, which was then added to the arraylist. It also checks if the instrument is already in the list or not. If the instrument does not exist in the arraylist then the object is created and then it is added. And if the instrument does exist it display the appropriate error message to make user know that the instrument is already added. It also checks if the data field is empty or not. If the data is empty, then it will let the user know about the problem. It also checks if the price is not written in the number format, if it isn't in the number format then the appropriate error message is shown to the user. At last after completing the process the text field will automatically get empty making place to add the new instrument
rent	Here, this button is one of the most important button in the course work. The main functionality of this button is to rent the instrument to the customer. Firstly, it checks if the field required to rent the instrument is empty or not, if it is empty then It will let the user know that the field is empty. Also, it will check if the customer pan no and no of days is in the right format i.e number format. If it is not in the right format, then the error message is displayed to the user. Now if all the text field is filled with the correct data then the instrument that is going to be rent is checked in the array list. If the

	instrument is not available in the array list then the instrument cannot be rented and dialog box will appear, but if the instrument is available in the array list then the rent method of the InstrumentToRent is called with the appropriate parameter and the instrument is rented. If the user tries to rent the same instrument again then the appropriate message letting the user know that the instrument is already rented with the respective username will be displayed. After completing all these tasks all the field that are used to rent the instrument will be clear for another instrument that is needed to be rent.
sell	Here, this button is one of the most important button in the course work. The main functionality of this button is to sell the instrument to the customer. Firstly, it checks if the field required to sell the instrument is empty or not, if it is empty then It will let the user know that the field is empty. Also, it will check if the customer pan no and discount is in the right format i.e number format. If it is not in the right format, then the error message is displayed to the user. Now if all the text field is filled with the correct data then the instrument that is going to be sell is checked in the array list. If the instrument is not available in the array list then the instrument cannot be sell and dialog box will appear, but if the instrument is available in the array list then the sell method of the InstrumentToSell is called with the appropriate parameter and the instrument is sell. If the user tries to sell the same instrument again then the appropriate message letting the user know that the

	instrument is already sold with the respective user name will be displayed. After completing all these tasks all the field that are used to sell the instrument will be clear for another instrument that is needed to be sell
returnInstrument	Here, the main objective of this button is to return the instrument that are rented by the customer. This button checks if the text field required by this button is empty or not. If it is empty, then the appropriate message will be shown. Now when the button is pressed it check that if the instrument is in the arraylist or not. If instrument not in the list, then the error message will appear with a message saying that the instrument is not found. If the instrument is found, then then it checks if the instrument is rented or not. If the instrument is not rented, then the message will come for the user to let them know that the item is not rented. But if the instrument is rented then the return method from the instrumentToRent class will be called and a message saying that the instrument will be returned with the total price will be displayed.
clearRent	Here, the main objective of this button is to clear all the section that are needed for the renting purposes. It resets all the combo box values; it clears all the text field values.
clearSell	Here, the main objective of this button is to clear all the section that are needed for the selling purposes. It resets all the combo box values; it clears all the text field values.

displaySell	Here, the main objective of this button is to display the data of all the users that have buy the instrument from the store. It checks if any of the item is sold to any customer, if the item is not sold to the customer then it will show the message letting the user know that no items is sold. But if the instrument is already sold then the data of the customer will be shown to the user
displayRent	Here, the main objective of this button is to display the data of all the users that have rented the instrument from the store. It checks if any of the item is rented to any customer, if the item is not rented to the customer then it will show the message letting the user know that no items is rented. But if the instrument is already rented then the data of the customer will be shown to the user

Table 1 Button Description

5. TEST

Different Tests were done in the GUI section to check if the code works fluently on different scenario like adding the instrument, renting the instrument, selling the instrument and returning the instrument.

5.1 Test 1: Command Prompt

The program for musical store is compiled and run using the command prompt.

Test No:	1
Objective:	Program was compiled and run using the command prompt
Action:	• Here, the command prompt was initialized in the project folder and the program was compiled using the code: javac SarangiSansar.java • The code is then run using the code: java SarangiSansar
Expected Result:	The GUI should be initialized
Actual Result:	The GUI was initialized
Conclusion:	The Test is successful.

Here, is the image for the evidence to show that the java was compiled using the command prompt.

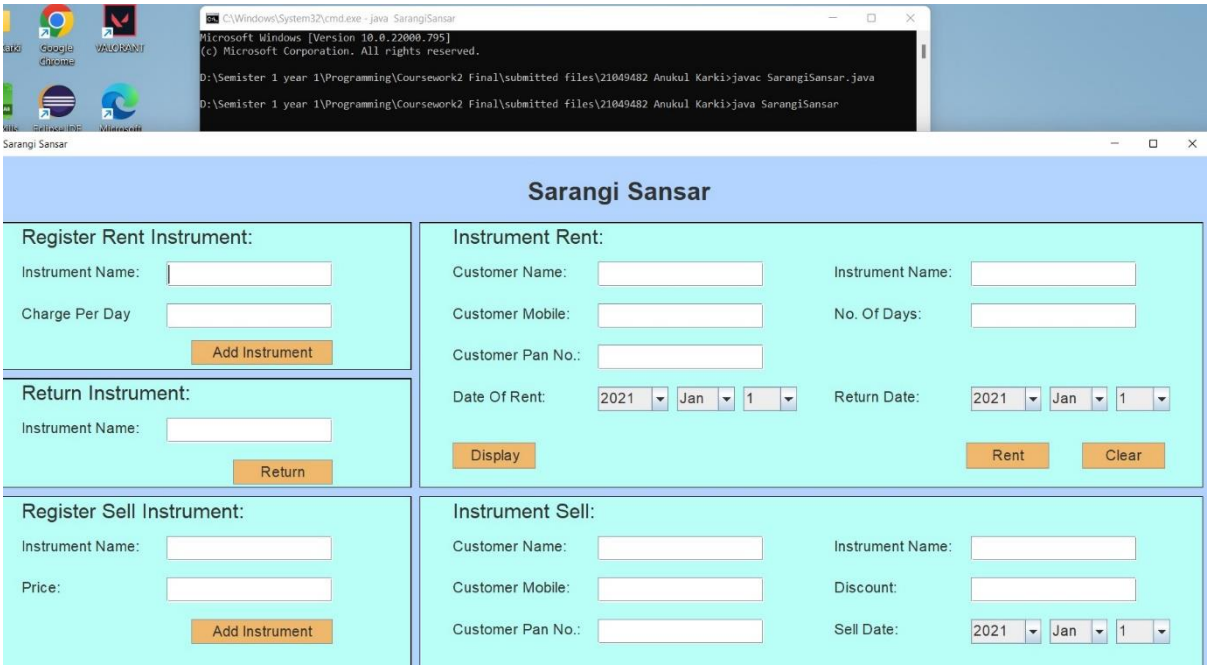

Figure 10 Compiling Code Using CMD

5.2 Test 2

Here, in the test 2 different functions like adding, renting, selling returning the instrument was done with the proper evidence of the screenshots.

a) Add Instrument for rent

Here, the instrument in the arraylist for the renting purposes.

Test No:	2.1
Objective:	Add the instrument for rent by passing the value in the GUI
Action:	• Here, the value of instrument name and the charge per day was added in the text fields of addInstrumentRentPanel are . InstrumentName = “drum” chargePerDay = 200 • The Button Add Instrument was pressed
Expected Result:	A message should be displayed with the information of instrument name and charge per day
Actual Result:	A message was displayed with the information of instrument name and charge per day
Conclusion:	The Test is successful.

Table 2 Test to add instrument for rent

Here, is the image for the evidence to show that instrument was added for the rent.

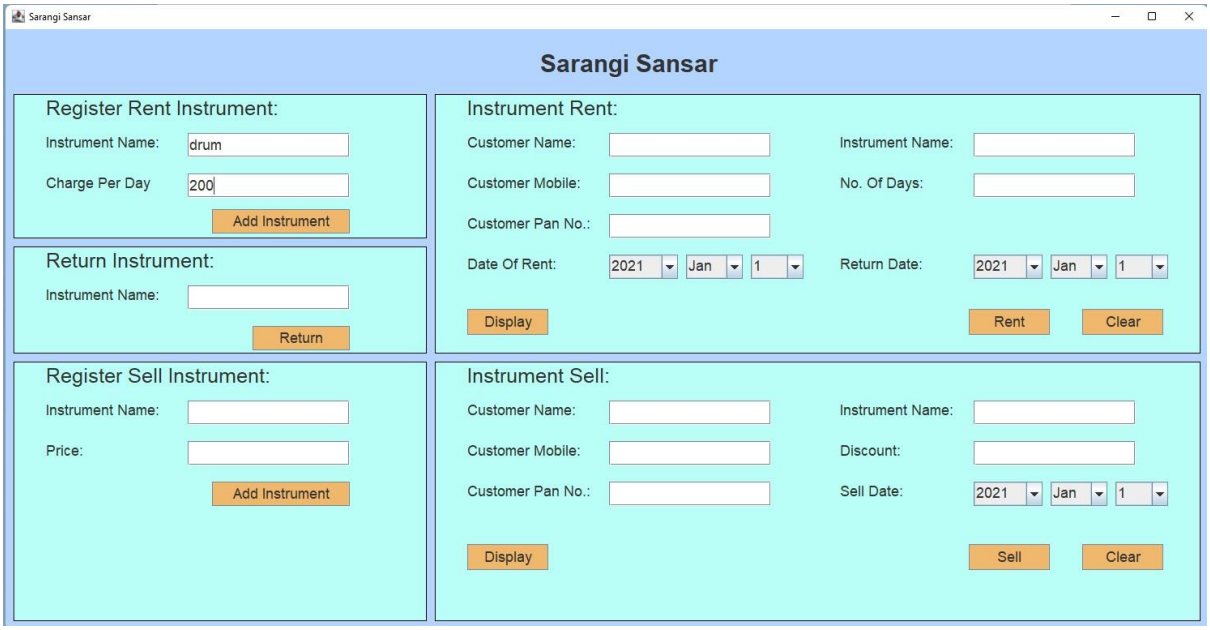

Figure 11 Before Adding for rent

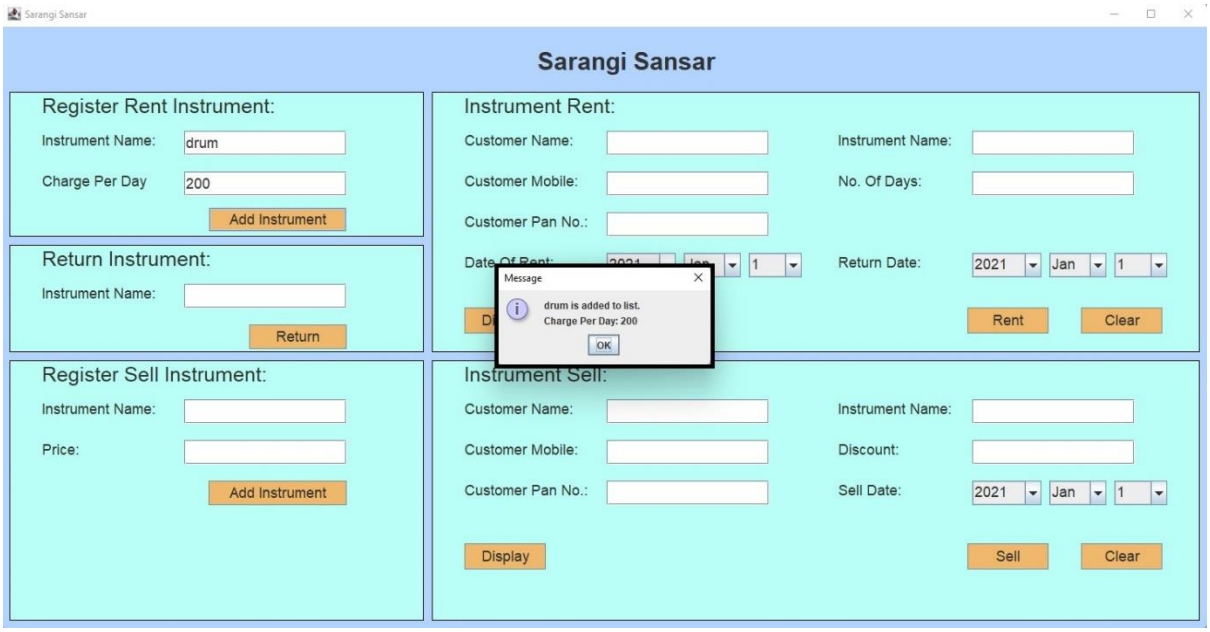

Figure 12 After adding instrument to rent

b) Add Instrument to sell

Here, the instrument in the arraylist for the selling purposes.

Test No:	2.2
Objective:	Add the instrument for sell by passing the value in the GUI
Action:	• Here, the value of instrument name and the price was added in the text fields of addInstrumentSellPanel are. InstrumentName = “guitar” price = 200 • The Button Add Instrument was pressed
Expected Result:	A message should be displayed with the information of instrument name and price
Actual Result:	A message was displayed with the information of instrument name and price
Conclusion:	The Test is successful.

Table 3 Test to add instrument for sell

Here, is the image for the evidence to show that instrument was added for the sell.

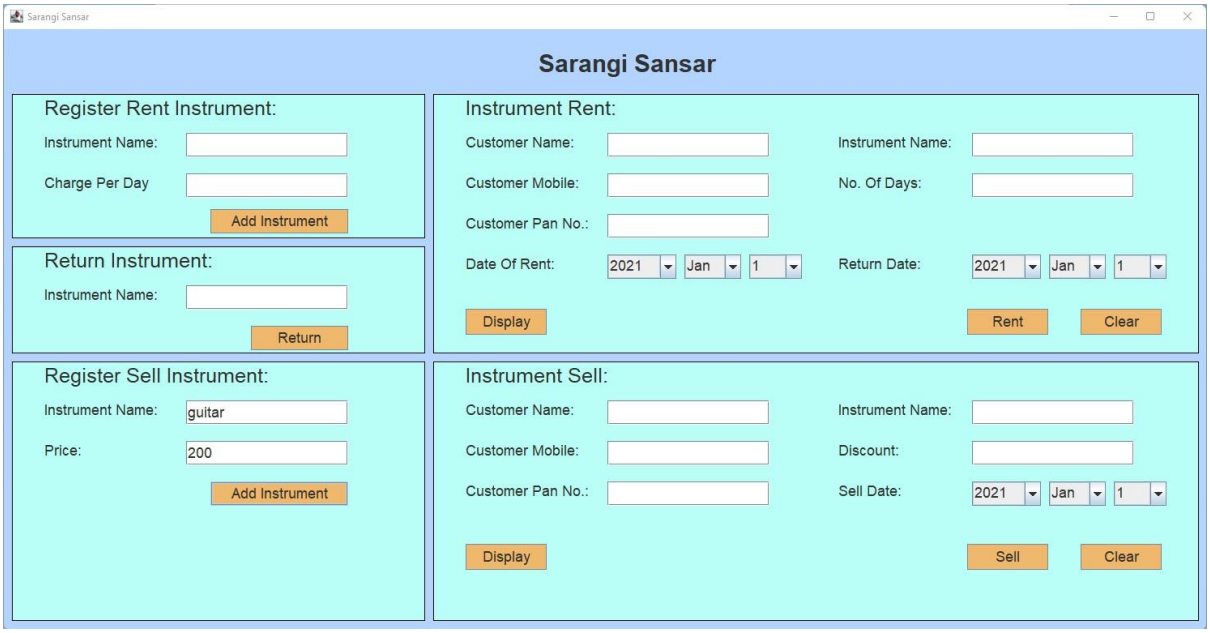

Figure 13 Before adding for sell

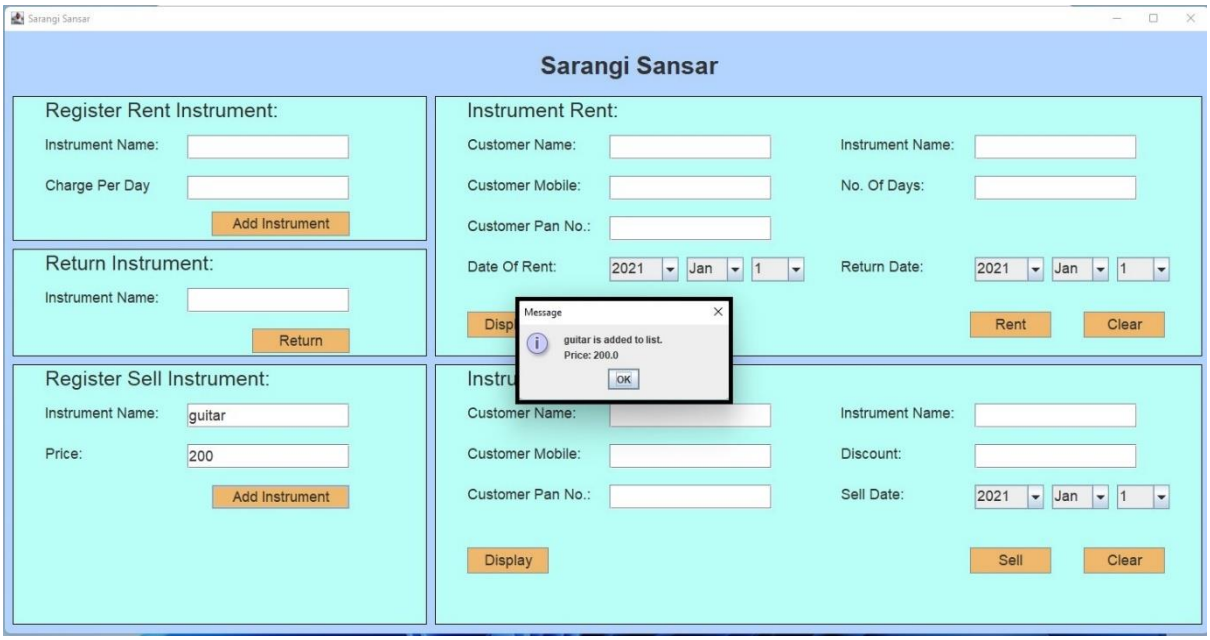

Figure 14 After Adding for sell

c) Rent the instrument

Here, the instrument is rented to the customer with the proper parameter.

Test No:	2.3
Objective:	The instrument is rented to the customer with the proper parameter
Action:	• Here, the given value of instrument name, customer name, customer mobile, number of days, customer pan no, date of rent and date of return in the respective GUI panel are: InstrumentName = “drum” Customer Name = “Anukul Karki” Customer Mobile = “9845634310” Customer Pan No = 2341243 No of days = 5 Date of rent = “2022 jan 1” Date of return = “2022 jan 5” • Then the rent button is pressed.
Expected Result:	A message should be displayed with the information of instrument name and charge per day saying the instrument is rented.
Actual Result:	A message was displayed with the information of instrument name and charge per day saying the instrument is rented
Conclusion:	The Test is successful.

Table 4 Test to renting the instrument

Here, is the image for the evidence to show that instrument was rented to the customer

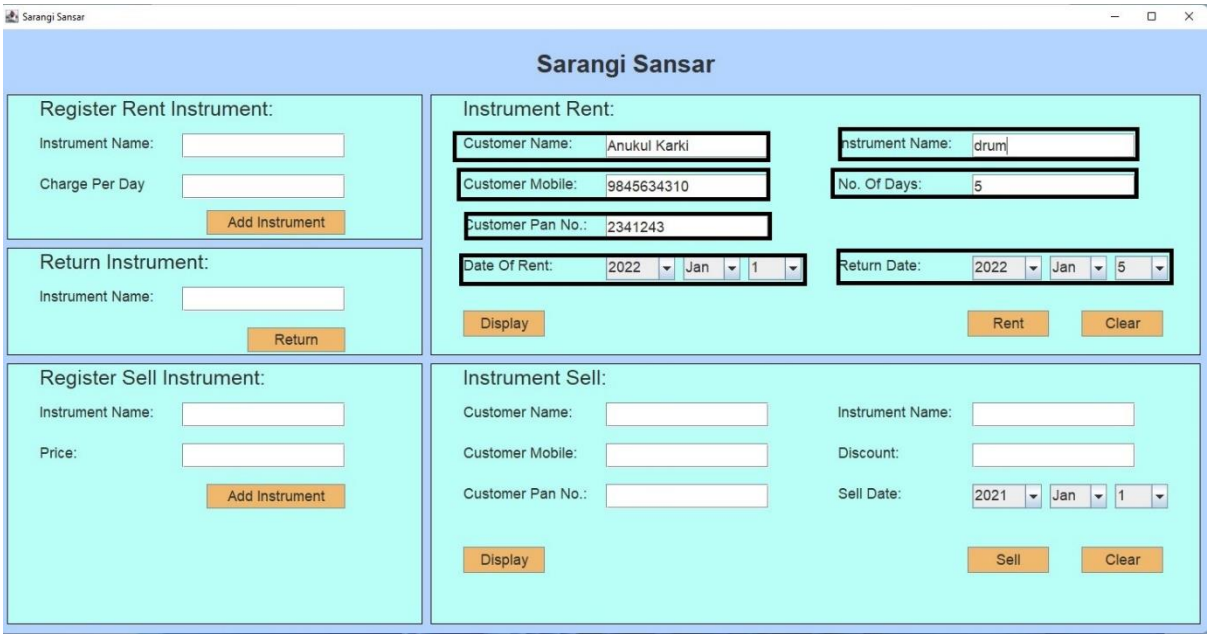

Figure 15 Before renting

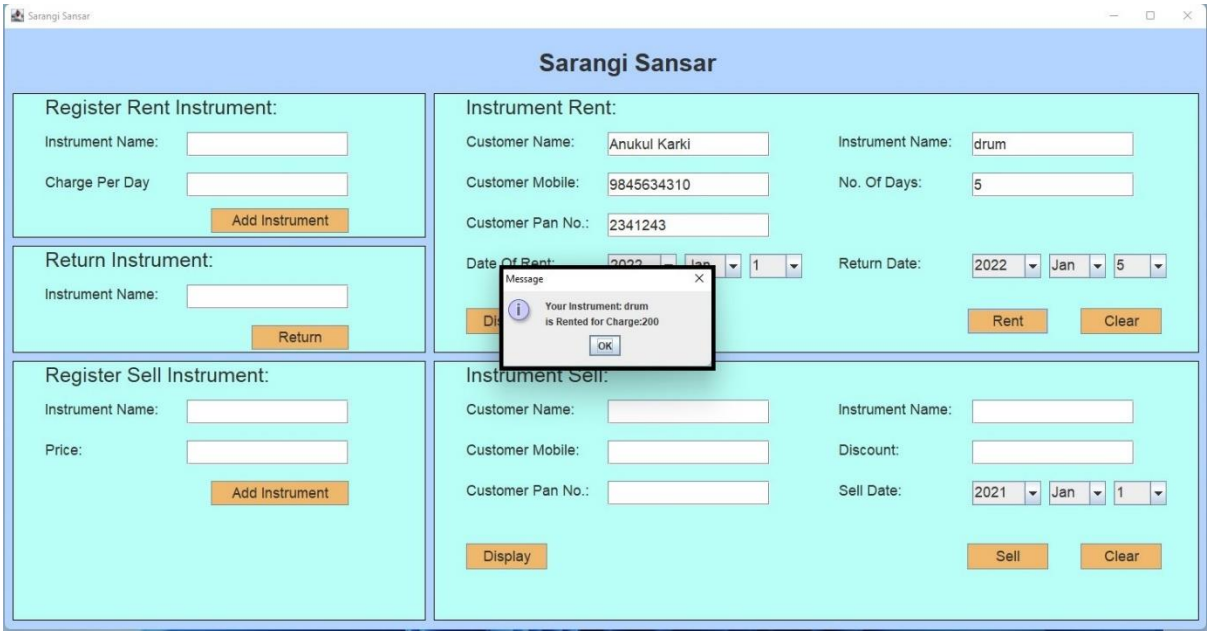

Figure 16 After Renting

d) Sell the instrument

Here, the instrument is sell to the customer with the proper parameter.

Test No:	2.4
Objective:	The instrument is sold to the customer with the proper parameter
Action:	• Here, the given value of instrument name, customer name, customer mobile, discount, customer pan no, sell date in the respective GUI panel are: Instrument Name = “guitar” Customer Name = “Anmol Karki” Customer Mobile = “9785423145” Customer Pan No = 123412 Discount = 5 Sell date = “2021 jan 1” • Then the Sell button is pressed.
Expected Result:	A message should be displayed with the information of instrument name and final price after discount saying the instrument is sold.
Actual Result:	A message was displayed with the information of instrument name and final price after discount saying the instrument is sold.
Conclusion:	The Test is successful.

Table 5 Test To Sell the instrument

Here, is the image for the evidence to show that instrument was sold to the customer

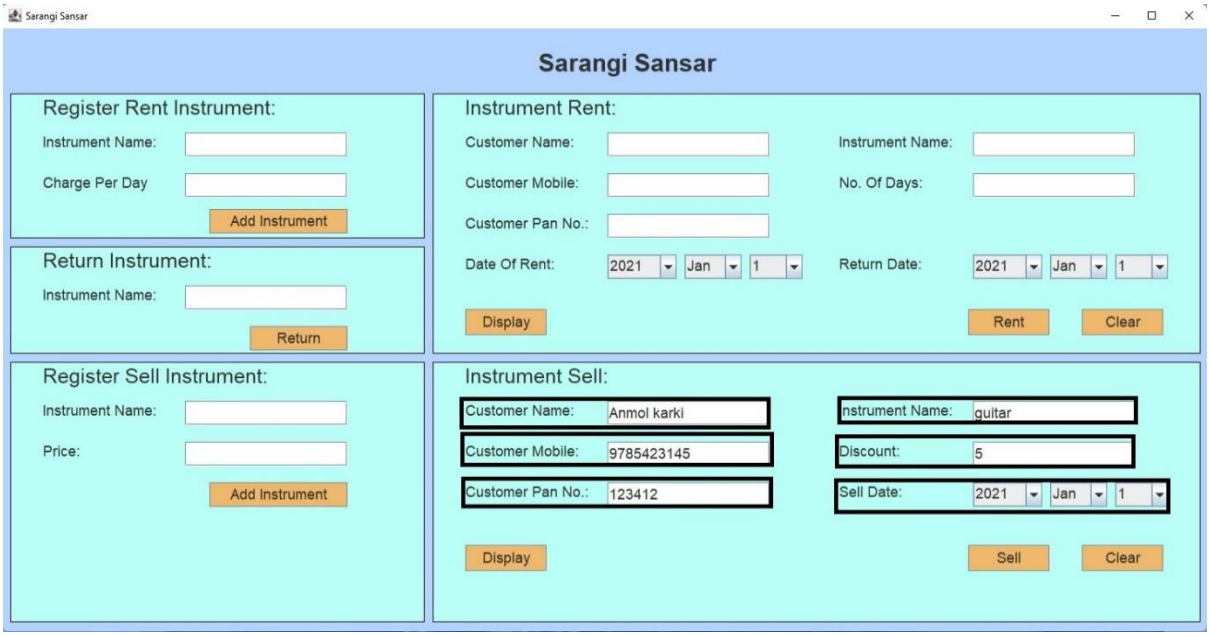

Figure 17 Before Selling

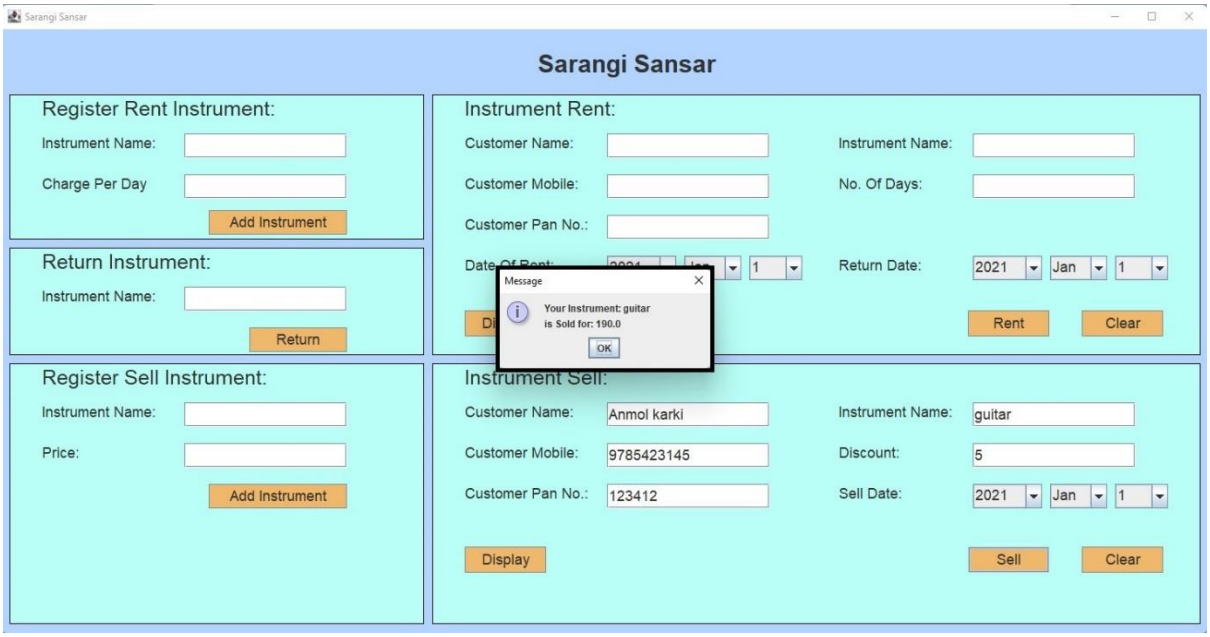

Figure 18 After Selling

e) Return the instrument

Here, the instrument is returned back by passing the instrument name

Test No:	2.5
Objective:	The instrument was return by passing the instrument name as value
Action:	• Here, the value of instrument name was added in the respective GUI panel are: Instrument Name = “drum • Then the Return button is pressed.
Expected Result:	A message should be displayed with the information of instrument name and total price of instrument.
Actual Result:	A message should be displayed with the information of instrument name and total price of instrument.
Conclusion:	The Test is successful.

Table 6 Test to return instrument

Here, is the image for the evidence to show that instrument was sold to the customer

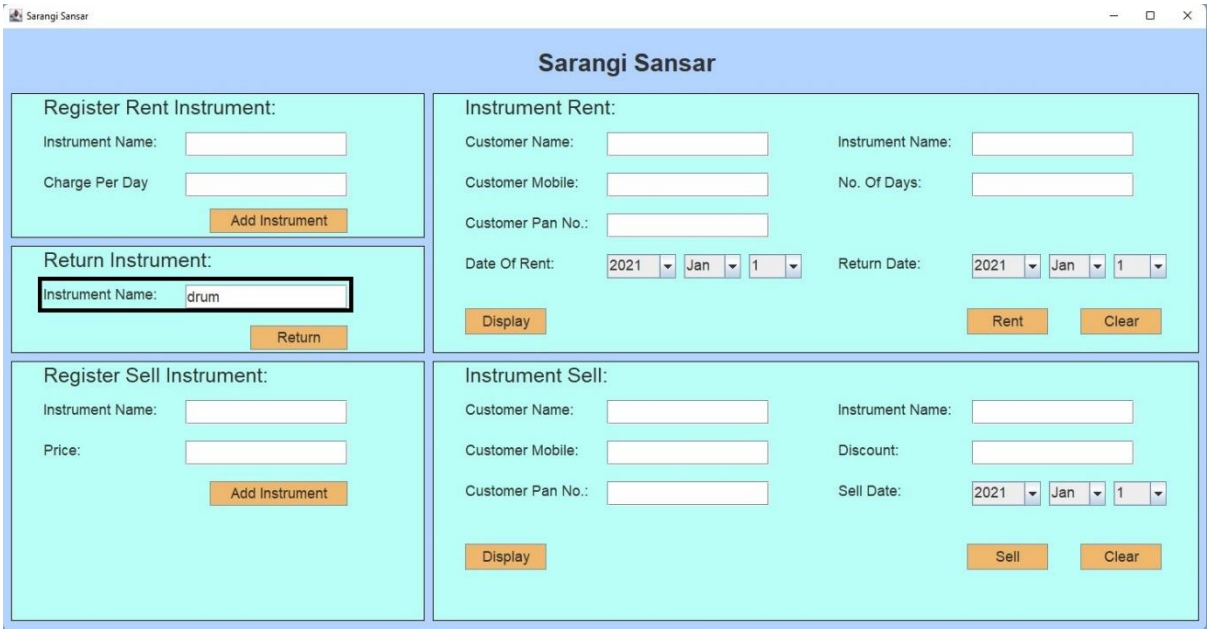

Figure 19 Before return instrument

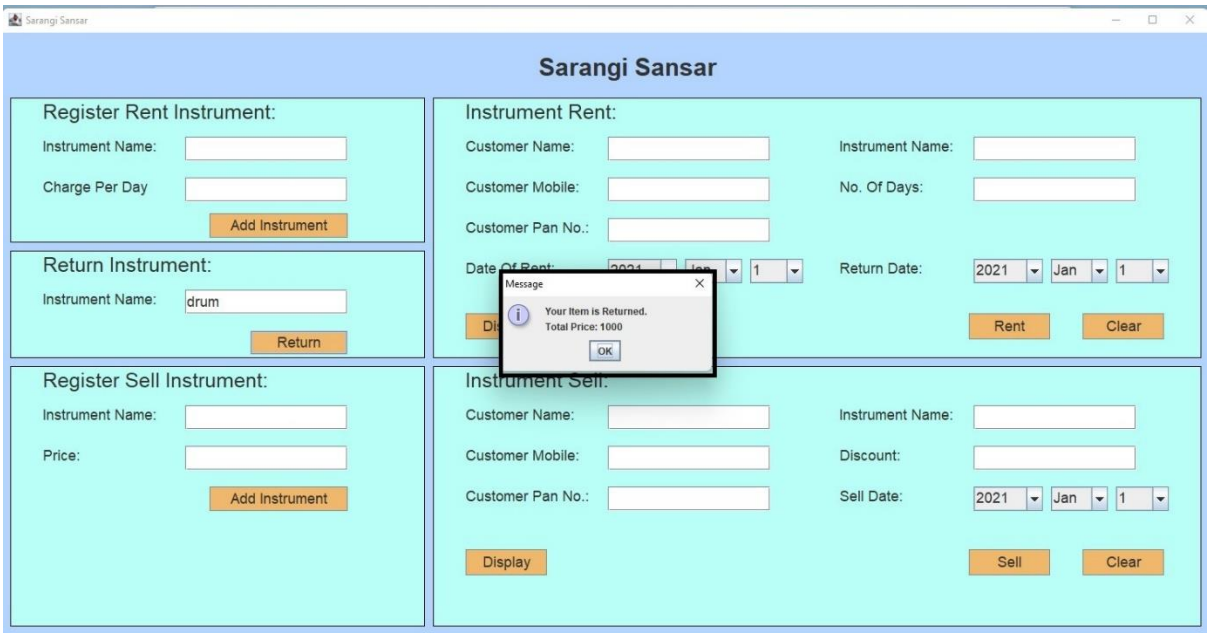

Figure 20 After Return the instrument

5.3 Test 3

Here, in the test 3, the empty field, duplicate data, wrong data type, will be checked

a) Empty Dialog Box

Here, the testing was done by pressing the button while leaving the field empty

Test No:	3.1
Objective:	Button is pressed by leaving the field empty
Action:	• Here, the text field for instrument name is left empty • The Add Instrument button is pressed.
Expected Result:	An error message should be displayed letting the user know that the field is empty.
Actual Result:	An error message was displayed letting the user know that the field is empty.
Conclusion:	The Test is successful.

Table 7 Empty field test

Here, is the evidence on the error message when the empty field is found.

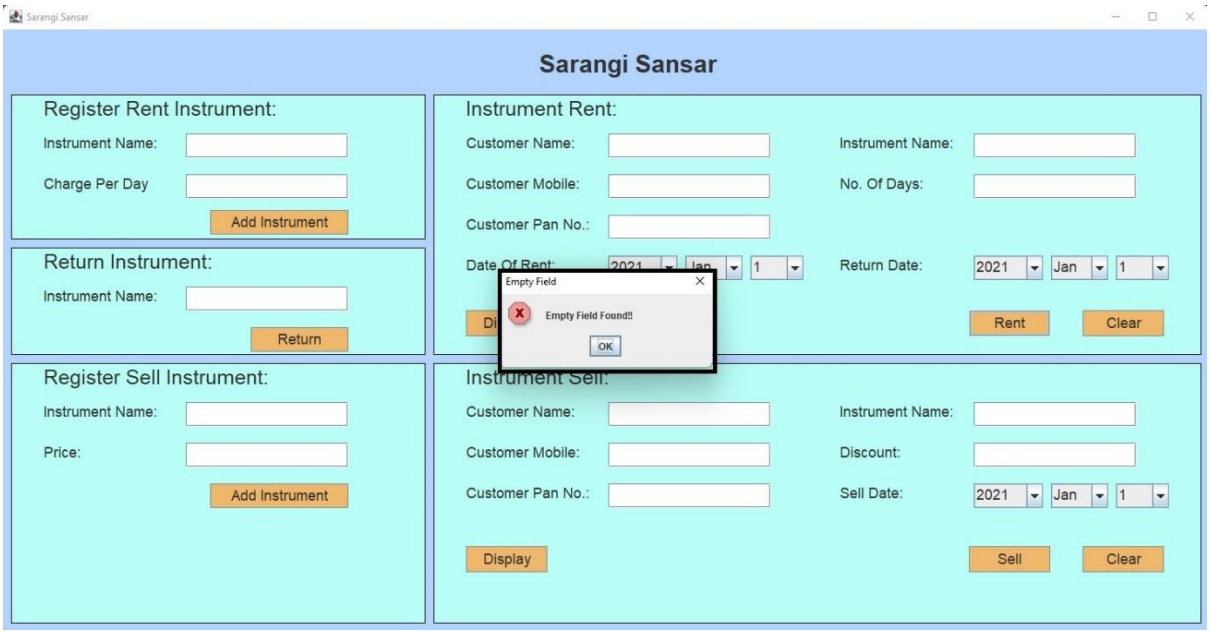

Figure 21 Empty error message

b) Wrong data type.

Here, the testing was done when the charge per day is given in the string form rather than the number format.

Test No:	3.2
Objective:	String value is given in the charge per day
Action:	• Here, the text field for charge per day is given in the form of string i.e Charge per day = abc • The Add Instrument button is pressed.
Expected Result:	An error message should be displayed letting the user know that the number format is wrong
Actual Result:	An error message was displayed letting the user know that the number format is wrong
Conclusion:	The Test is successful.

Table 8 Test for wrong data type

Here, is the evidence for an error message when the wrong data type is filled.

Figure 22 Wrong data type is inserted

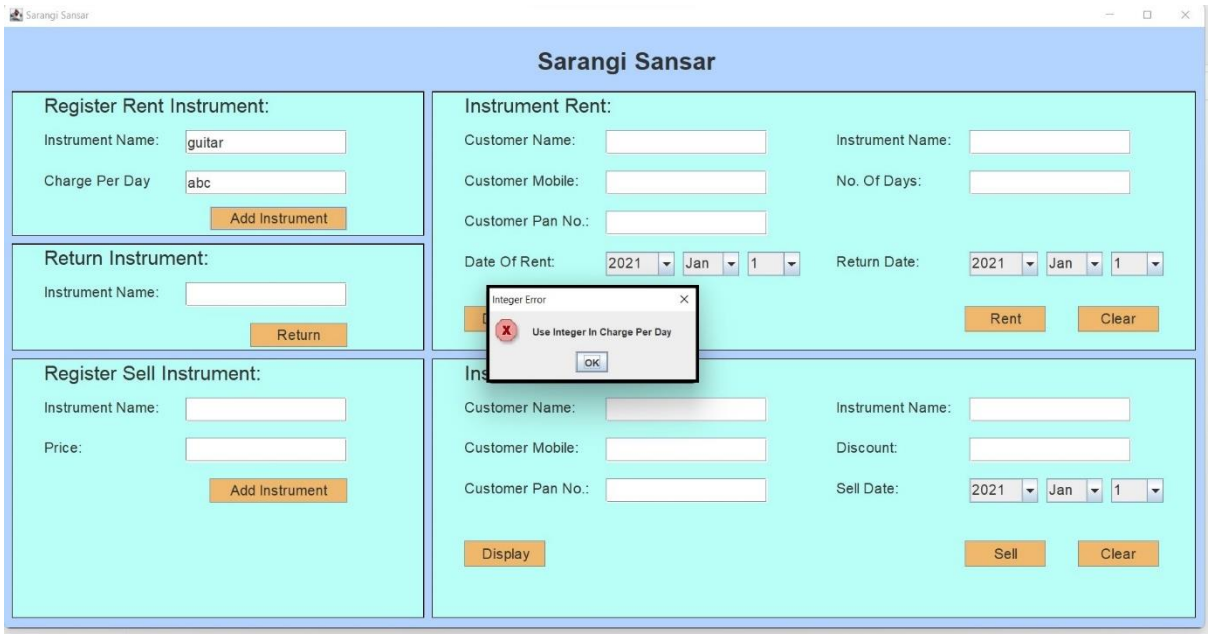

Figure 23 Error Message for wrong data type

c) Duplicate data add for rent

Here, the testing was done when the same instrument is added in the instrument twice.

Test No:	3.3
Objective:	Same instrument is added twice
Action:	• Here, the text field for instrument name and charge per day is filled to add the instrument to rent: Instrument name = “drum” Charge per day = 200 • Again the text field for instrument name is filled with the same name to add the instrument to rent • The Add Instrument button is pressed.
Expected Result:	An error message should be displayed letting the user know that the instrument is already added
Actual Result:	An error message was displayed letting the user know that the instrument is already added
Conclusion:	The Test is successful.

Table 9 Test for duplicate data add

Here, is the evidence proving that the error message is generated when the same instrument is added twice.

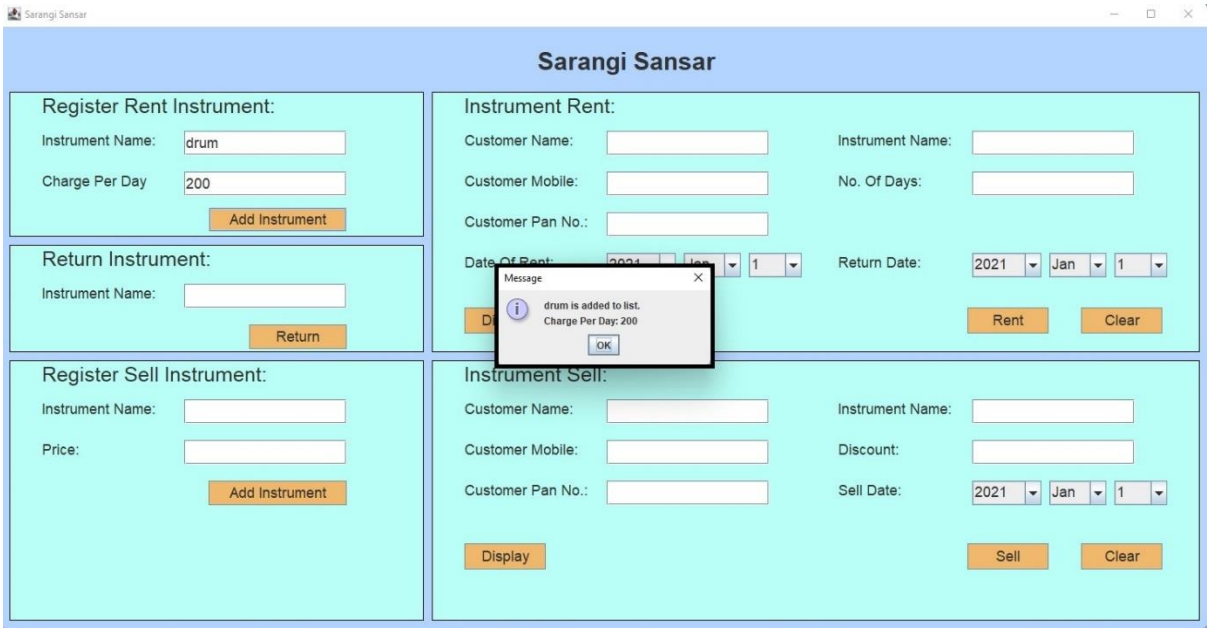

Figure 24 Adding Instrument First

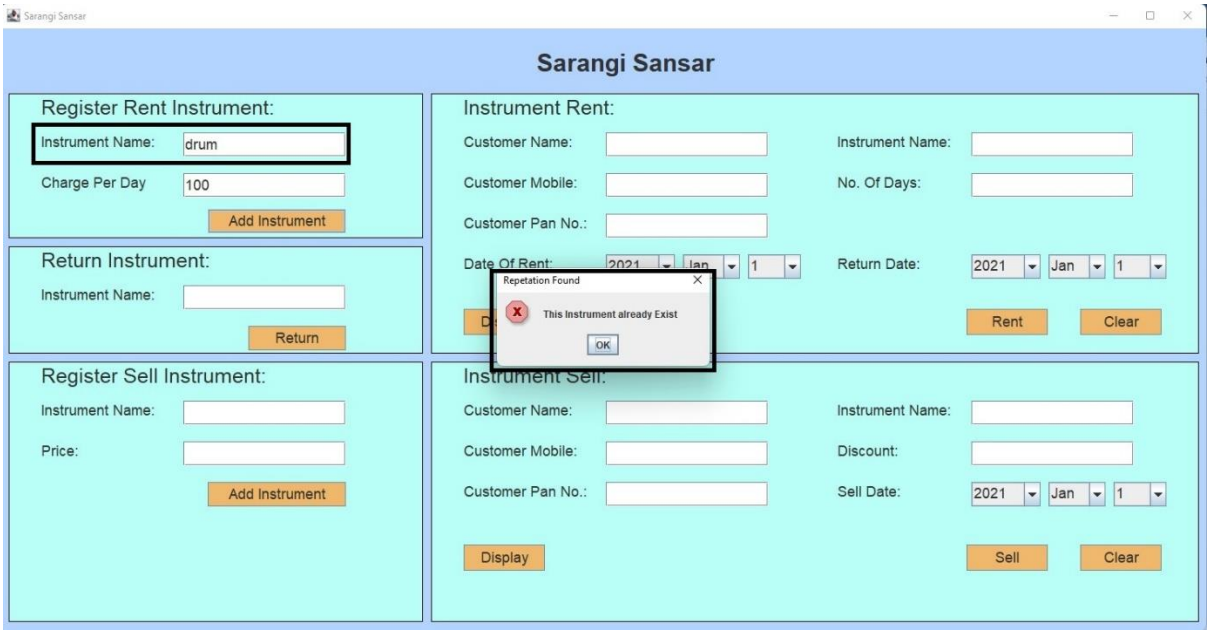

Figure 25 Adding same instrument twice

d) Renting same instrument twice

Here, the testing was done when the same instrument is rented in the instrument twice.

Test No:	3.4
Objective:	Same instrument is rented
Action:	• Here, the given value of instrument name, customer name, customer mobile, number of days, customer pan no, date of rent and date of return in the respective GUI panel are: InstrumentName = “drum” Customer Name = “Anukul Karki” Customer Mobile = “9845634310” Customer Pan No = 2341243 No of days = 5 Date of rent = “2022 jan 1” Date of return = “2022 jan 5” • Then the rent button is pressed • Again the text field for instrument name is entered same in order to rent the instrument again. • The rent button is pressed
Expected Result:	An error message should be displayed letting the user know that the instrument is already rented to the respective customer name
Actual Result:	An error message was displayed letting the user know that the instrument is already rented to the respective customer name
Conclusion:	The Test is successful.

Table 10 Renting same instrument twice

Here, is the evidence to show that the error message will be generated when the same instrument is rented twice.

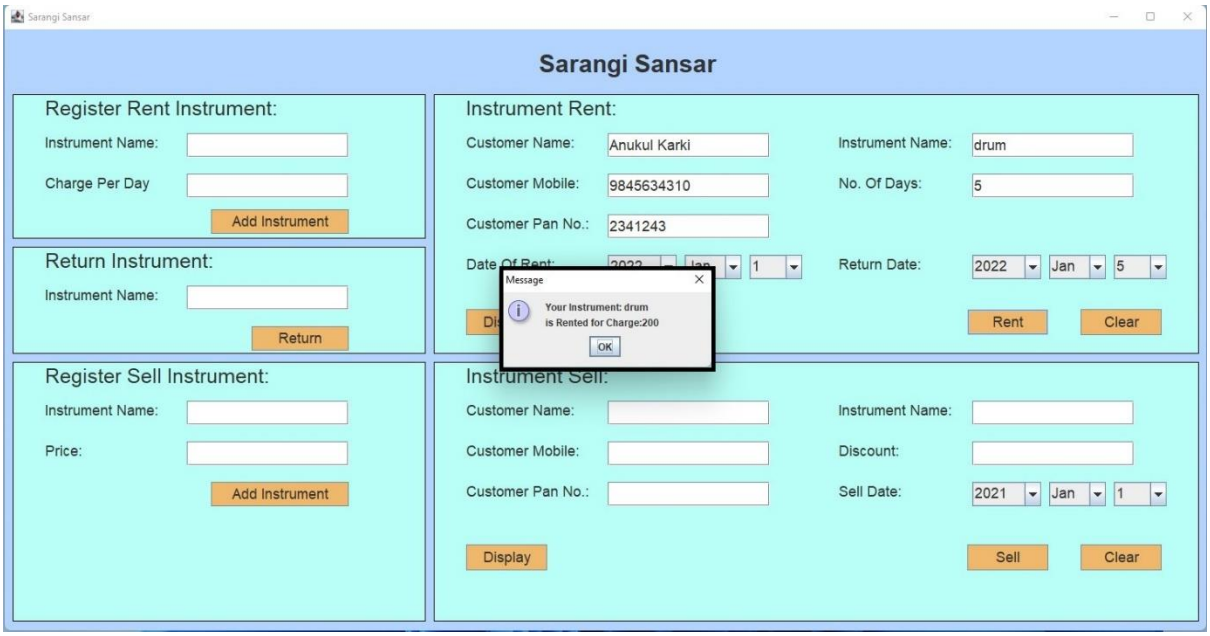

Figure 26 Renting the instrument at first

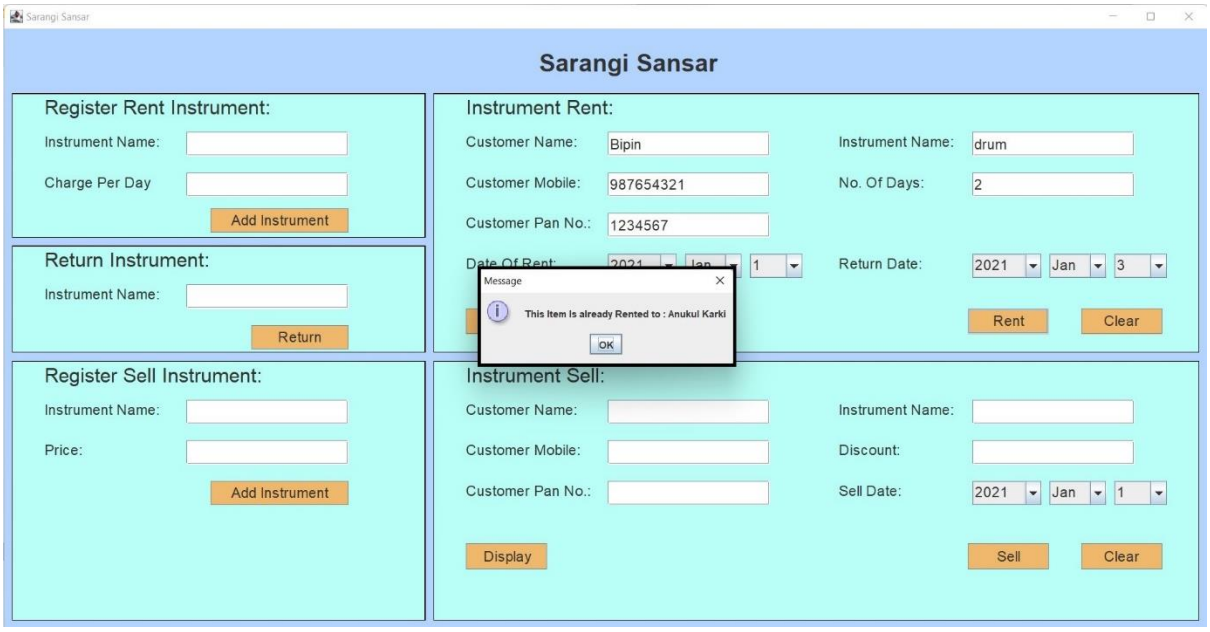

Figure 27 Renting same instrument twice

e) Selling same instrument twice

Here, the testing was done when the same instrument is sold in the instrument twice.

Test No:	3.5
Objective:	Same instrument is sold twice
Action:	• Here, the given value of instrument name, customer name, customer mobile, discount, customer pan no, sell date in the respective GUI panel are: Instrument Name = “guitar” Customer Name = “Anmol Karki” Customer Mobile = “9785423145” Customer Pan No = 123412 Discount = 5 Sell date = “2021 jan 1” • Then the sell button is pressed. • Again the instrument with the same name is entered again • Then the Sell button is pressed.
Expected Result:	An error message should be displayed letting the user know that the instrument is already sold to the respective customer name
Actual Result:	An error message was displayed letting the user know that the instrument is already sold to the respective customer name
Conclusion:	The Test is successful.

Table 11 Selling same instrument twice

Here, is the evidence in showing the error message when the same item is sold twice

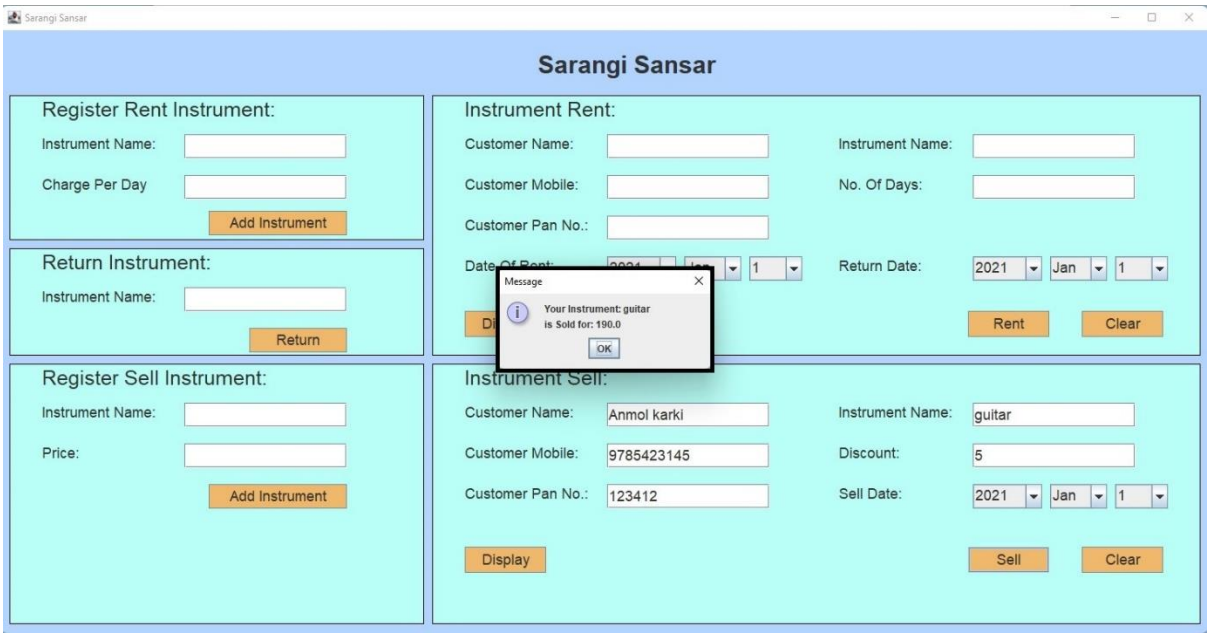

Figure 28 Item is sold at first

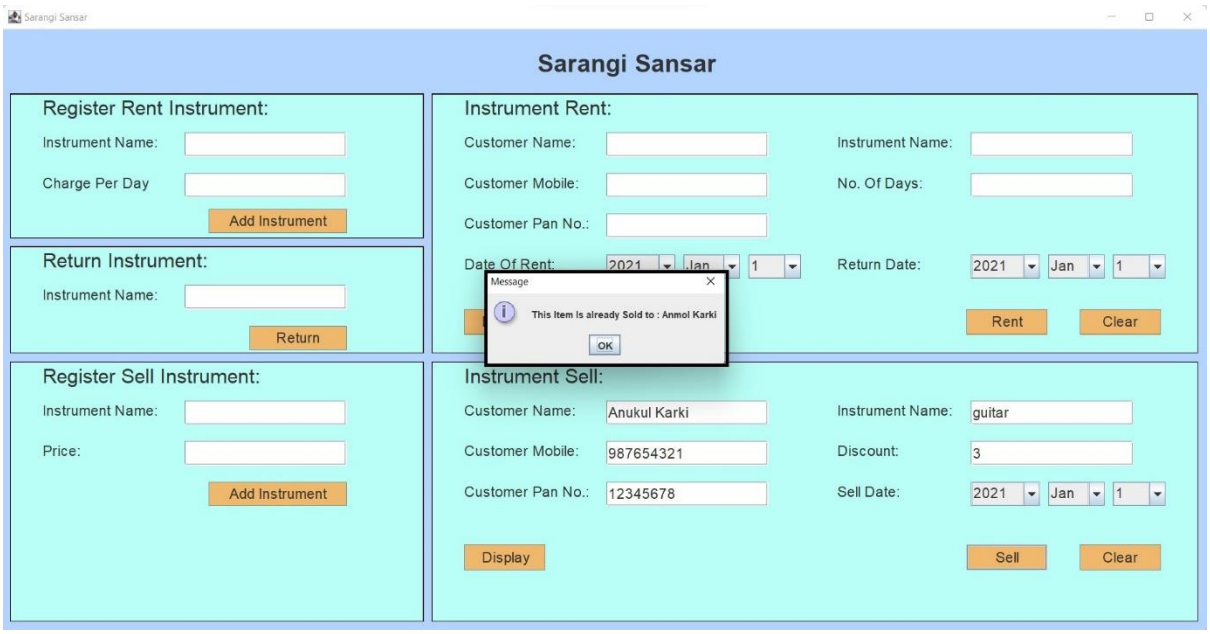

Figure 29 Same item sold twice

f) Testing for Display Button

Here, the testing is done for the display button for both selling and the renting

Test No:	3.6
Objective:	Display Button is pressed for both rent and sell
Action:	• Display Button is pressed for rent • Display Button is pressed for sell
Expected Result:	An message should be generated showing the customer detail of both the customer who have borrowed and buy the instrument
Actual Result:	An message was generated showing the customer detail of both the customer who have borrowed and buy the instrument
Conclusion:	The Test is successful.

Table 12 Testing display button

Here, is the evidence to show that the message was generated when the display button was clicked.

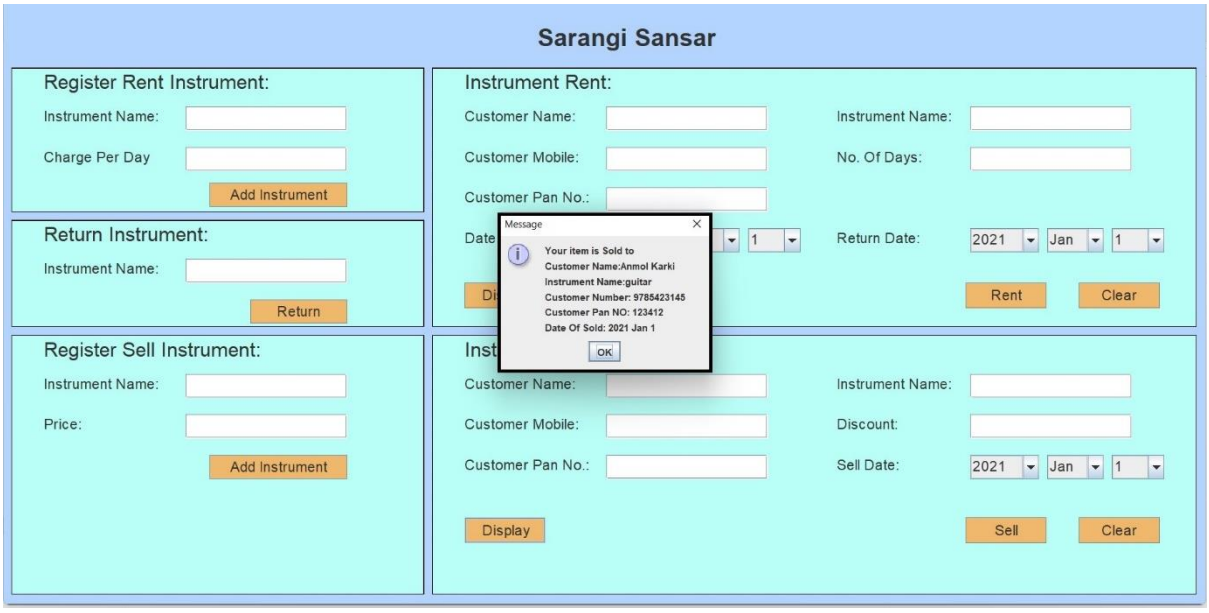

Figure 30 Display for rent

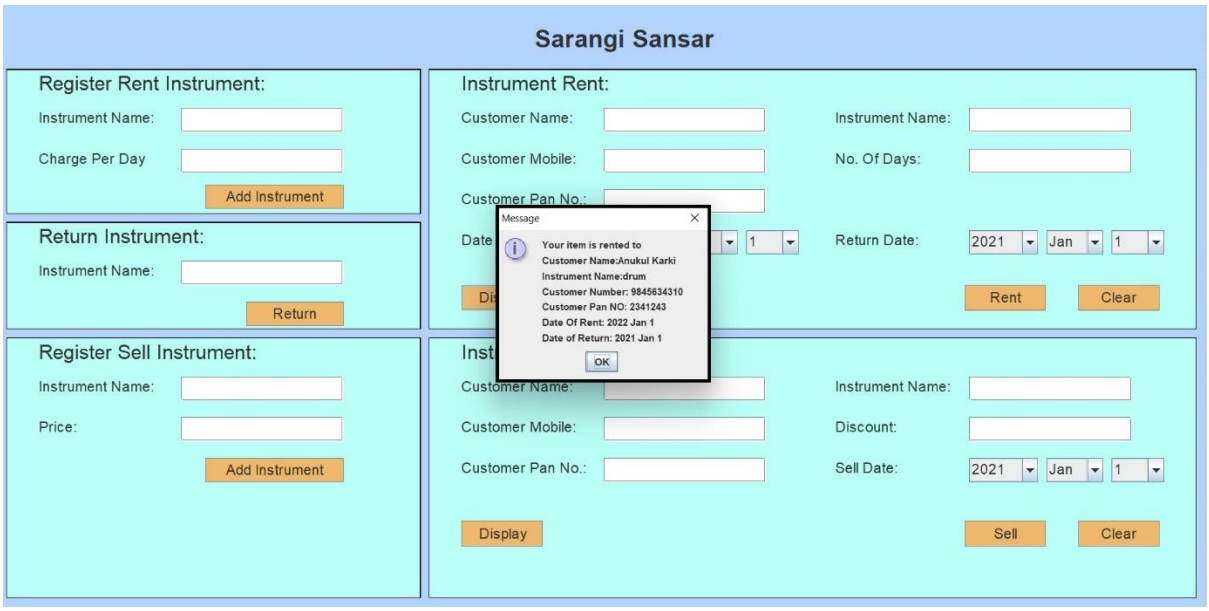

Figure 31 Display for sell

g) Display with no data

Here, in this testing the display button is clicked without any data

Test No:	3.7
Objective:	Display Button is pressed for both rent and sell without data
Action:	• Display Button is pressed for rent • Display Button is pressed for sell
Expected Result:	An message should be generated showing that the customer haven't borrowed and buy the instrument
Actual Result:	An message was be generated showing that the customer haven't borrowed and buy the instrument
Conclusion:	The Test is successful.

Table 13 Testing display button with no data

Here, is the evidence that shows when the display button is clicked without any data in the rent and sell

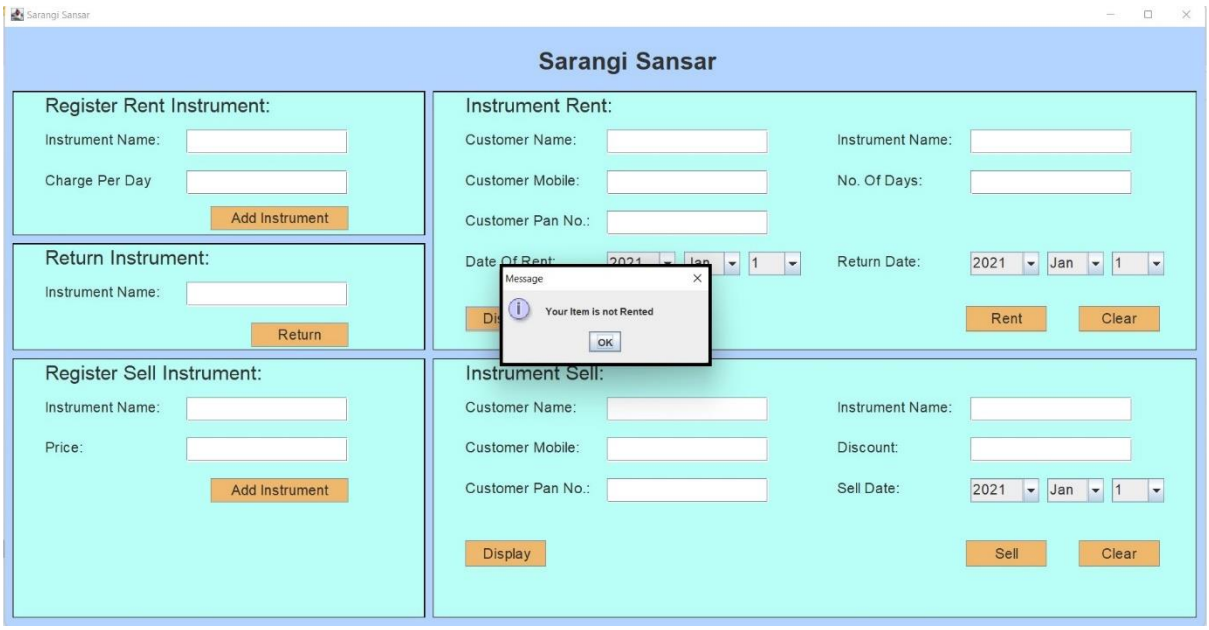

Figure 32 Display rent without data

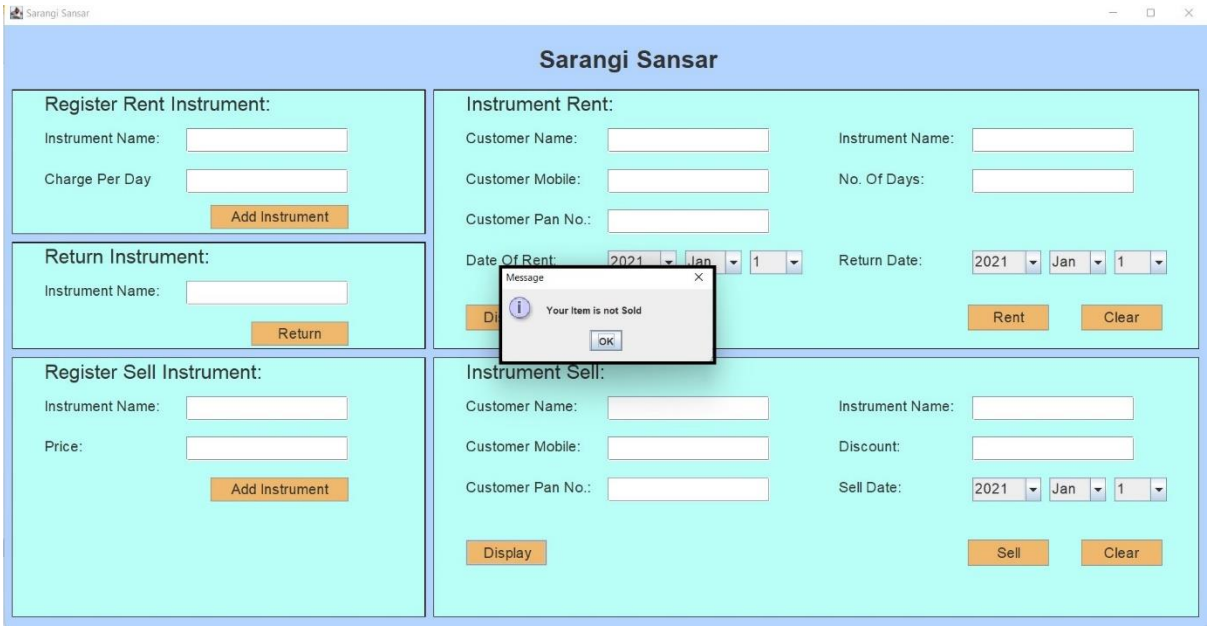

Figure 33 Display Sell without data

6. Error Detection and Correction

Different errors were occurred while developing GUI for the musical store Sarangi Sansar. There are many types of error and they are as follow.

6.1 Syntax Error: Detection

It is a type of error that was created due to different syntax than it should be. The syntax error doesn't allow the code to be compile. The syntax error that was faced while developing the code for sarangi sansar was given below.

```
public class SarangiSansar implements ActionListener
{
    long customerPanNoRent, customerPanNoSellValue;
    float discountValue, priceList;
    String instrumentNameList, instrumentNameSellList, rentInstrument, customerNameRent, customerMobileRent,  rentYearDate, rentMonthDate, rentDayDate, ren
    returnYearDate, returnMonthDate, returnDayDate, returnDate1, customerNameSellValue, customerMobileSellValue,  instrumentNameSellValue, sellYearDate, se
    Int chargePerDayList,  chargePerDay, numberOfDays;
    JPanel rentPanel , sellPanel, rentRegisterPanel, sellRegisterPanel, returnRentPanel;
    JFrame frame;

    JLabel main,registerRentHead, registerSellHead,returnRentHead, rentHead, customerName, customerMobile, customerPanNo, dateOfRent, instrumentName,instru
     noOfDays, returnDate, sellHead, customerNameSell, customerMobileSell, customerPanNoSell, instrumentNameSell, instrumentNameSellRegister, priceSellRegi
     discount, sellDate, instrumentNameReturn, instrumentNameHolderReturn;

    JTextField customerNameHolder, customerMobileHolder, customerPanNoHolder, instrumentNameHolder, noOfDaysHolder,
    customerNameSellHolder, customerMobileSellHolder, customerPanNoSellHolder, instrumentNameSellHolder,  discountHolder, instrumentNameRegisterHolder, cha
    instrumentNameSellRegisterHolder, priceSellRegisterHolder, instrumentNameReturnHolder;

    JComboBox<String> rentYear, rentMonth,  rentDay, returnYear, returnMonth, returnDay, sellYear, sellMonth, sellDay;

    JButton rent, clearRent,sell, clearSell, displaySell, displayRent, returnInstrument, addInstrumentRegister, addInstrumentSellRegister;
    ArrayList <Instrument> list  = new ArrayList<Instrument>();

    public SarangiSansar()
    {
```

Figure 34 Syntax error

Here this error is the syntax error. The syntax error was generated as the int was supposed to have the small letter 'I' but instead in the code capital letter 'I' was written so that's why the error was generated. Hence, the code wasn't able to compile.

6.2 Syntax Error: Correction

As, while compiling the code syntax error was found but it was easy to correct as changing the upper case I to lower case I was the solution of the problem. As after the code is debugged it started compiling again smoothly and was ready to use.

```
public class SarangiSansar implements ActionListener
{
    long customerPanNoRent, customerPanNoSellValue;
    float discountValue, priceList;
    String instrumentNameList, instrumentNameSellList, rentInstrument, customerNameRent, customerMobileRent,  rentYearDate, rentMonthDate, rentDayDate, rent
    returnYearDate, returnMonthDate, returnDayDate, returnDate1, customerNameSellValue, customerMobileSellValue,  instrumentNameSellValue, sellYearDate, sel
    int chargePerDayList,  chargePerDay, numberOfDays;
    JPanel rentPanel , sellPanel, rentRegisterPanel, sellRegisterPanel, returnRentPanel;
    JFrame frame;

    JLabel main,registerRentHead, registerSellHead,returnRentHead, rentHead, customerName, customerMobile, customerPanNo, dateOfRent, instrumentName,instrum
     noOfDays, returnDate, sellHead, customerNameSell, customerMobileSell, customerPanNoSell, instrumentNameSell, instrumentNameSellRegister, priceSellRegis
     discount, sellDate, instrumentNameReturn, instrumentNameHolderReturn;

    JTextField customerNameHolder, customerMobileHolder, customerPanNoHolder, instrumentNameHolder, noOfDaysHolder,
    customerNameSellHolder, customerMobileSellHolder, customerPanNoSellHolder, instrumentNameSellHolder,  discountHolder, instrumentNameRegisterHolder, char
    instrumentNameSellRegisterHolder, priceSellRegisterHolder, instrumentNameReturnHolder;

    JComboBox<String> rentYear, rentMonth,  rentDay, returnYear, returnMonth, returnDay, sellYear, sellMonth, sellDay;

    JButton rent, clearRent,sell, clearSell, displaySell, displayRent, returnInstrument, addInstrumentRegister, addInstrumentSellRegister;
    ArrayList <Instrument> list  = new ArrayList<Instrument>();

    public SarangiSansar()
    {
        //creating frame with the heading Sarangi Sansar
```

Figure 35 Syntax error fix

Hence, the code was easily fixed and it started running smoothly.

6.3 Semantics Error: Detection

It is the error whose syntax is correct but doesn't make sense in the overall code. Semantics error also doesn't let the code to be compiled. The semantic error that was faced during the development of the code is

Figure 36 Semantic Error

In this error the data type of the local variable sellExist should be Boolean but instead it was found to be int, which was the major reason behind the error. This code can be easily fixed just by changing the int to Boolean. Hence the code will run smoothly again.

6.4 Semantics Error: Correction

This semantic error was easily fixed just by changing the data type of the variable to Boolean instead of int.

```
        }catch(NumberFormatException exception)
        {
            JOptionPane.showMessageDialog(frame,"Use Integer In Charge Per Day","Integer Error",JOptionPane.ERROR_MESSAGE);
            instrumentNameRegisterHolder.setText("");
            chargePerDayRegisterHolder.setText("");
        }
    }
}
if(event.getSource() == addInstrumentSellRegister)

{
    boolean sellExist = false;
    if(instrumentNameSellRegisterHolder.getText().isEmpty() || priceSellRegisterHolder.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(frame,"Empty Fields Found!! Please Enter the value", "Empty Field", JOptionPane.ERROR_MESSAGE);
    }
    else
    {

       try
        {
            instrumentNameSellList = instrumentNameSellRegisterHolder.getText();
            priceList = Float.parseFloat(priceSellRegisterHolder.getText());

            if( list.size() == 0)
            {
                InstrumentToSell instrumentSellObj = new InstrumentToSell(instrumentNameSellList, priceList);
                list.add(instrumentSellObj);
                JOptionPane.showMessageDialog(frame,instrumentNameSellList+ " is added to list. \nPrice: "+ priceList );
```

This code was fixed just by changing the data type. Hence the code started to compile again and was ready to run again smoothly.

6.5 Logical Error: Detection

It is the type of the error that doesn't stop the code from compiling. It will run the code smoothly but the main logic that the code was supposed to run will be altered. The logical error that came along the way while developing the code was

```
boolean exist = false;
//checking if the field is empty
if(instrumentNameRegisterHolder.getText().isEmpty() || chargePerDayRegisterHolder.getText().isEmpty())
{
    JOptionPane.showMessageDialog(frame,"Empty Fields Found!! Please Enter the value", "Empty Field", JOptionPane.ERROR_MESSAGE);
}
else
{

    //try catch block to handle exception
    try
    {

        instrumentNameList = instrumentNameRegisterHolder.getText();
        chargePerDayList = Integer.parseInt(chargePerDayRegisterHolder.getText());
        //checking if the size of the list is 0
        if(list.size() == 0 )
        {
            InstrumentToRent instrumentRentObj = new InstrumentToRent(instrumentNameList, chargePerDayList);
            list.add(instrumentRentObj);
            JOptionPane.showMessageDialog(frame,chargePerDayList+ " is added to list. \nCharge Per Day: "+instrumentNameList );
        }
        else
        {
            //checking if the instrument already exist
            for(Instrument instrumentNameCheck: list)
```

Figure 37 Logical error

The consequences of this logical error is given below.

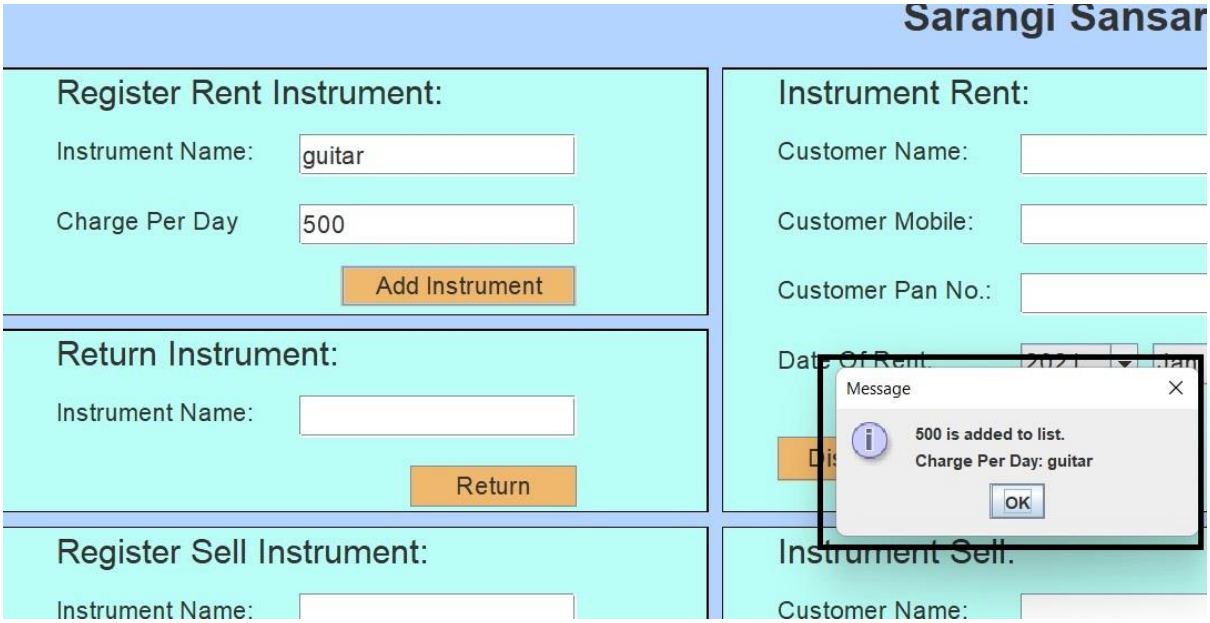

Figure 38 logical error display

Here in this logical error the name of the instrument and charge per day was interchanged which cause the logical error. It can be fixed easily just by interchanging the variable.

6.6 Logical Error: Correction

The logical error was corrected just by changing the place of variable. Which cause the code to run as it was intended again. The fix of the logical error is given below.

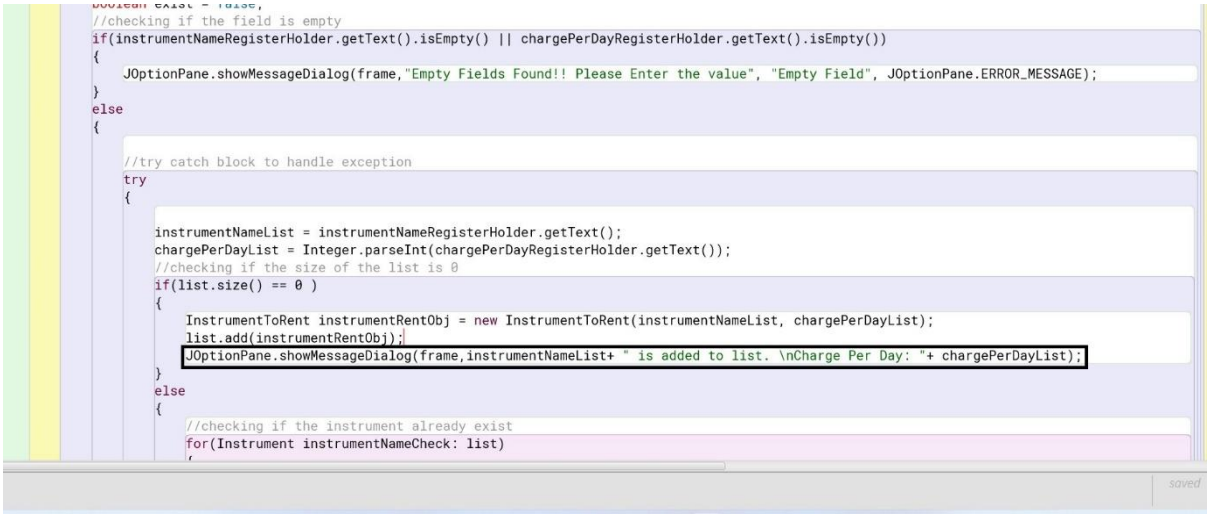

Figure 39 Logical Error fix

Display after the logical error fix is given below:

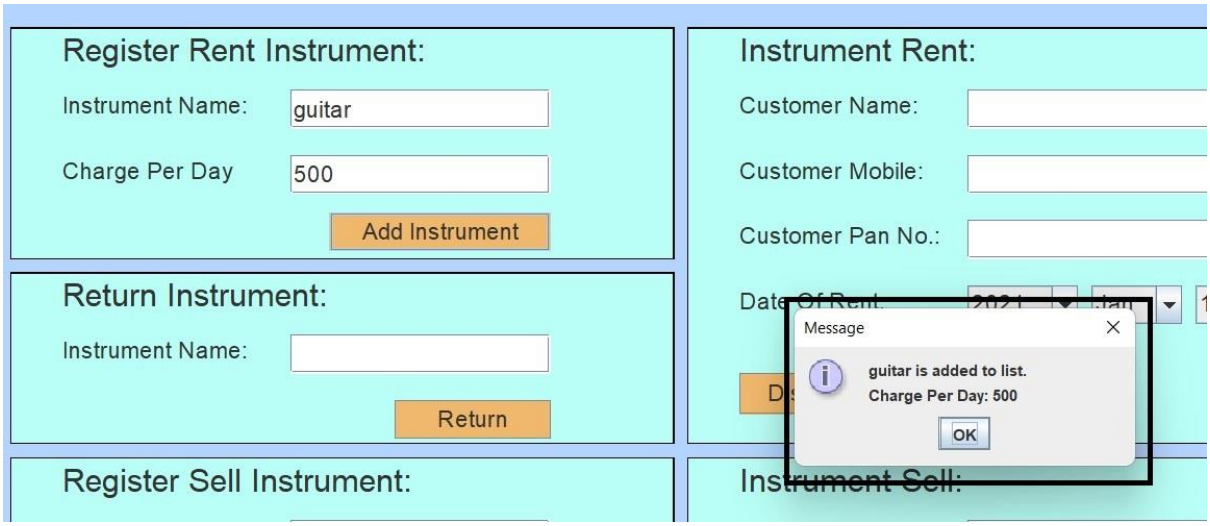

Figure 40 Display of fixed logical error

Hence after fixing the error the code started running smoothly.

7. Conclusion

The project that was given in the coursework was particularly designed to assist in handling the data for the musical store Sarangi Sansar. As a result, while developing the code and preparing the report different concept of program was learned. As developing the GUI three different classes were used which requires different core concepts in order to make the program works which made the me enthusiastic to learn more about the code.

As, a result of this course work various new features that java provides were learned. Using different java components like swing, awt was completely new experience, although it was quite hectic at the first to understand and use it with complete ease. But while working in this project, using those components was a piece of cake. Even though building GUI was becoming easy, the main difficulty was how to make that GUI attractive, user friendly and unique. But with the help of the resources provided by the tutors helped a lot and the GUI section was complete. As the GUI section was completed I thought that the main difficult part was over, but the main obstacle was about to hit hard and that was event handling. Developing the code for event handling was on another level than the event handling that was done in the workshop session. Different concepts like exception handling, object casting, traverse, working with the Array list were used in this project. Using the resources provided by the tutors weren't enough to complete it. So, to complete the projects tutors helped a lot. They gave us the concept for programs repeatedly until everyone was clear about it, they gave us different ideas on how to debug the current program. They also suggested on what can we add in the program to make it work fluently. So, now thanks to the tutors, slides from the classes that helped in completing the projects, as now developing the similar kind of the project will not be such a big deal, as most of the concepts that was used in the code was learned.

Nonetheless, this project helped in understand the core concept of the java programming language which was not possible just by using the slides, and the

lab work. This project really increased the confidence for developing the GUI and make it work. However, there is a long way to become a good programmer, and with the projects like this, that day will surely come.

References

crunchbase, 2015. *crunchbase.* [Online]
Available at: https://www.crunchbase.com/organization/moqups
[Accessed 2 8 2022].

Png, F. I., 2021. *Freeiconspng.* [Online]
Available at: https://www.freeiconspng.com/images/bluej-icon-png

Techopedia, 2020. *Techopedia.* [Online]
Available at: https://www.techopedia.com/definition/29530/bluej#:~:text=BlueJ%20is%20an%20Integrated%20Development,projects%20and%20coding%20in%20Java.
[Accessed 1 8 2022].

Appendix

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;



public class SarangiSansar implements ActionListener
{
   long customerPanNoRent, customerPanNoSellValue;
   float discountValue, priceList;
   String instrumentNameList, instrumentNameSellList, rentInstrument, customerNameRent, customerMobileRent, rentYearDate, rentMonthDate, rentDayDate, rentDate1,
   returnYearDate, returnMonthDate, returnDayDate, returnDate1, customerNameSellValue, customerMobileSellValue, instrumentNameSellValue, sellYearDate, sellMonthDate, sellDayDate, sellDate1;
   int chargePerDayList,  chargePerDay, numberOfDays;
   JPanel rentPanel , sellPanel, rentRegisterPanel, sellRegisterPanel, returnRentPanel;
   JFrame frame;

   JLabel main,registerRentHead, registerSellHead,returnRentHead, rentHead, customerName, customerMobile, customerPanNo, dateOfRent, instrumentName,instrumentNameRegister, chargePerDayRegister,
    noOfDays, returnDate, sellHead, customerNameSell, customerMobileSell, customerPanNoSell, instrumentNameSell, instrumentNameSellRegister, priceSellRegister,
    discount, sellDate, instrumentNameReturn, instrumentNameHolderReturn;

   JTextField customerNameHolder, customerMobileHolder, customerPanNoHolder, instrumentNameHolder, noOfDaysHolder,
```

customerNameSellHolder, customerMobileSellHolder, customerPanNoSellHolder, instrumentNameSellHolder, discountHolder, instrumentNameRegisterHolder, chargePerDayRegisterHolder,
instrumentNameSellRegisterHolder, priceSellRegisterHolder, instrumentNameReturnHolder;

JComboBox<String> rentYear, rentMonth, rentDay, returnYear, returnMonth, returnDay, sellYear, sellMonth, sellDay;

JButton rent, clearRent,sell, clearSell, displaySell, displayRent, returnInstrument, addInstrumentRegister, addInstrumentSellRegister;
ArrayList <Instrument> list = new ArrayList<Instrument>();

```
public SarangiSansar()
{

    frame = new JFrame("Sarangi Sansar");

    rentPanel = new JPanel();
    sellPanel = new JPanel();
    rentRegisterPanel = new JPanel();
    sellRegisterPanel = new JPanel();
    returnRentPanel = new JPanel();
    main = new JLabel("Sarangi Sansar");
    rentHead = new JLabel("Instrument Rent:");
    customerName = new JLabel("Customer Name:");
    customerMobile = new JLabel("Customer Mobile:");
    customerPanNo = new JLabel("Customer Pan No.:");
    dateOfRent = new JLabel("Date Of Rent:");
    instrumentName = new JLabel("Instrument Name:");
    noOfDays = new JLabel("No. Of Days:");
    returnDate = new JLabel("Return Date:");
    registerRentHead = new JLabel("Register Rent Instrument:");
    instrumentNameRegister = new JLabel("Instrument Name:");
```

```
        chargePerDayRegister = new JLabel("Charge Per Day");
        returnRentHead = new JLabel("Return Instrument:");
        instrumentNameReturn = new JLabel("Instrument Name:");
        sellHead = new JLabel("Instrument Sell:");
        customerNameSell = new JLabel("Customer Name:");
        customerMobileSell = new JLabel("Customer Mobile:");
        customerPanNoSell = new JLabel("Customer Pan No.:");
        instrumentNameSell = new JLabel("Instrument Name:");
        discount = new JLabel("Discount:");
        sellDate = new JLabel("Sell Date:");
        registerSellHead = new JLabel("Register Sell Instrument:");
        instrumentNameSellRegister = new JLabel("Instrument Name:");
        priceSellRegister = new JLabel("Price:");
        customerNameHolder = new JTextField();
        customerMobileHolder = new JTextField();
        customerPanNoHolder = new JTextField();
        instrumentNameHolder = new JTextField();
        noOfDaysHolder = new JTextField();
        instrumentNameRegisterHolder = new JTextField();
        chargePerDayRegisterHolder = new JTextField();
        instrumentNameReturnHolder = new JTextField();
        customerNameSellHolder = new JTextField();
        customerMobileSellHolder = new JTextField();
        customerPanNoSellHolder = new JTextField();
        instrumentNameSellHolder = new JTextField();
        discountHolder = new JTextField();
        instrumentNameSellRegisterHolder = new JTextField();
        priceSellRegisterHolder = new JTextField();
        String[] year =
{"2021","2022","2023","2024","2025","2026","2027","2028","2029","2030"};
        String[] day =
{"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20",
"21","22","23","24","25","26","27","28","29","30","31","32"};
```

```
        String[] month =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
        rentYear = new JComboBox<String>(year);
        rentMonth = new JComboBox<String>(month);
        rentDay = new JComboBox<String>(day);
        returnYear = new JComboBox<String>(year);
        returnMonth = new JComboBox<String>(month);
        returnDay = new JComboBox<String>(day);
        sellYear = new JComboBox<String>(year);
        sellMonth = new JComboBox<String>(month);
        sellDay = new JComboBox<String>(day);
        rent = new JButton("Rent");
        rent.addActionListener(this);
        clearRent = new JButton("Clear");
        clearRent.addActionListener(this);
        sell = new JButton("Sell");
        sell.addActionListener(this);
        clearSell = new JButton("Clear");
        clearSell.addActionListener(this);
        displaySell = new JButton("Display");
        displaySell.addActionListener(this);
        displayRent = new JButton("Display");
        displayRent.addActionListener(this);
        addInstrumentRegister = new JButton("Add Instrument");
        addInstrumentRegister.addActionListener(this);

        addInstrumentSellRegister = new JButton("Add Instrument");
        addInstrumentSellRegister.addActionListener(this);
        returnInstrument = new JButton("Return");
        returnInstrument.addActionListener(this);
        Font mainFont = new Font("Arial", Font.BOLD, 30);
        Font subHeadFont = new Font("Arial",Font.PLAIN, 25);
        Font font = new Font("Arial",Font.PLAIN,18);
        main.setBounds(660,26,250,35);
```

```
rentHead.setBounds(40,3,200,30);
customerName.setBounds(40,50,150,20);
customerMobile.setBounds(40,100,150,20);
customerPanNo.setBounds(40,150,170,20);
dateOfRent.setBounds(40,200,138,20);
instrumentName.setBounds(500,50,148,20);

noOfDays.setBounds(500,100,138,20);
returnDate.setBounds(500,200,138,20);
registerRentHead.setBounds(40,3,300,30);
instrumentNameRegister.setBounds(40,50,150,20);
chargePerDayRegister.setBounds(40,100,150,20);
returnRentHead.setBounds(40,3,300,30);
instrumentNameReturn.setBounds(40,50,150,20);
registerSellHead.setBounds(40,3,300,30);
instrumentNameSellRegister.setBounds(40,50,150,20);
priceSellRegister.setBounds(40,100,150,20);
sellHead.setBounds(40,3,200,30);
customerNameSell.setBounds(40,50,150,20);
customerMobileSell.setBounds(40,100,150,20);
customerPanNoSell.setBounds(40,150,170,20);
instrumentNameSell.setBounds(500,50,148,20);
discount.setBounds(500,100,138,20);
sellDate.setBounds(500,150,138,20);
customerNameHolder.setBounds(215,48,200,30);
customerMobileHolder.setBounds(215,98,200,30);
customerPanNoHolder.setBounds(215,148,200,30);
instrumentNameHolder.setBounds(665,48,200,30);
noOfDaysHolder.setBounds(665,98,200,30);
instrumentNameRegisterHolder.setBounds(215,48,200,30);
chargePerDayRegisterHolder.setBounds(215,98,200,30);
instrumentNameReturnHolder.setBounds(215,48,200,30);
instrumentNameSellRegisterHolder.setBounds(215,48,200,30);
priceSellRegisterHolder.setBounds(215,98,200,30);
```

```
customerNameSellHolder.setBounds(215,48,200,30);
customerMobileSellHolder.setBounds(215,98,200,30);
customerPanNoSellHolder.setBounds(215,148,200,30);
instrumentNameSellHolder.setBounds(665,48,200,30);
discountHolder.setBounds(665,98,200,30);
rentYear.setBounds(215,198,85,30);
rentMonth.setBounds(310,198,70,30);
rentDay.setBounds(390,198,65,30);
returnYear.setBounds(665,198,85,30);
returnMonth.setBounds(760,198,70,30);
returnDay.setBounds(840,198,65,30);
sellYear.setBounds(665,148,85,30);
sellMonth.setBounds(760,148,70,30);
sellDay.setBounds(840,148,65,30);
rent.setBounds(659,265,100,32);
clearRent.setBounds(799,265,100,32);
displayRent.setBounds(40,265,100,32);
addInstrumentRegister.setBounds(245,142,170,30);
returnInstrument.setBounds(295,98,120,30);
addInstrumentSellRegister.setBounds(245,148,170,30);
sell.setBounds(659,225,100,32);
clearSell.setBounds(799,225,100,32);
displaySell.setBounds(40,225,100,32);
main.setFont(mainFont);
rentHead.setFont(subHeadFont);
registerRentHead.setFont(subHeadFont);
returnRentHead.setFont(subHeadFont);
registerSellHead.setFont(subHeadFont);
customerName.setFont(font);
customerMobile.setFont(font);
customerPanNo.setFont(font);
dateOfRent.setFont(font);
instrumentName.setFont(font);
noOfDays.setFont(font);
```

```
returnDate.setFont(font);
instrumentNameRegister.setFont(font);
chargePerDayRegister.setFont(font);
instrumentNameReturn.setFont(font);
instrumentNameSellRegister.setFont(font);
priceSellRegister.setFont(font);
sellHead.setFont(subHeadFont);
customerNameSell.setFont(font);
customerMobileSell.setFont(font);
customerPanNoSell.setFont(font);
instrumentNameSell.setFont(font);
discount.setFont(font);
sellDate.setFont(font);
customerNameHolder.setFont(font);
customerMobileHolder.setFont(font);
customerPanNoHolder.setFont(font);
instrumentNameHolder.setFont(font);
noOfDaysHolder.setFont(font);
instrumentNameRegisterHolder.setFont(font);
chargePerDayRegisterHolder.setFont(font);
instrumentNameReturnHolder.setFont(font);
instrumentNameSellRegisterHolder.setFont(font);
priceSellRegisterHolder.setFont(font);
customerNameSellHolder.setFont(font);
customerMobileSellHolder.setFont(font);
customerPanNoSellHolder.setFont(font);
instrumentNameSellHolder.setFont(font);
discountHolder.setFont(font);
rentYear.setFont(font);
rentMonth.setFont(font);
rentDay.setFont(font);
returnYear.setFont(font);
returnMonth.setFont(font);
returnDay.setFont(font);
```

```
sellYear.setFont(font);
sellMonth.setFont(font);
sellDay.setFont(font);
rent.setFont(font);
clearRent.setFont(font);
displayRent.setFont(font);
rent.setBackground(Color.decode("#f0b86c"));
clearRent.setBackground(Color.decode("#f0b86c"));
displayRent.setBackground(Color.decode("#f0b86c"));
addInstrumentRegister.setFont(font);
addInstrumentRegister.setBackground(Color.decode("#f0b86c"));
returnInstrument.setFont(font);
returnInstrument.setBackground(Color.decode("#f0b86c"));
addInstrumentSellRegister.setFont(font);
addInstrumentSellRegister.setBackground(Color.decode("#f0b86c"));
sell.setFont(font);
clearSell.setFont(font);
displaySell.setFont(font);
sell.setBackground(Color.decode("#f0b86c"));
clearSell.setBackground(Color.decode("#f0b86c"));
displaySell.setBackground(Color.decode("#f0b86c"));
frame.getContentPane().setBackground(Color.decode("#b3d4ff"));

frame.add(main);
rentPanel.add(rentHead);
rentPanel.add(customerName);
rentPanel.add(customerMobile);
rentPanel.add(customerPanNo);
rentPanel.add(dateOfRent);
rentPanel.add(instrumentName);
rentPanel.add(noOfDays);
rentPanel.add(returnDate);

rentRegisterPanel.add(registerRentHead);
```

```
rentRegisterPanel.add(instrumentNameRegister);
rentRegisterPanel.add(chargePerDayRegister);

returnRentPanel.add(returnRentHead);
returnRentPanel.add(instrumentNameReturn);
sellRegisterPanel.add(registerSellHead);
sellRegisterPanel.add(instrumentNameSellRegister);
sellRegisterPanel.add(priceSellRegister);
sellPanel.add(sellHead);
sellPanel.add(customerNameSell);
sellPanel.add(customerMobileSell);
sellPanel.add(customerPanNoSell);
sellPanel.add(instrumentNameSell);
sellPanel.add(discount);
sellPanel.add(sellDate);
rentPanel.add(customerNameHolder);
rentPanel.add(customerMobileHolder);
rentPanel.add(customerPanNoHolder);
rentPanel.add(instrumentNameHolder);
rentPanel.add(noOfDaysHolder);

rentRegisterPanel.add(instrumentNameRegisterHolder);
rentRegisterPanel.add(chargePerDayRegisterHolder);

returnRentPanel.add(instrumentNameReturnHolder);
sellRegisterPanel.add(instrumentNameSellRegisterHolder);
sellRegisterPanel.add(priceSellRegisterHolder);

sellPanel.add(customerNameSellHolder);
sellPanel.add(customerMobileSellHolder);
sellPanel.add(customerPanNoSellHolder);
sellPanel.add(instrumentNameSellHolder);
sellPanel.add(discountHolder);
rentPanel.add(rentYear);
```

```
rentPanel.add(rentMonth);
rentPanel.add(rentDay);
rentPanel.add(returnYear);
rentPanel.add(returnMonth);
rentPanel.add(returnDay);

sellPanel.add(sellYear);
sellPanel.add(sellMonth);
sellPanel.add(sellDay);
rentPanel.add(rent);
rentPanel.add(clearRent);
rentPanel.add(displayRent);
rentRegisterPanel.add(addInstrumentRegister);
returnRentPanel.add(returnInstrument);
sellRegisterPanel.add(addInstrumentSellRegister);
sellPanel.add(sell);
sellPanel.add(clearSell);
sellPanel.add(displaySell);
rentPanel.setBounds(530,80,945,320);
sellPanel.setBounds(530,410,945,320);
rentRegisterPanel.setBounds(10,80,510,178 );
returnRentPanel.setBounds(10,268,510,132);
sellRegisterPanel.setBounds(10,410,510,320);
rentPanel.setBorder(BorderFactory.createLineBorder(Color.black));
sellPanel.setBorder(BorderFactory.createLineBorder(Color.black));
rentRegisterPanel.setBorder(BorderFactory.createLineBorder(Color.black));
returnRentPanel.setBorder(BorderFactory.createLineBorder(Color.black));
sellRegisterPanel.setBorder(BorderFactory.createLineBorder(Color.black));
rentPanel.setLayout(null);
sellPanel.setLayout(null);
rentRegisterPanel.setLayout(null);
returnRentPanel.setLayout(null);
sellRegisterPanel.setLayout(null);
rentPanel.setBackground(Color.decode("#B9FFF7"));
```

```
        sellPanel.setBackground(Color.decode("#B9FFF7"));
        returnRentPanel.setBackground(Color.decode("#B9FFF7"));
        rentRegisterPanel.setBackground(Color.decode("#B9FFF7"));
        sellRegisterPanel.setBackground(Color.decode("#B9FFF7"));
        frame.add(rentPanel);
        frame.add(sellPanel);
        frame.add(rentRegisterPanel);
        frame.add(returnRentPanel);
        frame.add(sellRegisterPanel);
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setSize(1500,780);


    }

    public void actionPerformed(ActionEvent event)
    {

        if(event.getSource() == addInstrumentRegister)
        {
            boolean exist = false;

            if(instrumentNameRegisterHolder.getText().isEmpty()                    ||
chargePerDayRegisterHolder.getText().isEmpty())
            {
                JOptionPane.showMessageDialog(frame,"Empty Fields Found!! Please
Enter the value", "Empty Field", JOptionPane.ERROR_MESSAGE);
            }
            else
            {
                try
                {
```

```
            instrumentNameList = instrumentNameRegisterHolder.getText();
            chargePerDayList =
Integer.parseInt(chargePerDayRegisterHolder.getText());
            if(list.size() == 0 )
            {
               InstrumentToRent instrumentRentObj = new
InstrumentToRent(instrumentNameList, chargePerDayList);
               list.add(instrumentRentObj);
               JOptionPane.showMessageDialog(frame,instrumentNameList+ " is
added to list. \nCharge Per Day: "+chargePerDayList );
            }
            else
            {

               for(Instrument instrumentNameCheck: list)
               {

if(instrumentNameCheck.getInstrumentName().equals(instrumentNameList) &&
instrumentNameCheck instanceof InstrumentToRent)
                  {
                     exist = true;
                     break;
                  }
               }

               if(exist == false )
               {
                  InstrumentToRent instrumentRentObj = new
InstrumentToRent(instrumentNameList, chargePerDayList);
                  list.add(instrumentRentObj);
                  JOptionPane.showMessageDialog(frame,instrumentNameList+ " is
added to list. \nCharge Per Day: "+ chargePerDayList );
               }
               else
```

```
                {
                    JOptionPane.showMessageDialog(frame,"This Instrument already Exist","Repetation Found", JOptionPane.ERROR_MESSAGE);
                }
            }

            instrumentNameRegisterHolder.setText("");
            chargePerDayRegisterHolder.setText("");
        }catch(NumberFormatException exception)
        {
            JOptionPane.showMessageDialog(frame,"Use Integer In Charge Per Day","Integer Error",JOptionPane.ERROR_MESSAGE);
            instrumentNameRegisterHolder.setText("");
            chargePerDayRegisterHolder.setText("");
        }
      }
    }
    if(event.getSource() == addInstrumentSellRegister)

    {
        boolean sellExist = false;
        if(instrumentNameSellRegisterHolder.getText().isEmpty() || priceSellRegisterHolder.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(frame,"Empty Fields Found!! Please Enter the value", "Empty Field", JOptionPane.ERROR_MESSAGE);
        }
        else
        {

          try
          {
            instrumentNameSellList = instrumentNameSellRegisterHolder.getText();
            priceList = Float.parseFloat(priceSellRegisterHolder.getText());
```

```
            if( list.size() == 0)
            {
               InstrumentToSell instrumentSellObj = new
InstrumentToSell(instrumentNameSellList, priceList);
               list.add(instrumentSellObj);
               JOptionPane.showMessageDialog(frame,instrumentNameSellList+ "
is added to list. \nPrice: "+ priceList );

            }
            else
            {
               for(Instrument instrumentNameSellCheck : list)
               {

if(instrumentNameSellCheck.getInstrumentName().equals(instrumentNameSellList)
&& instrumentNameSellCheck instanceof InstrumentToSell)
                  {
                     sellExist = true;
                  }
               }
               if(sellExist == false )
               {
                  InstrumentToSell instrumentSellObj = new
InstrumentToSell(instrumentNameSellList, priceList);
                  list.add(instrumentSellObj);
                  JOptionPane.showMessageDialog(frame,instrumentNameSellList+
" is added to list. \nPrice: "+ priceList);
               }
               else
               {
                  JOptionPane.showMessageDialog(frame,"This Instrument already
Exist","Repetation Found", JOptionPane.ERROR_MESSAGE);
               }
```

```
                }
                instrumentNameSellRegisterHolder.setText("");
                priceSellRegisterHolder.setText("");

            }catch(NumberFormatException exception)
            {
                JOptionPane.showMessageDialog(frame,"Use Integer","Integer Error",JOptionPane.ERROR_MESSAGE);
                instrumentNameSellRegisterHolder.setText("");
                priceSellRegisterHolder.setText("");
            }
        }

    }
    if(event.getSource() == rent)
    {
        if(instrumentNameHolder.getText().isEmpty() || customerNameHolder.getText().isEmpty() || customerPanNoHolder.getText().isEmpty() ||
        customerMobileHolder.getText().isEmpty() || noOfDaysHolder.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(frame,"Empty Field Found!!", "Empty Field", JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            try
            {
                rentInstrument = instrumentNameHolder.getText();
                numberOfDays = Integer.parseInt(noOfDaysHolder.getText());
                customerNameRent = customerNameHolder.getText();
                customerMobileRent = customerMobileHolder.getText();
```

```
            customerPanNoRent =
Long.parseLong(customerPanNoHolder.getText());
            rentYearDate = rentYear.getSelectedItem().toString();
            rentMonthDate = rentMonth.getSelectedItem().toString();
            rentDayDate = rentDay.getSelectedItem().toString();
            rentDate1 = rentYearDate +" " +  rentMonthDate +" " +  rentDayDate;

            returnYearDate = sellYear.getSelectedItem().toString();
            returnMonthDate = sellMonth.getSelectedItem().toString();
            returnDayDate = sellDay.getSelectedItem().toString();
            returnDate1 = returnYearDate +" " + returnMonthDate + " " +
returnDayDate;

            int rentNumber = 0;
            for(Instrument instrument: list)
            {
               //checking if the instrument name is same or not
               if(instrument.getInstrumentName().equals(rentInstrument))
               {
                  if(instrument instanceof InstrumentToRent)
                  {
                     InstrumentToRent instrumentRent =
(InstrumentToRent)instrument;

                     if(instrumentRent.getIsRented())
                     {
                        for(Instrument checkName : list)
                        {
                           if(checkName.getInstrumentName().equals(rentInstrument)
&& instrumentRent.getIsRented() && checkName instanceof InstrumentToRent)
                           {
                              JOptionPane.showMessageDialog(frame,"This Item Is
already Rented to : " + checkName.getCustomerName());
                              break;
```

```
                        }
                    }
                    break;
                }
                else
                {
                    instrumentRent.Rent(customerNameRent,
customerMobileRent, customerPanNoRent, rentDate1, returnDate1, numberOfDays);
                    JOptionPane.showMessageDialog(frame, "Your Instrument: "+
rentInstrument+" \nis Rented for Charge:" + instrumentRent.getChargePerDay());

                    customerNameHolder.setText("");
                    customerMobileHolder.setText("");
                    customerPanNoHolder.setText("");
                    instrumentNameHolder.setText("");
                    noOfDaysHolder.setText("");
                    rentYear.setSelectedIndex(0);
                    rentMonth.setSelectedIndex(0);
                    rentDay.setSelectedIndex(0);
                    returnYear.setSelectedIndex(0);
                    returnMonth.setSelectedIndex(0);
                    returnDay.setSelectedIndex(0);

                    break;
                }
            }

        }
        rentNumber++;


    }
    if(rentNumber == list.size())
    {
```

```
                    JOptionPane.showMessageDialog(frame, "No Instrument
Available", "Not Found", JOptionPane.ERROR_MESSAGE);
                }
            }
            catch(NumberFormatException exception1)
            {
                JOptionPane.showMessageDialog(frame,"Use Days In Integer", "Intger
Error", JOptionPane.ERROR_MESSAGE);
            }
        }

    }
    if(event.getSource() == sell)
    {
        if(customerNameSellHolder.getText().isEmpty()                    ||
customerMobileSellHolder.getText().isEmpty()                    ||
customerPanNoSellHolder.getText().isEmpty() ||
        instrumentNameSellHolder.getText().isEmpty()                    ||
discountHolder.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(frame,"Empty Field Found!!", "Empty
Field", JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            try
            {
                customerNameSellValue = customerNameSellHolder.getText();
                customerMobileSellValue = customerMobileSellHolder.getText();
                instrumentNameSellValue = instrumentNameSellHolder.getText();
                discountValue = Float.parseFloat(discountHolder.getText());
                customerPanNoSellValue                                    =
Long.parseLong(customerPanNoSellHolder.getText());
                sellYearDate = sellYear.getSelectedItem().toString();
```

```
            sellMonthDate = sellMonth.getSelectedItem().toString();
            sellDayDate = sellDay.getSelectedItem().toString();
            sellDate1 = sellYearDate+" " + sellMonthDate+ " " + sellDayDate;

            int sellNumber = 0;

            for(Instrument instrument : list)
            {

if(instrument.getInstrumentName().equals(instrumentNameSellValue))
                {

                    if(instrument instanceof InstrumentToSell )
                    {
                        InstrumentToSell instrumentSell = (InstrumentToSell) instrument;
                        if(instrumentSell.getIsSold())
                        {
                            for(Instrument checkName : list)
                            {

if(checkName.getInstrumentName().equals(instrumentNameSellValue)          &&
checkName instanceof InstrumentToSell)
                                {
                                    JOptionPane.showMessageDialog(frame,"This  Item  Is
already Sold to : " + checkName.getCustomerName());
                                }
                            }
                            break;
                        }
                        else
                        {
                            instrumentSell.sellInstrument(customerNameSellValue,
customerMobileSellValue, customerPanNoSellValue, sellDate1, discountValue);
```

```
                    JOptionPane.showMessageDialog(frame, "Your Instrument: "+
instrumentNameSellValue + " \nis Sold for: "+ (instrumentSell.getPrice() -
(instrumentSell.getPrice() * (instrumentSell.getDiscountPercent()/100 ))));
                    customerNameSellHolder.setText("");
                    customerMobileSellHolder.setText("");
                    customerPanNoSellHolder.setText("");
                    instrumentNameSellHolder.setText("");
                    discountHolder.setText("");
                    sellYear.setSelectedIndex(0);
                    sellMonth.setSelectedIndex(0);
                    sellDay.setSelectedIndex(0);
                    break;

                }

            }
          }
          sellNumber++;
        }
        if(sellNumber == list.size())
        {
            JOptionPane.showMessageDialog(frame, "No Instrument
Available", "Not Found", JOptionPane.ERROR_MESSAGE);
        }

      }
      catch(NumberFormatException event1)
      {
        JOptionPane.showMessageDialog(frame,"Use in intger format", "Integer
Error", JOptionPane.ERROR_MESSAGE);
      }

    }
  }
```

```
if(event.getSource() == clearRent)
{
    //clearing all the data
    instrumentNameRegisterHolder.setText("");
    chargePerDayRegisterHolder.setText("");
    customerNameHolder.setText("");
    customerMobileHolder.setText("");
    customerPanNoHolder.setText("");
    instrumentNameHolder.setText("");
    noOfDaysHolder.setText("");
    instrumentNameReturnHolder.setText("");
    rentYear.setSelectedIndex(0);
    rentMonth.setSelectedIndex(0);
    rentDay.setSelectedIndex(0);
    returnYear.setSelectedIndex(0);
    returnMonth.setSelectedIndex(0);
    returnDay.setSelectedIndex(0);
}
if(event.getSource() == clearSell)
{
    //clearing all the data
    instrumentNameSellRegisterHolder.setText("");
    priceSellRegisterHolder.setText("");
    customerNameSellHolder.setText("");
    customerMobileSellHolder.setText("");
    customerPanNoSellHolder.setText("");
    instrumentNameSellHolder.setText("");
    discountHolder.setText("");
    sellYear.setSelectedIndex(0);
    sellMonth.setSelectedIndex(0);
    sellDay.setSelectedIndex(0);
}
if(event.getSource() == returnInstrument )
```

```
    {
        if(instrumentNameReturnHolder.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(frame,"Empty Field Found", "Empty Field", JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            String returnInstrumentName = instrumentNameReturnHolder.getText();
            boolean itemCheck = false;
            for(Instrument instrument: list)
            {
                if(instrument.getInstrumentName().equals(returnInstrumentName))
                {
                    if(instrument instanceof InstrumentToRent)
                    {
                        InstrumentToRent instrumentRent = (InstrumentToRent) instrument;
                        itemCheck = true;
                        if(instrumentRent.getIsRented() == true)
                        {

                            JOptionPane.showMessageDialog(frame,"Your Item is Returned.\nTotal Price: "+ instrumentRent.getChargePerDay() * instrumentRent.getNoOfDays());
                            instrumentRent.Return();
                            instrumentNameReturnHolder.setText("");
                        }
                        else
                        {

                            JOptionPane.showMessageDialog(frame,"Your Item is not Rented", "Rent Not Found", JOptionPane.ERROR_MESSAGE);
                        }
                        break;
```

```
                }
            }
        }
        if(itemCheck == false)
        {
            JOptionPane.showMessageDialog(frame,"No Item Found","Not Found",
JOptionPane.ERROR_MESSAGE);
        }
      }
    }
    if(event.getSource() == displayRent )

    {
        int counter = 0 ;
        int rentCounter = 0;
        for(Instrument instrument: list)
        {
            if(instrument instanceof InstrumentToRent)
            {
                InstrumentToRent instrumentRent = (InstrumentToRent) instrument;

                instrumentRent.Display();
                counter ++;
                if(instrumentRent.getIsRented() == false)
                {
                    rentCounter++;

                }
                else
                {
                    JOptionPane.showMessageDialog(frame,"Your item is rented to
\nCustomer Name:" + instrumentRent.getCustomerName()+"\nInstrument Name:" +
instrumentRent.getInstrumentName()+ "\nCustomer Number: " +
instrumentRent.getCustomerMobile()+
```

```
                "\nCustomer Pan NO: "+ instrumentRent.getCustomerPanNo()+
"\nDate Of Rent: "+ instrumentRent.getDateOfRent() + "\nDate of Return: "+
instrumentRent.getDateOfReturn());
            }


        }
    }
    if(counter == rentCounter)
    {
        JOptionPane.showMessageDialog(frame,"Your Item is not Rented");
    }

}
if(event.getSource() == displaySell)
{
    int sellCustomer = 0;
    int sellNumber = 0;
    for(Instrument instrument: list)
    {
        if(instrument instanceof InstrumentToSell)
        {
            InstrumentToSell instrumentSell = (InstrumentToSell) instrument;
            instrumentSell.Display();
            sellNumber++;
            if(instrumentSell.getIsSold() == false)
            {
                sellCustomer++;

            }
            else
            {
                JOptionPane.showMessageDialog(frame,"Your item is Sold to
\nCustomer Name:" + instrumentSell.getCustomerName()+"\nInstrument Name:" +
```

```
instrumentSell.getInstrumentName()+ "\nCustomer Number: " +
instrumentSell.getCustomerMobile()+
                    "\nCustomer Pan NO: "+ instrumentSell.getCustomerPanNo()+
"\nDate Of Sold: "+ instrumentSell.getSellDate());
                }


            }


        }
        if(sellCustomer == sellNumber)
        {
            JOptionPane.showMessageDialog(frame,"Your Item is not Sold");
        }
      }

   }


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

}
```