# Table of Contents

[Table of Contents 1](#_Toc102568455)

[Assignment 1 – Federation Car Rental (FedCR) 2](#_Toc102568456)

[1. Use Case Diagram to review the requirements 3](#_Toc102568457)

[2. Use Case Diagram of the program generated 4](#_Toc102568458)

[3. Class diagram: 5](#_Toc102568459)

[4. Sequence Diagram 6](#_Toc102568460)

[5. Output: 7](#_Toc102568461)

[6. Source Code 10](#_Toc102568462)

[6.1 Main.java 10](#_Toc102568463)

[6.2 Menu.java 10](#_Toc102568464)

[6.3 Payment.java 20](#_Toc102568465)

[StandardPayment.java 20](#_Toc102568466)

[6.4 LoyaltyPayment.java 22](#_Toc102568467)

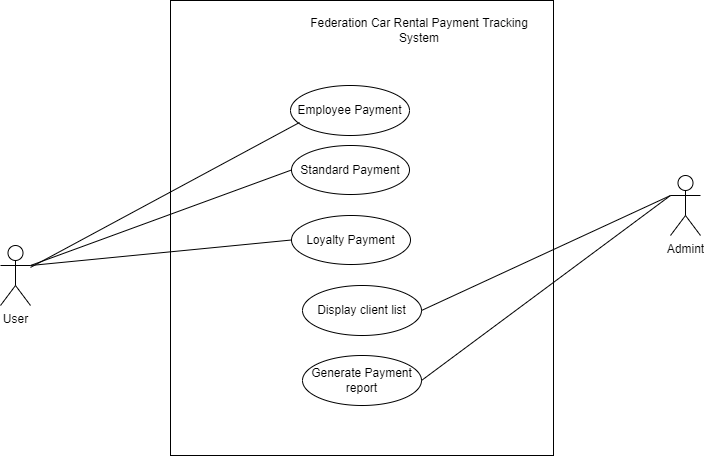
[6.5 EmployeePayment.java 24](#_Toc102568468)

[Knowledge Gained 26](#_Toc102568469)

# Assignment 1 – Federation Car Rental (FedCR)

## Use Case Diagram to review the requirements

Use case diagram is the primary modelling label for the system requirements. It includes the behavior of the actors and their responsibilities within the software requirements.

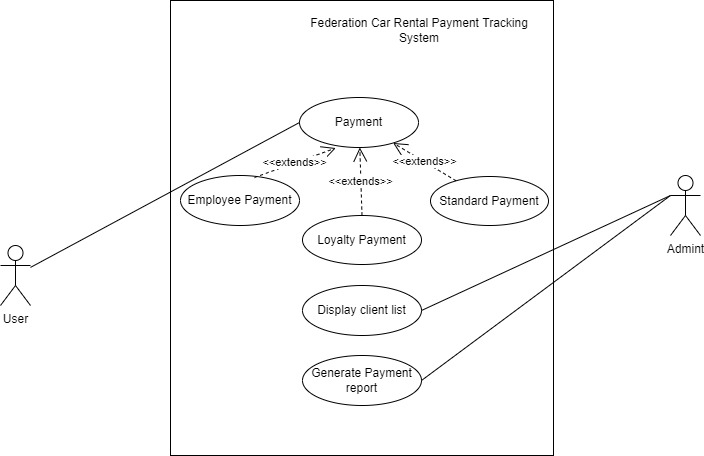


In the above use case, user should be able to perform payments like:

* Employee Payment
* Standard Payment
* Loyalty Payment

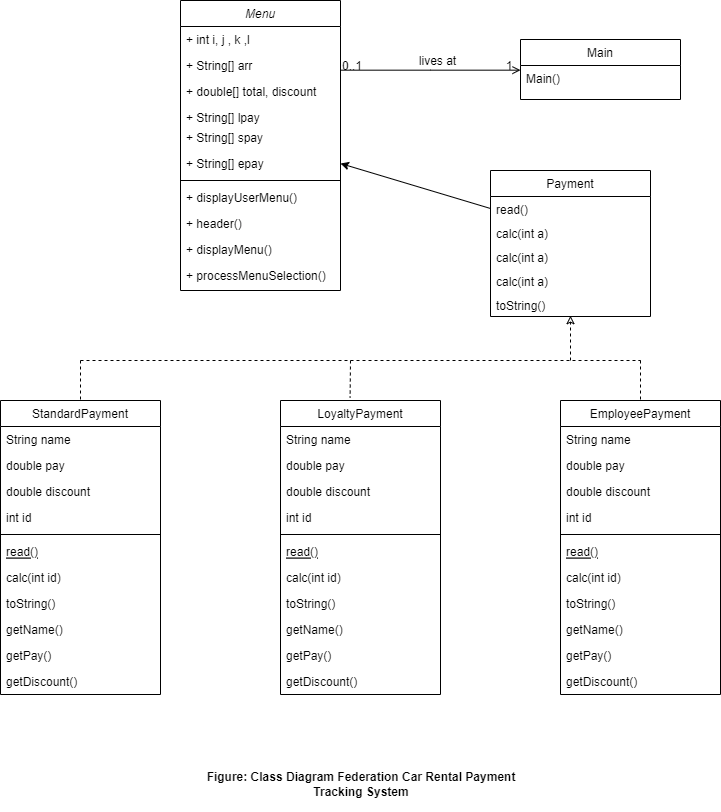
On the other hand admin has the role to display client lists and generate the payment report from the transaction performed.

## Use Case Diagram of the program generated



The extend relationship describes that types of payments includes the behavior of the base use case Payment.

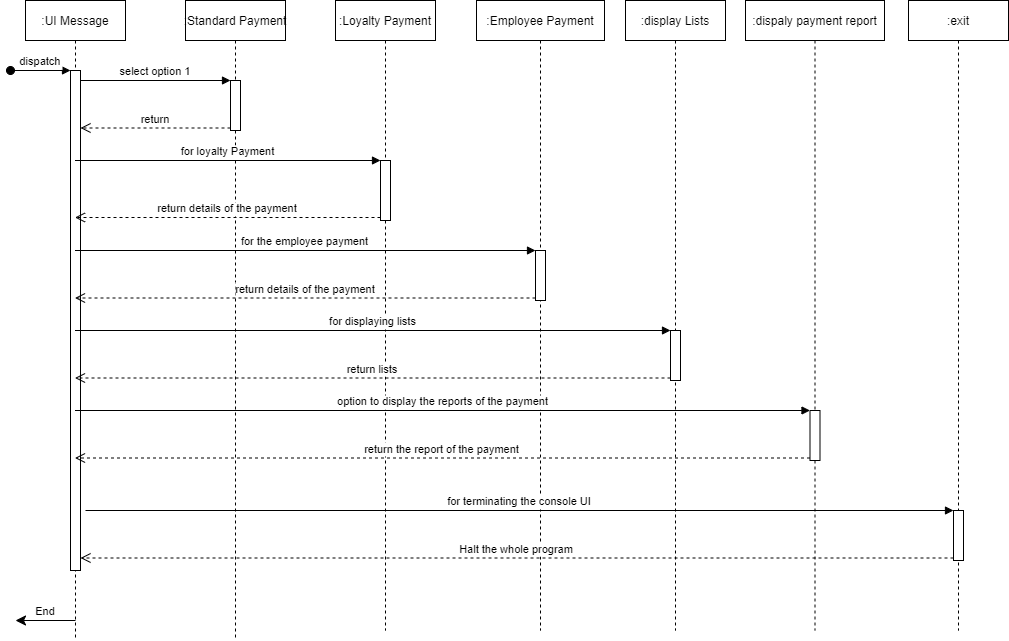
## Class diagram:



The above class diagram represents the system structure defining its classes, their attributes, operations and relationship among all the objects within the inside and outside of the class scope.

Here, Payment as an interface is used to define the class allowing classes to have multiple inheritance and to achieve abstraction. Interface includes abstract methods having no body section. In an interface we general define default and static methods.

## Sequence Diagram



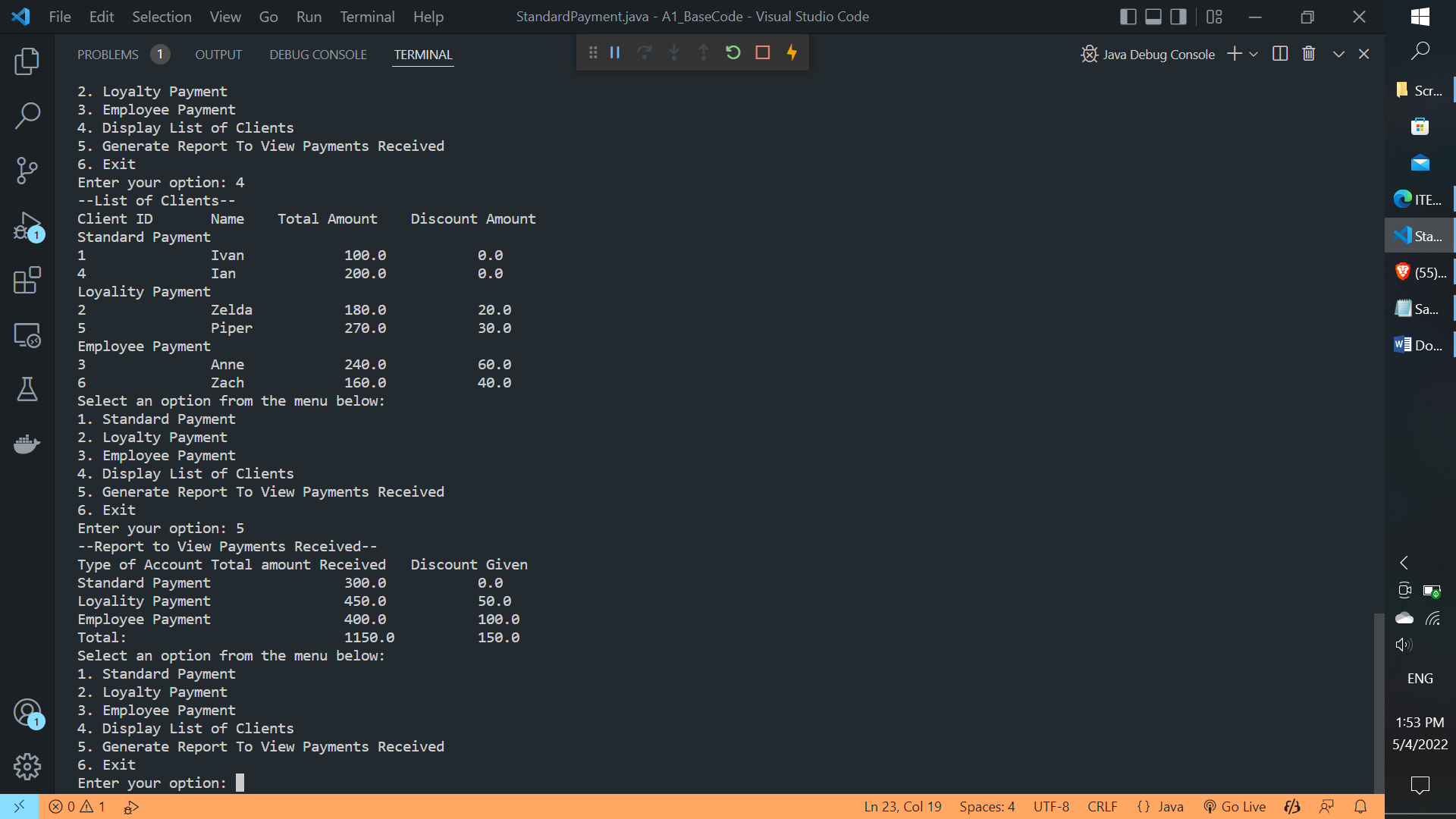
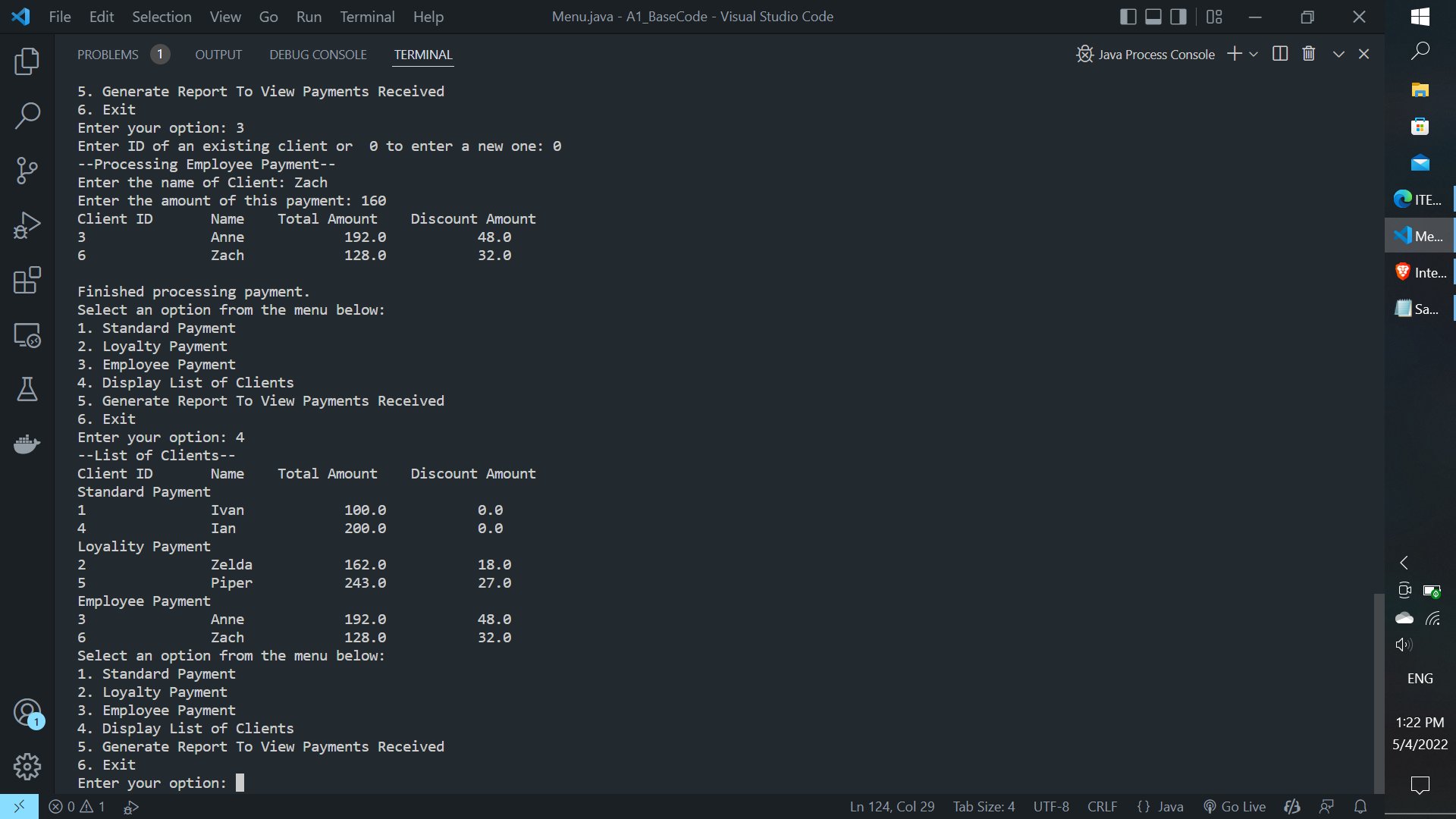
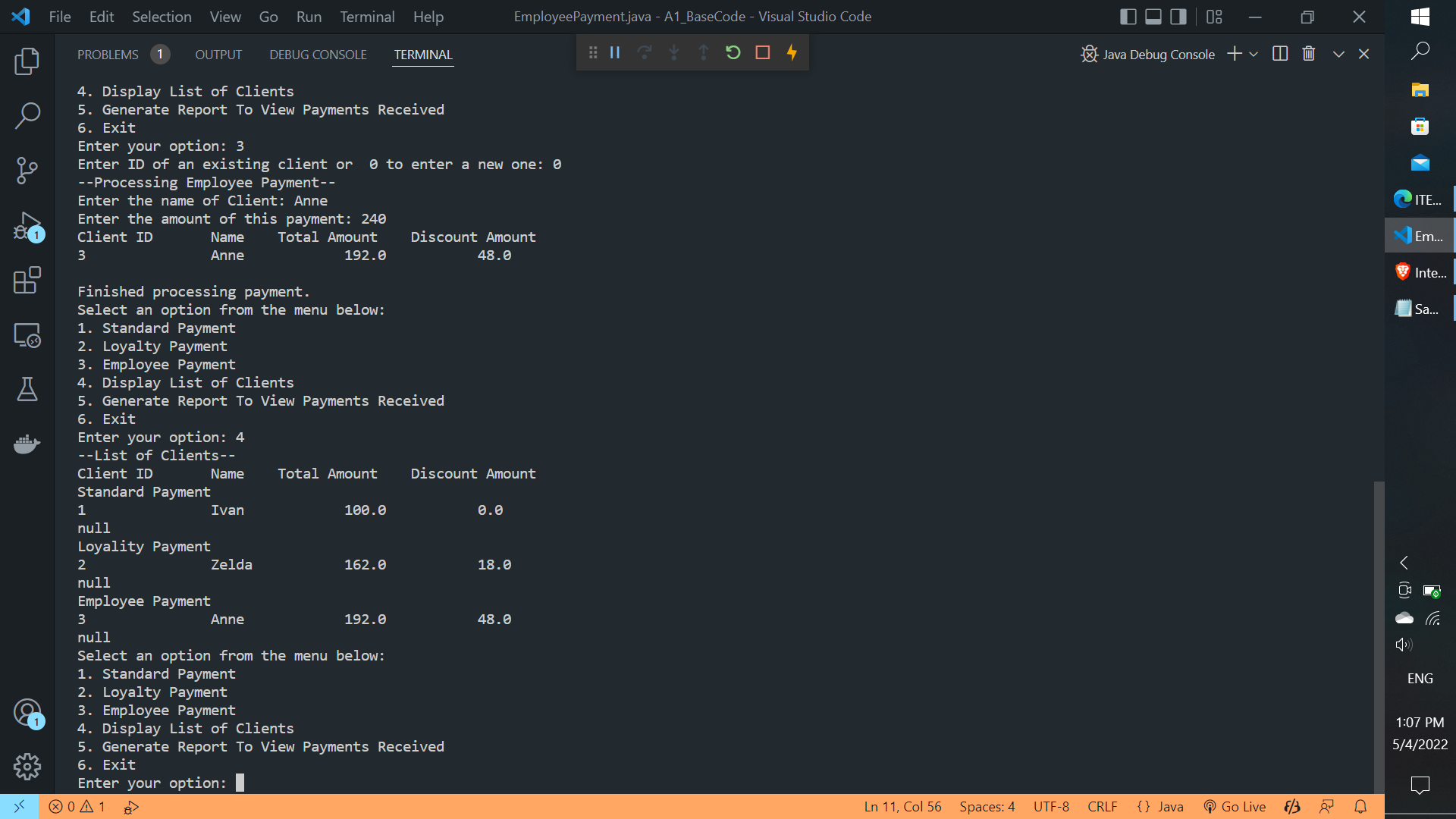
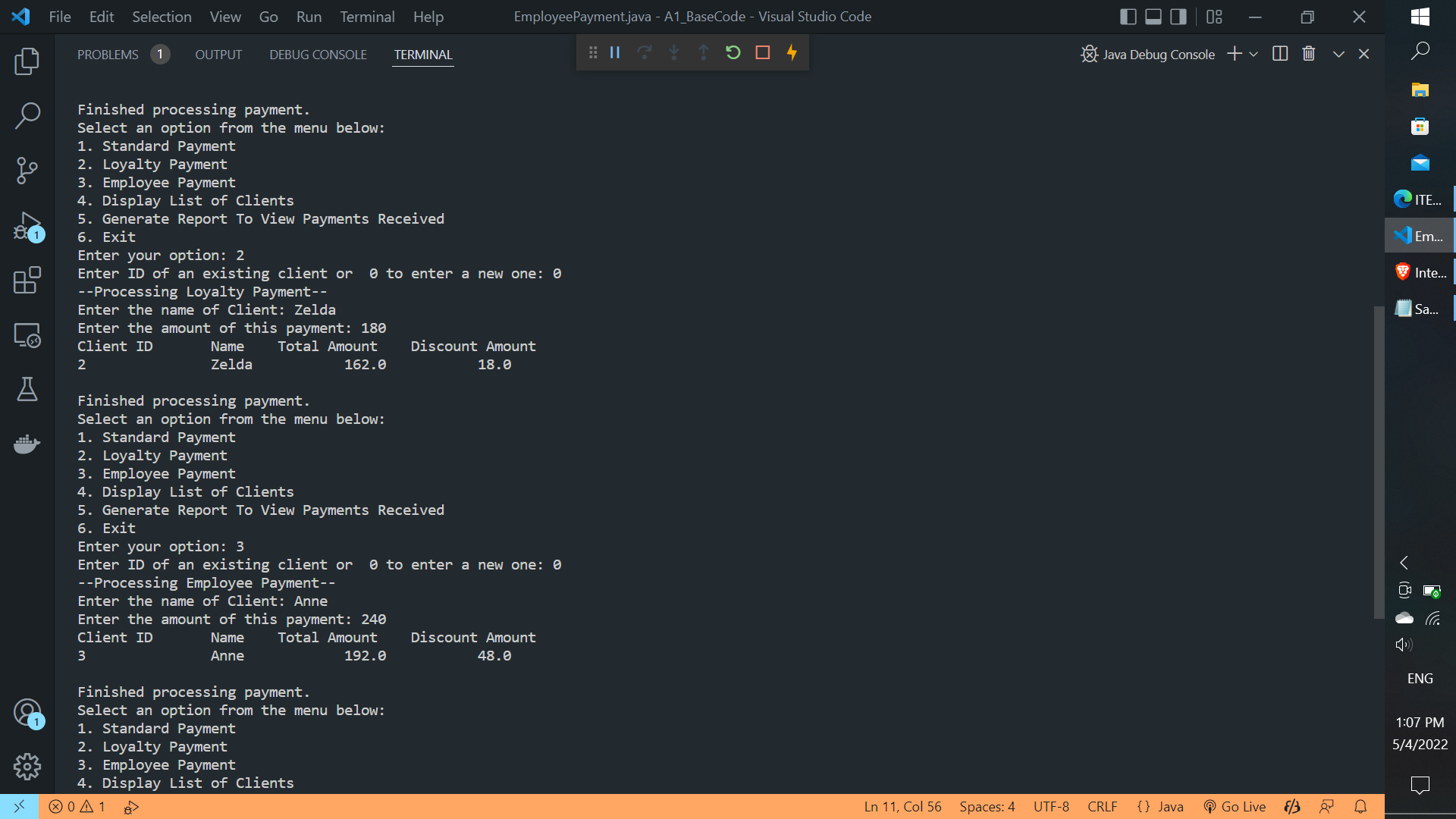
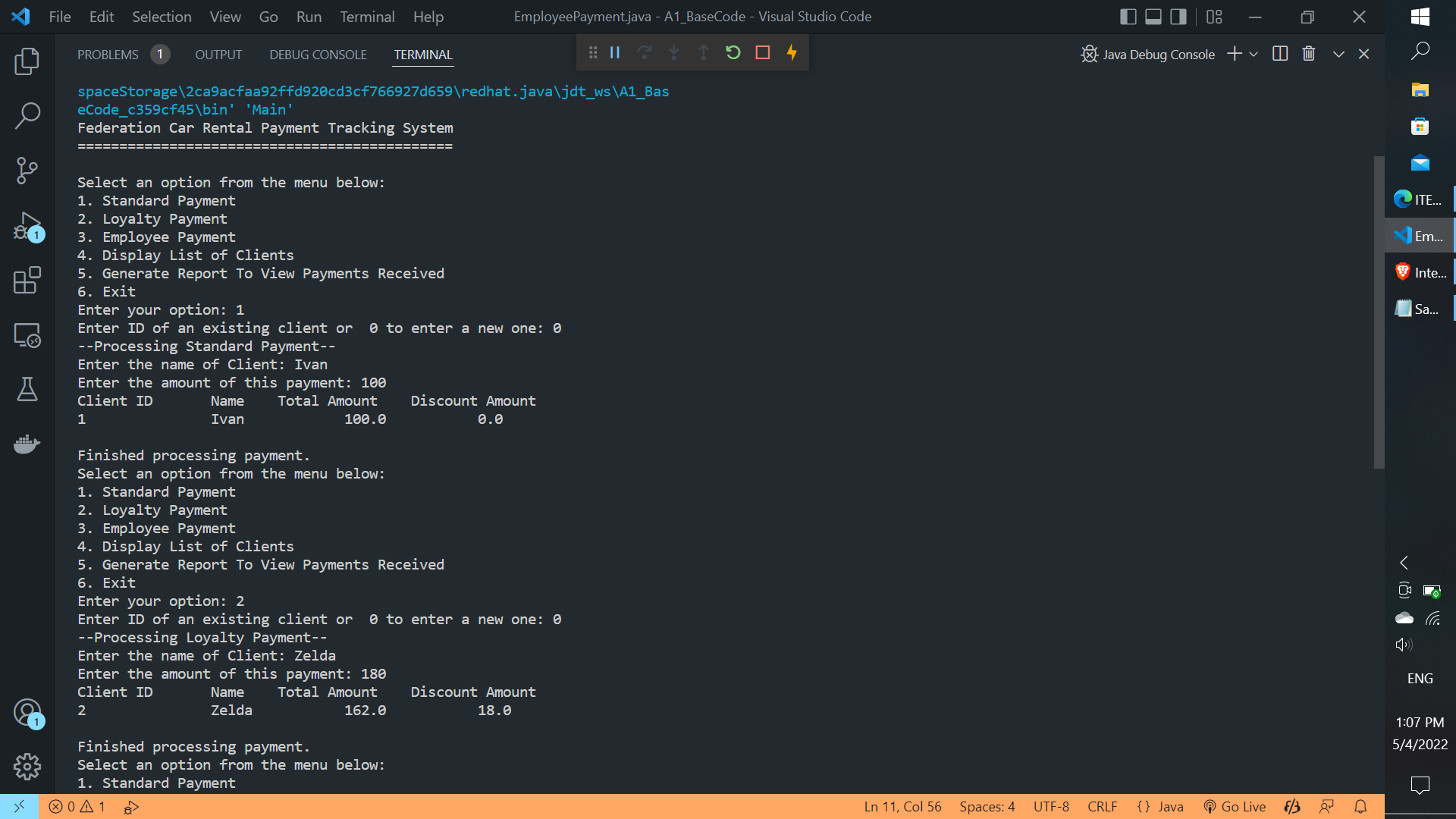
Sequence diagram are responsible for describing the flow of the diagram showing the interactions between the object with the order of interaction of objects along with time.

In FedCR, each menu options are treated as objects. When the program starts, it displays the menu and continues in a loop until the user select to exit the program which is in option 6.

Standard Payment, Loyalty Payment and Employee payment includes hard core coding where user are registered into a unique id and made payment according to the type of payment they selects.

The other two options is responsible for displaying user list and the report form the payment made by the users during the commencement of the whole program.

## Output:



## Source Code

### 6.1 Main.java

public class Main {

    public static void main(String[] args) {

        // Control program execution

        Menu thisMenu = new Menu();

        thisMenu.displayUserMenu();

    }

}

### 6.2 Menu.java

import java.util.Scanner;

public class Menu {

    public int i =0;

    String[] arr = new String[6];

    public void displayUserMenu() {

        // Control the menu navigation.  Includes display of menu, acceptance and processing of user input and

        // exiting the menu based on the user's selections.

        boolean blnContinue = true;

        Scanner scanInput = null;

        header();

        try {       //try-catch-finally to ensure compatibility with all java versions.

            scanInput = new Scanner(System.in);

            while (blnContinue) {

                displayMenu();

                if (scanInput.hasNextInt()) {

                    blnContinue = processMenuSelection(scanInput.nextInt());

                }

                else {

                    scanInput.nextLine();

                    System.out.println("Wrong kind of input");

                    blnContinue = processMenuSelection(0);

                }

            }

        }

        catch (Exception e){

            // Generic error catch - this can be made more specific to the expected errors.

            System.out.println("An unexpected exception has occurred with input");

        }

        finally {

            // Close the scanner

            if (scanInput != null) {

                scanInput.close();

            }

        }

    }

    private void header() {

        // Display program header information

        System.out.println("Federation Car Rental Payment Tracking System");

        System.out.println("=============================================");

        System.out.println();

    }

    private void displayMenu() {

        // Display the menu for user to select which function to run

        System.out.println("Select an option from the menu below:");

        System.out.println("1. Standard Payment");

        System.out.println("2. Loyalty Payment");

        System.out.println("3. Employee Payment");

        System.out.println("4. Display List of Clients");

        System.out.println("5. Generate Report To View Payments Received");

        System.out.println("6. Exit");

        System.out.print("Enter your option: ");

    }

    double[] total = new double[3];

    double[] discount = new double[3];

    int uid = 1;

    int m=0;

    int k = 0;

    int l=0;

    int count1=0, count2=0, count3=0;

    // for displaying object of classes differently

    String[] spay = new String[2];

    String[] lpay = new String[2];

    String[] epay = new String[2];

    private boolean processMenuSelection(int menuSelection) {

        // Use the input provided by the user to activate the appropriate code function

        boolean blnContinue = true;

        Scanner sc = new Scanner(System.in);

        switch (menuSelection){

        case 1:

            // call method to process standard payments here

            StandardPayment sp = new StandardPayment();

            System.out.print("Enter ID of an existing client or  0 to enter a new one: ");

            int sid = sc.nextInt();

            again:

            if(sid == 0)

            {

                System.out.println("--Processing Standard Payment--");

                sp.read();

                sp.calc(uid);

                count1++;

                arr[i] = sp.toString();

                // for displaying StandarPaymetn objects separately

                spay[m] = sp.toString();

                // for payment review

                double tot = sp.getPay();

                total[0] += tot;

                double dis = sp.getDiscount();

                discount[0] += dis;

                // displaying the ojects of the Standard Payment class

                System.out.println("Client ID" + "\t" + "Name" + "\t" + "Total Amount"+ "\t" + "Discount Amount");

                for(int j=0; j < count1; j++)

                {

                    System.out.println(spay[j]);

                }

                System.out.println("\nFinished processing payment.");

                uid++;

                m++;

                i++;

            }

        //  else if(sid == sp.getId() )

            else

            {

                System.out.println("Wrong kind of Input.");

                break again;

            }

            break;

        case 2:

            // call method to process loyalty rewards payments here

            LoyalityPayment lp = new LoyalityPayment();

            System.out.print("Enter ID of an existing client or  0 to enter a new one: ");

            int lid = sc.nextInt();

            again:

            if(lid == 0)

            {

                System.out.println("--Processing Loyalty Payment--");

                lp.read();

                lp.calc(uid);

                count2++;

            //  sp.read();

                arr[i] = lp.toString();

                // to display array of loyality payment only

                lpay[k] = lp.toString();

                // to review the payment received

                double tot = lp.getPay();

                total[1] =total[1] + tot;

                double dis = lp.getDiscount();

                discount[1] = discount[1] + dis;

                System.out.println("Client ID" + "\t" + "Name" + "\t" + "Total Amount"+ "\t" + "Discount Amount");

                for(int j=0; j < count2; j++)

                {

                    System.out.println(lpay[j]);

                }

                System.out.println("\nFinished processing payment.");

                uid++;

                i++;

                k++;

            }

            else

            {

                System.out.println("Wrong kind of Input.");

                break again;

            }

            break;

        case 3:

            // call method to process employee payments here

            EmployeePayment ep= new EmployeePayment();

            System.out.print("Enter ID of an existing client or  0 to enter a new one: ");

            int eid = sc.nextInt();

            again:

            if(eid == 0)

            {

                System.out.println("--Processing Employee Payment--");

                ep.read();

                ep.calc(uid);

                count3++;

                arr[i] = ep.toString();

                // to display employee payments only

                epay[l] = ep.toString();

                // to review the payment and discount received

                double tot = ep.getPay();

                total[2] += tot;

                double dis = ep.getDiscount();

                discount[2] += dis;

                System.out.println("Client ID" + "\t" + "Name" + "\t" + "Total Amount"+ "\t" + "Discount Amount");

                for(int j=0; j < count3; j++)

                {

                    System.out.println(epay[j]);

                }

                System.out.println("\nFinished processing payment.");

                uid++;

                i++;

                l++;

            }

            else

            {

                System.out.println("Wrong kind of Input.");

                break again;

            }

            break;

        case 4:

            // call method to  display list of clients here

            System.out.println("--List of Clients--");

            System.out.println("Client ID" + "\t" + "Name" + "\t" + "Total Amount"+ "\t" + "Discount Amount");

            System.out.println("Standard Payment");

            for(int x = 0; x < spay.length; x++)

            {

                System.out.println(spay[x]);

            }

            System.out.println("Loyality Payment");

            for(int x = 0; x < lpay.length; x++)

            {

                System.out.println(lpay[x]);

            }

            System.out.println("Employee Payment");

            for(int x = 0; x < spay.length; x++)

            {

                System.out.println(epay[x]);

            }

            break;

        case 5:

            // call method to generate report to display payments received here

            System.out.println("--Report to View Payments Received--");

            System.out.println("Type of Account" + "\t" +"Total amount Received" + "\t" + "Discount Given");

            System.out.println("Standard Payment" + "\t\t" + total[0] + "\t\t" + discount[0]);

            System.out.println("Loyality Payment" + "\t\t" + total[1] + "\t\t" + discount[1]);

            System.out.println("Employee Payment" + "\t\t" + total[2] + "\t\t" + discount[2]);

            double sum =0;

            double dis =0;

            for(int x =0; x<3; x++)

            {

                sum = sum + total[x];

                dis = dis + discount[x];

            }

            System.out.println("Total: " + "\t\t\t\t" + sum + "\t\t"+ dis );

            break;

        case 6:

            System.out.println("-- Exiting Federation Car Rental Payment Tracking System --\n ....\n  -- Goodbye! --\n");

            blnContinue = false;

            break;

        default:

            // no valid selection made

            System.out.println("Invalid selection! A number between 1 and 6 was expected.");

        }

        return blnContinue;

    }

}//public class Menu

### 6.3 Payment.java

 interface Payment {

    public void read();

    public void calc(int id);

    public String toString();

}

### StandardPayment.java

import java.util.Scanner;

public class StandardPayment implements Payment{

String name;

int id;

double pay, discount = 0;

Scanner sc = new Scanner(System.in);

public void read(){

System.out.print("Enter the name of Client: ");

name = sc.nextLine();

System.out.print("Enter the amount of this payment: ");

pay = sc.nextInt();

}

public void calc(int id)

{

this.id = id;

}

public int getId()

{

return id;

}

public String getName()

{

return name;

}

public double getPay()

{

return pay;

}

public double getDiscount()

{

return discount;

}

@Override

public String toString(){

return id + "\t\t" +name + "\t\t" + pay + "\t\t" + discount;

}

}

### 6.4 LoyaltyPayment.java

import java.util.Scanner;

public class LoyalityPayment implements Payment{

String name;

int id;

double pay, discount = 0;

Scanner sc = new Scanner(System.in);

public void read(){

System.out.print("Enter the name of Client: ");

name = sc.nextLine();

System.out.print("Enter the amount of this payment: ");

pay = sc.nextInt();

}

public void calc(int id){

discount = pay \* 0.10;

pay = pay - discount;

this.id = id;

}

public String getName()

{

return name;

}

public double getPay()

{

return pay;

}

public double getDiscount()

{

return discount;

}

@Override

public String toString(){

return id + "\t\t" + name + "\t\t" + pay + "\t\t" + discount;

}

}

### 6.5 EmployeePayment.java

import java.util.Scanner;

public class EmployeePayment implements Payment{

String name;

int id;

double pay, discount = 0;

Scanner sc = new Scanner(System.in);

public void read(){

System.out.print("Enter the name of Client: ");

name = sc.nextLine();

System.out.print("Enter the amount of this payment: ");

pay = sc.nextInt();

}

public void calc(int id){

discount = pay \* 0.2;

pay = pay - discount;

this.id = id;

}

public String getName()

{

return name;

}

public double getPay()

{

return pay;

}

public double getDiscount()

{

return discount;

}

@Override

public String toString(){

return id + "\t\t" +name + "\t\t" + pay + "\t\t" + discount;

}

}

## Knowledge Gained

Throughout the project work, I was capable of handling designing and analyzing system requirements through the help of UML diagrams.

Use case diagram helped me in designing functionality of the system through the aspect of actors as users and admin. The activities are the requirement for the system to overcome the actor behaviors.

Class diagram to show the relationship between the objects, its attributes and methods within the class scope as well as outside of the class.

Sequence diagram to show the behavior of the flow of the program including from the start of the program to the end of the program.

Along with the designing techniques, the concept of polymorphism and interface has been clear throughout this assignment. I was able to understand multiple inheritance concept thorough the use of interface which was class name followed by ‘implements’ to interface name. During implementation methods overriding takes places to child class declaring data abstraction.