UNIVERSITI TEKNOLOGI MARA

FUNDAMENTALS OF DATA STRUCTURES (CSC248)

FINAL GROUP PROJECT

No.	Name	Student ID	Mark
1	Syahril Rumizam Bin Abdul Razak	2020843956	
2	Muhamad Adib Asyraaf Bin Azis	2020868324	
3	Muhammad Khairul Haziq Bin Mohamad Khairi	2020475884	
4	Muhammad Harith Iqbal Bin Mohd Hanizun	2020450636	
Group: KCS1103D **Project Title:** Genting SkyWorlds Theme Park's Entrance Ticket Processing **Project Link:** https://github.com/Kyziq/CSC248-Project			

Lecturer Name: Dr. Taniza Tajuddin

Table of Content

Project Summary

Title: Genting SkyWorlds Theme Park's Entrance Ticket Processing

Project Description:

Genting SkyWorlds Theme Park offers great discount for entrance ticket during the school holiday. They provide two types of tickets, namely normal and express tickets. The advantage of purchasing an express ticket is that visitors will not have to wait in the same line as normal queue but instead queueing in their exclusive lane to experience all the theme park's attractions. However, the express ticket is somewhat more expensive than the regular one. The following are the price and discount given for normal and express ticket.

Ticket Type	Age	Normal Price (RM)
Normal (N)	Children(C)	45
	Adult(A)	55
Express (E)	Children(C)	70
	Adult(A)	80

Diagram 1.0 show the table for ticket prices.

If the total fee exceeds RM200, a 17% discount will be applied to the total fee. Otherwise, a 10% discount will be applied to the total fee because of school holiday.

The data from user will be read into a LinkedList to decide which discount will be given to the visitor. The software will process data in the Queue, where all visitors' information will be saved in different sequential lists. The following information will be displayed at the end of the programme:

a. The name of customers and their identity number.
b. The quantity of ticket bought by customer, listed according to type of ticket.
c. The date of the purchase and the date of the ticket purchased by customer.
d. The total amount to be paid after discount.

Show insert, modify, delete, and display data.

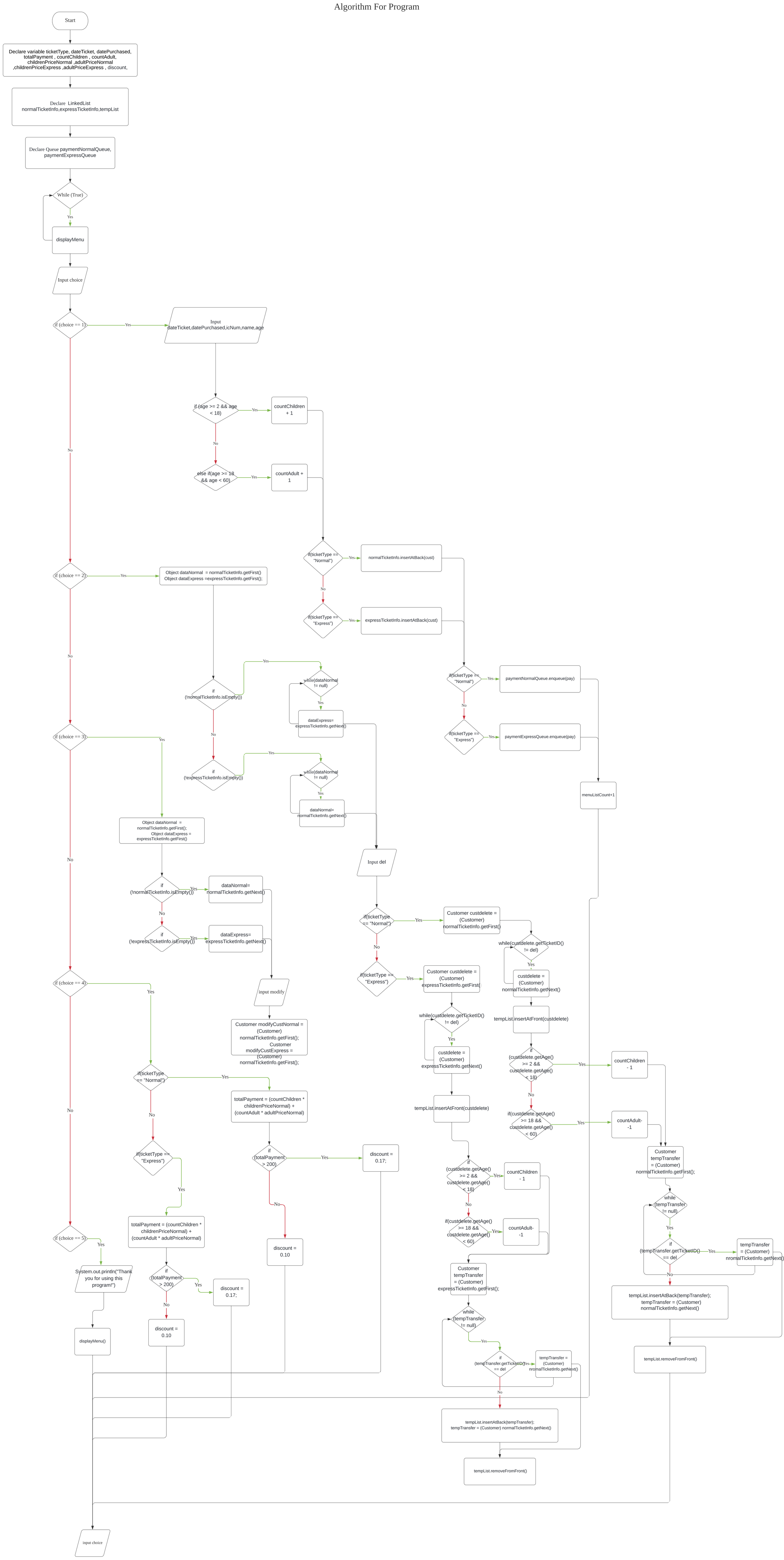
Problem Statement & Requirement

The client required a ticket system to purchase and calculate the amount of payment based on the amount of ticket that client purchase, the type of ticket, client's age and discount that were offered to them. The program needs to read every information that program user provided and display the total payment that they need to pay.

The program that needed to be created should be able to store the data into the program in case the user wants to modify any data. This requires the usage of linked list in the program that need to run for Genting SkyWorld Theme Park. Clients need the payment according to who book the ticket first hence the needs of using Queue data structure.

The requirements for this program to run in Java would be linked list data structure, queue data structure, calculation for payment, calculation for ticket according to user's age and display receipt for user.

Algorithm For Program



Data Structure for Program

Firstly, the abstract data type that has been used for this program is Linked List. Linked list have a few advantages such as they are dynamic data structure which means that they are resizable during run time of a program. In this case, user can input as many tickets as they can and can access any of the data by linking them again. This also ease the process of inserting ticket and deleting ticket when user use the program. Even though, it uses more memory than arraylist or array, it is useful as its reference to other node can be handy in this program. When user input their personal data into the program, the data will be stored into the linked list. The link list will be referred as node. Each node will contain all data of one user. When user wants to delete a ticket, they basically are deleting a single node. When user delete a node, the node will shift from behind to front. User can also change their data inside the node by accessing them using ticket id. This enables them to change their user error such as name, identification number or age. Changing the data means the user access their node and set a new data into the node.

Secondly, the data structure that has been used is Queue. The advantages that queue have among other data structures are adding the data into the back and delete data in the beginning queue (First In First Out). This is useful to keep track on transaction made when user purchasing ticket. The first data that user input will be placed last so when Genting SkyWorld wants to see which transaction was made first, then the queue will show which data was insert first in order. The data is display when user wants to display the final cost that they need to pay for all the tickets they purchased.

Program Design

In designing our program which is for Genting SkyWorlds Theme Park's Entrance Ticket Processing, being a user-friendly program is our top priority. What user-friendly program means to us is that the customer app is intuitive, easy to use, simple and that the customer can rely on the product. How is that even possible?

First of all, we always make sure that this program has a pleasant and easy-to-navigate Graphical User Interface (GUI). Good User Interface Design can make a product easy to understand and use, which results in greater user acceptance. For example, in the main menu below:

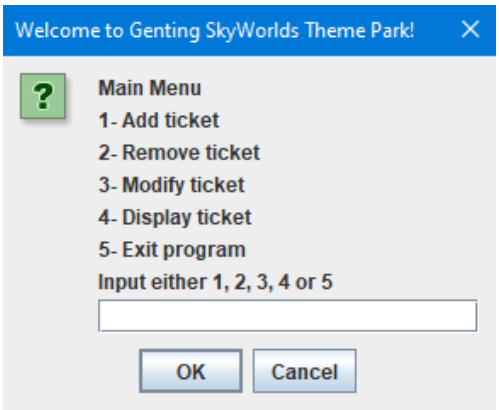


Figure shows main menu 1.

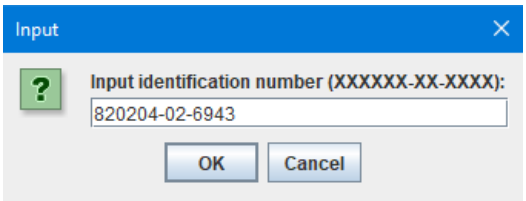


Figure shows adding ticket data

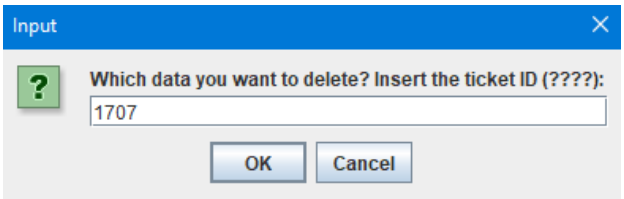
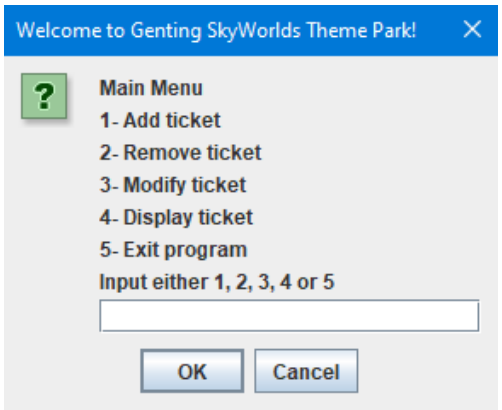


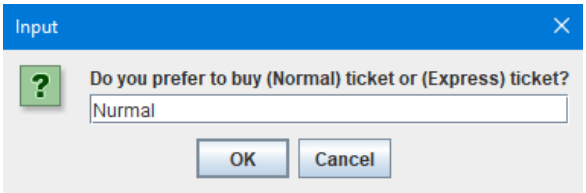
Figure shows user want to delete the ticket

- The interface is simple.
- Consistency and common UI elements.
- User can choose which option to choose easily.

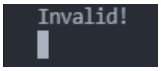
But what if a piece of the program encounters an error? Does it just go away without warning? Does it try to rectify the issue? Additionally, in making this program, we also make sure that if there are any errors found during user input, the program will manage it effectively. With this at least, when a program runs into an error, users will not left with their eyes bugged out and their hands in the air. To illustrate here,



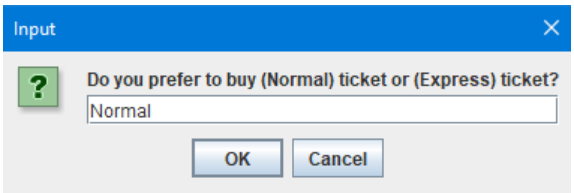
we want to add a ticket to buy, we input number 1.



And then the dialog above will show if we type the input other than normal or express (does not count uppercase and lowercase). It will shows an error and prompt to insert the input again.

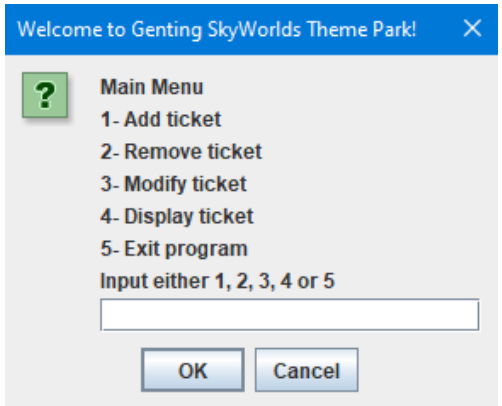


Error in the VSCode Terminal.



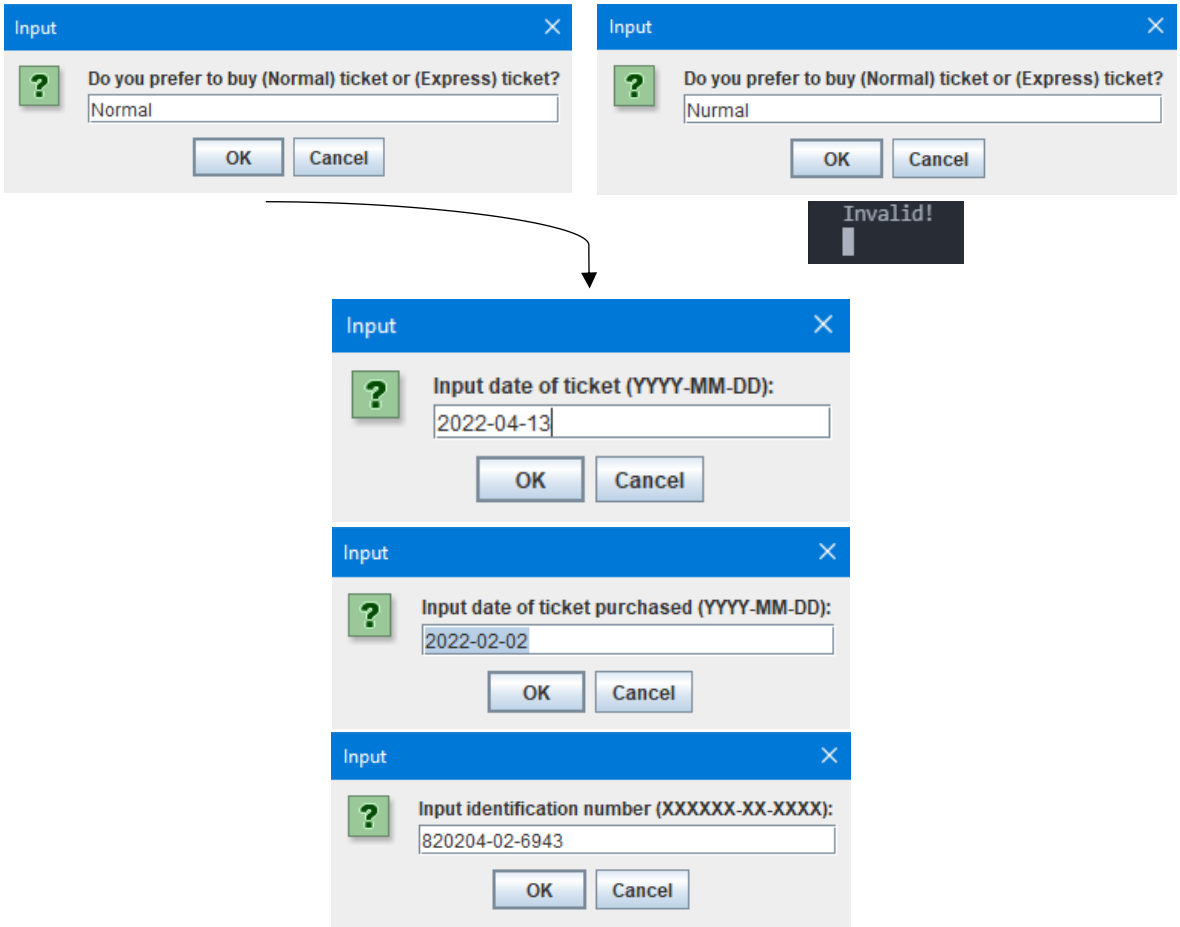
But if we input the word correctly, we input it as Normal. It will then run successfully and continues.

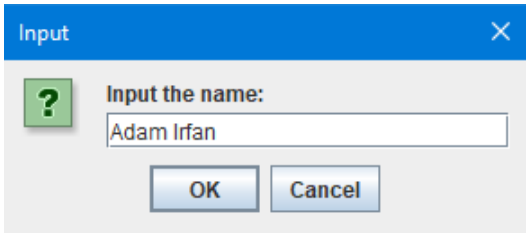
Test Program

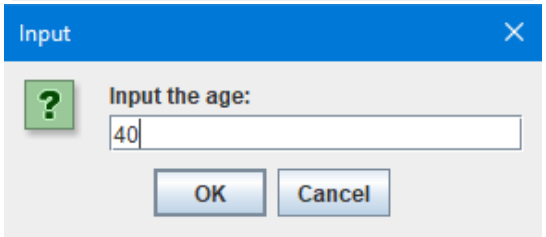


Main Menu 1

And then we insert number 1.







Input:

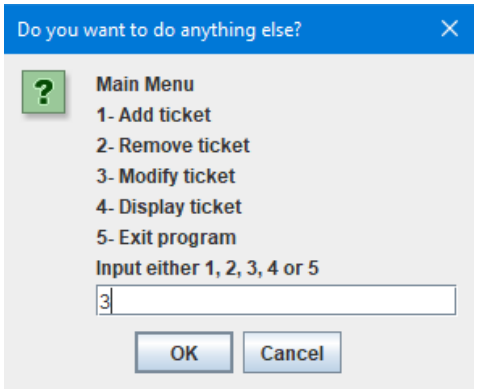
820204-02-6943 Adam Irfan 40

And then input another three family members:

810405-02-4954 Alyaa Haziqah 41

060312-02-5993 Aysar Amzar 16

080518-02-9422 Adibah Asyhura 14



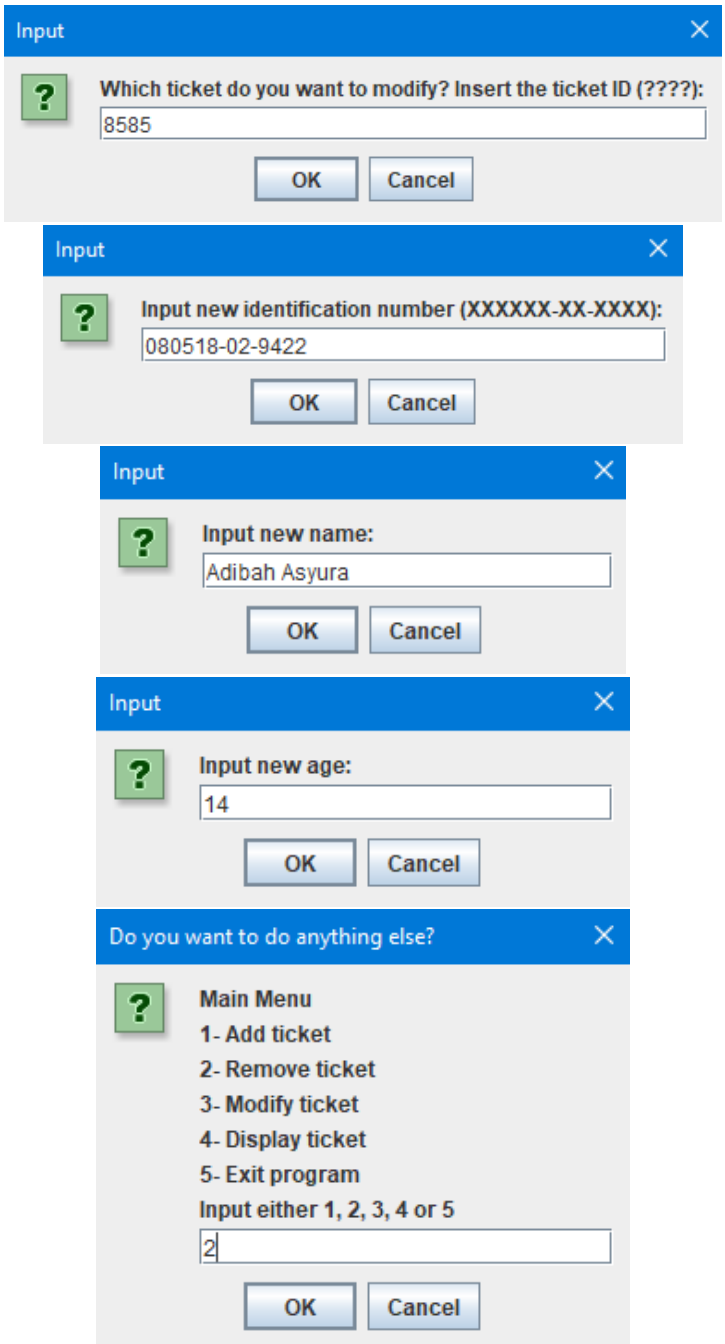
Main Menu 2

And then we insert number 3 to modify ticket. It will output the ticket list first to choose which ticket to modify.

```
Ticket ID: 1846 | IC Number: 820204-02-6943      | Name: ADAM IRFAN          | Age: 40
Ticket ID: 1707 | IC Number: 810405-02-4954      | Name: ALYAA HAZIQAH       | Age: 41
Ticket ID: 1982 | IC Number: 060312-02-5993      | Name: AYSAR AMZAR         | Age: 16
Ticket ID: 8585 | IC Number: 080518-02-9422      | Name: ADIBAH ASYHURA      | Age: 14
```

From VSCode Terminal

For example, we want to modify the ticket for Adibah Asyhura (Ticket ID 8585) and change her name to Adibah Asyura because of human error.

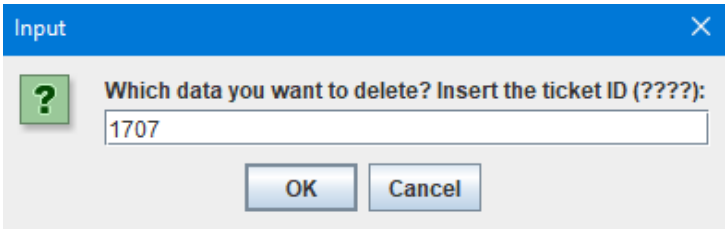


Main Menu 2

And then we insert number 2 to remove ticket. It will output the ticket list first to show the ticket ID to remove.

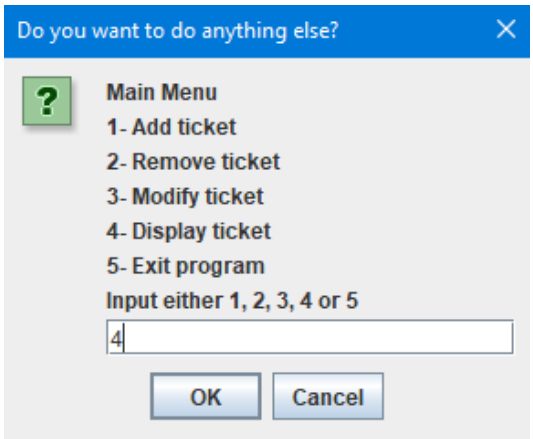
```
Ticket ID: 1846 | IC Number: 820204-02-6943       | Name: ADAM IRFAN          | Age: 40
Ticket ID: 1707 | IC Number: 810405-02-4954       | Name: ALYAA HAZIQAH       | Age: 41
Ticket ID: 1982 | IC Number: 060312-02-5993       | Name: AYSAR AMZAR         | Age: 16
Ticket ID: 8585 | IC Number: 080518-02-9422       | Name: ADIBAH ASYURA       | Age: 14
```

From VSCode Terminal



For example, we want to delete the ticket for Alyaa Haziqah (Ticket ID 1707) because of change of plan.

And then we insert number 2 to remove ticket.



Main Menu 2

And then we insert number 4 to display all the ticket data.

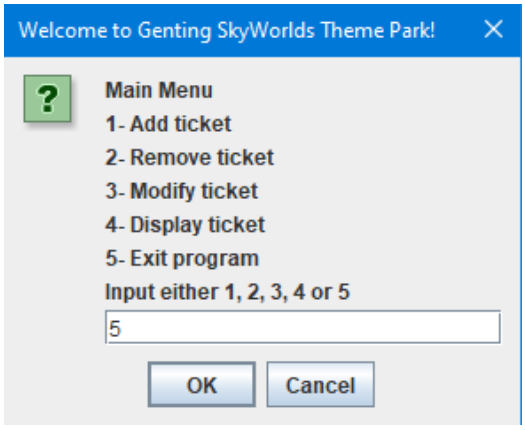
```
-----------------------------------------------------------------------------------------------------
                        Genting SkyWorlds Theme Park Ticket List
-----------------------------------------------------------------------------------------------------
Ticket ID: 1846 | IC Number: 820204-02-6943       | Name: ADAM IRFAN           | Age: 40
Ticket ID: 1982 | IC Number: 060312-02-5993       | Name: AYSAR AMZAR          | Age: 16
Ticket ID: 8585 | IC Number: 080518-02-9422       | Name: ADIBAH ASYURA        | Age: 14

-----------------------------------------------------------------------------------------------------

                                       Payment Info
-----------------------------------------------------------------------------------------------------
Total Payment: RM130.50 | Ticket Type: NORMAL | Date Ticket: 2022-04-13 | Date Purchased: 2022-02-02
-----------------------------------------------------------------------------------------------------

Thank you for using this program!
```

From VSCode Terminal

It will then output all the ticket list along with the payment result. The program then ends here.

In case user wants to end the program early, which the

choice is number 5 to exit program instantly.



Main Menu 1

And then we insert number 5 to exit the program

```
Thank you for using this program!
```

And the program ends here.

Documentation

mainApp.java

```
import java.util.Scanner;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class mainApp {
    // Global Variable
    static int menuListCount=1;
    static String menuChoice2; static int menuChoice;
    // Input
    static JFrame f = new JFrame();
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args)
    {
        // Variable
        String ticketType="";
        String dateTicket="";
        String datePurchased="";
        double totalPayment=0.00;

        // Count age list
        int countChildren=0, countAdult=0;

        // Price Tickets
        final int childrenPriceNormal = 45;
        final int adultPriceNormal = 55;
        final int childrenPriceExpress = 70;
        final int adultPriceExpress = 80;

        // Discount
        double discount;

        // Linked Lists
        LinkedList normalTicketInfo = new LinkedList();
        LinkedList expressTicketInfo = new LinkedList();
        LinkedList tempList = new LinkedList();

        // Queue
        Queue paymentNormalQueue = new Queue();
        Queue paymentExpressQueue = new Queue();
```

```
        while(true) {
            displayMenu(); // Intro display, menu 1

            if (menuChoice == 1) // Add ticket
            {
                if (ticketType.isEmpty()) {
                    do {
                        ticketType = JOptionPane.showInputDialog(f,"Do you prefer to buy (Normal) ticket
or (Express) ticket?");

                        if (!ticketType.equalsIgnoreCase("Normal") &&
!ticketType.equalsIgnoreCase("Express")) {
                            System.out.println("Invalid!");
                        }
                    }
                    while (!ticketType.equalsIgnoreCase("Normal") &&
!ticketType.equalsIgnoreCase("Express"));
                }

                if (dateTicket.isEmpty()) {
                    dateTicket = JOptionPane.showInputDialog(f,"Input date of ticket (YYYY-MM-DD):
");
                }

                if (datePurchased.isEmpty()) {
                    datePurchased = JOptionPane.showInputDialog(f,"Input date of ticket purchased (YYYY-
MM-DD): ");
                }

                String icNum = JOptionPane.showInputDialog(f,"Input identification number (XXXXXX-XX-
XXXX): ");

                String name = JOptionPane.showInputDialog(f,"Input the name: ");

                String age2 = JOptionPane.showInputDialog(f,"Input the age: ");
                int age = Integer.parseInt(age2);

                int ticketID = GenerateRandom();

                // Add headcount for easier calculation
                if (age >= 2 && age < 18) {
                    countChildren++;
                }
                else if(age >= 18 && age < 60) {
                    countAdult++;
```

```
            }

            // LinkedList data structure
            Customer cust = new Customer (icNum, name, age, ticketID);
            if(ticketType.equalsIgnoreCase("Normal")) {
                normalTicketInfo.insertAtBack(cust);
            }
            else if(ticketType.equalsIgnoreCase("Express")) {
                expressTicketInfo.insertAtBack(cust);
            }

            // Queue data structure
            Payment pay = new Payment (totalPayment, ticketType, dateTicket, datePurchased);
            if(ticketType.equalsIgnoreCase("Normal")) {
                paymentNormalQueue.enqueue(pay);
            }
            else if(ticketType.equalsIgnoreCase("Express")) {
                paymentExpressQueue.enqueue(pay);
            }
            menuListCount++;
            continue; // Continue Loop
        }
        else if (menuChoice == 2) // Remove
        {
            Customer Cus = null;
            Object dataNormal  = normalTicketInfo.getFirst();
            Object dataExpress = expressTicketInfo.getFirst();

            // Show the ticket data to remove which ticketID
            if (!normalTicketInfo.isEmpty()) // For normal tickets
            {
                while (dataNormal != null)
                {
                    Cus = (Customer) dataNormal;
                    Cus.CustomerPrint();
                    dataNormal= normalTicketInfo.getNext();
                }
            }
            else if (!expressTicketInfo.isEmpty()) // For express tickets
            {
                while (dataExpress != null)
                {
                    Cus = (Customer) dataExpress;
                    Cus.CustomerPrint();
                    dataExpress= expressTicketInfo.getNext();
```

```
                    }
                }
                String del2 = JOptionPane.showInputDialog(f,"Which data you want to delete? Insert the
ticket ID (????): ");
                int del = Integer.parseInt(del2);

                // Delete stuff (linked list)
                if (ticketType.equalsIgnoreCase("Normal")) // For normal tickets
                {
                    Customer custdelete = (Customer) normalTicketInfo.getFirst();

                    // USER WANT (Insert linked list front)
                    while(custdelete.getTicketID() != del)
                        custdelete = (Customer) normalTicketInfo.getNext();
                    tempList.insertAtFront(custdelete);

                    // Delete the count head (calculation) for USER WANT delete
                    if (custdelete.getAge() >= 2 && custdelete.getAge() < 18)
                        countChildren--;
                    else if(custdelete.getAge() >= 18 && custdelete.getAge() < 60)
                        countAdult--;

                    // Other than USER WANT (After front, 2nd, 3rd, ...)
                    Customer tempTransfer = (Customer) normalTicketInfo.getFirst();
                    while (tempTransfer != null)  // To transfer elements from original linkedList to
temporary linkedList
                    {
                        if (tempTransfer.getTicketID() == del) // Except USER WANT
                            tempTransfer = (Customer) normalTicketInfo.getNext();
                        else
                        {
                            tempList.insertAtBack(tempTransfer);
                            tempTransfer = (Customer) normalTicketInfo.getNext();
                        }
                    }
                    tempList.removeFromFront(); // Remove USER WANT

                    // To clear original linked list
                    normalTicketInfo.getFirst();
                    while(!normalTicketInfo.isEmpty())
                        normalTicketInfo.removeFromFront();

                    // To transfer from temporary to original
                    Customer custprint = (Customer) tempList.getFirst();
                    while(custprint != null)
```

```
                {
                    normalTicketInfo.insertAtBack(custprint);
                    custprint = (Customer) tempList.getNext();
                }

                // Clear temporary linked list
                while(!tempList.isEmpty())
                    tempList.removeFromFront();
            }
            else if (ticketType.equalsIgnoreCase("Express")) // For express tickets
            {
                Customer custdelete = (Customer) expressTicketInfo.getFirst();

                // USER WANT (Insert linked list front)
                while(custdelete.getTicketID() != del)
                    custdelete = (Customer) expressTicketInfo.getNext();
                tempList.insertAtFront(custdelete);

                // Delete the count head (calculation) for USER WANT delete
                if (custdelete.getAge() >= 2 && custdelete.getAge() < 18)
                    countChildren--;
                else if(custdelete.getAge() >= 18 && custdelete.getAge() < 60)
                    countAdult--;

                // Other than USER WANT (After front, 2nd, 3rd, ...)
                Customer tempTransfer = (Customer) expressTicketInfo.getFirst();
                while (tempTransfer != null)  // To transfer elements from original linkedList to
temporary linkedList
                {
                    if (tempTransfer.getTicketID() == del) // Except USER WANT
                        tempTransfer = (Customer) expressTicketInfo.getNext();
                    else
                    {
                        tempList.insertAtBack(tempTransfer);
                        tempTransfer = (Customer) expressTicketInfo.getNext();
                    }
                }
                tempList.removeFromFront(); // Remove USER WANT

                // To clear original linked list
                expressTicketInfo.getFirst();
                while(!expressTicketInfo.isEmpty())
                expressTicketInfo.removeFromFront();

                // To transfer from temporary to original
```

```
                    Customer custprint = (Customer) tempList.getFirst();
                    while(custprint != null)
                    {
                        expressTicketInfo.insertAtBack(custprint);
                        custprint = (Customer) tempList.getNext();
                    }

                    // Clear temporary linked list
                    while(!tempList.isEmpty())
                        tempList.removeFromFront();
                }
                continue;
            }
            else if (menuChoice == 3) // Modify datas
            {
                // Modify data
                Customer Cus = null;
                Object dataNormal  = normalTicketInfo.getFirst();
                Object dataExpress = expressTicketInfo.getFirst();

                // Show the datas
                if (!normalTicketInfo.isEmpty()) // For normal tickets
                {
                    while (dataNormal != null)
                    {
                        Cus = (Customer) dataNormal;
                        Cus.CustomerPrint();
                        dataNormal= normalTicketInfo.getNext();
                    }
                }
                else if (!expressTicketInfo.isEmpty()) // For express tickets
                {
                    while (dataExpress != null)
                    {
                        Cus = (Customer) dataExpress;
                        Cus.CustomerPrint();
                        dataExpress= expressTicketInfo.getNext();
                    }
                }

                // Input
                String modify2 = JOptionPane.showInputDialog(f,"Which ticket do you want to modify? Insert
the ticket ID (????): ");
                int modify = Integer.parseInt(modify2);
```

```
                Customer modifyCustNormal = (Customer) normalTicketInfo.getFirst();
                Customer modifyCustExpress = (Customer) normalTicketInfo.getFirst();

                // Edit normal ticket type
                if (ticketType.equalsIgnoreCase("Normal"))
                {
                    while(modifyCustNormal.getTicketID() != modify)
                        modifyCustNormal = (Customer) normalTicketInfo.getNext();

                    // Delete the count head (calculation) before
                    if (modifyCustNormal.getAge() >= 2 && modifyCustNormal.getAge() < 18)
                        countChildren--;
                    else if(modifyCustNormal.getAge() >= 18 && modifyCustNormal.getAge() < 60)
                        countAdult--;

                    String icNum= JOptionPane.showInputDialog(f,"Input new identification number (XXXXXX-
XX-XXXX): ");
                    modifyCustNormal.setIC(icNum);

                    String name= JOptionPane.showInputDialog(f,"Input new name: ");
                    modifyCustNormal.setName(name);

                    String age2= JOptionPane.showInputDialog(f,"Input new age: ");
                    int age = Integer.parseInt(age2);
                    modifyCustNormal.setAge(age);

                    // Add the count head (calculation) after modified
                    if (modifyCustNormal.getAge() >= 2 && modifyCustNormal.getAge() < 18)
                        countChildren++;
                    else if(modifyCustNormal.getAge() >= 18 && modifyCustNormal.getAge() < 60)
                        countAdult++;
                }
                // Edit express ticket type
                else if(ticketType.equalsIgnoreCase("Express"))
                {
                    while(modifyCustExpress.getTicketID() != modify)
                        modifyCustExpress = (Customer) expressTicketInfo.getNext();

                    // Delete the count head (calculation) before
                    if (modifyCustExpress.getAge() >= 2 && modifyCustExpress.getAge() < 18)
                        countChildren--;
                    else if(modifyCustExpress.getAge() >= 18 && modifyCustExpress.getAge() < 60)
                        countAdult--;
```

```
                    String icNum= JOptionPane.showInputDialog(f,"Input new identification number (XXXXXX-
XX-XXXX): ");
                    modifyCustExpress.setIC(icNum);

                    String name= JOptionPane.showInputDialog(f,"Input new name: ");
                    modifyCustExpress.setName(name);

                    String age2= JOptionPane.showInputDialog(f,"Input new age: ");
                    int age = Integer.parseInt(age2);
                    modifyCustExpress.setAge(age);

                    // Add the count head (calculation) after modified
                    if (modifyCustExpress.getAge() >= 2 && modifyCustExpress.getAge() < 18)
                        countChildren++;
                    else if(modifyCustExpress.getAge() >= 18 && modifyCustExpress.getAge() < 60)
                        countAdult++;
                }
                continue; // Continue loop
            }
            else if (menuChoice == 4) // Display
            {
                // Calculate Payment....
                if(ticketType.equalsIgnoreCase("Normal"))
                {
                    // Total heads
                    totalPayment = (countChildren * childrenPriceNormal) + (countAdult *
adultPriceNormal);

                    // Discount
                    if (totalPayment > 200)
                        discount = 0.17;
                    else
                        discount = 0.10;
                    totalPayment = totalPayment * (1 - discount);

                    // Insert data
                    Payment pay = (Payment) paymentNormalQueue.getFront(); // Get normal queue
                    pay.setTotalPayment(totalPayment);
                }
                else if(ticketType.equalsIgnoreCase("Express"))
                {
                    // Total heads
                    totalPayment = (countChildren * childrenPriceExpress) + (countAdult *
adultPriceExpress);
```

```
                // Discount
                if (totalPayment > 200)
                    discount = 0.17;
                else
                    discount = 0.10;
                totalPayment = totalPayment * (1 - discount);

                Payment pay = (Payment) paymentExpressQueue.getFront(); // Get express queue
                pay.setTotalPayment(totalPayment);
            }
            System.out.printf("");
            System.out.println("-----------------------------------------------------------------------
-----------------------------");
            System.out.println("\t\t\tGenting SkyWorlds Theme Park Ticket List");
            System.out.printf("-----------------------------------------------------------------------
----------------------------\n");

            Customer Cus = null;
            Object dataNormal  = normalTicketInfo.getFirst();
            Object dataExpress = expressTicketInfo.getFirst();
            // Show Customer tickets.
            if (!normalTicketInfo.isEmpty()) // For normal tickets
            {
                while (dataNormal != null)
                {
                    Cus = (Customer) dataNormal;
                    Cus.CustomerPrint();
                    dataNormal= normalTicketInfo.getNext();
                }
            }
            else if (!expressTicketInfo.isEmpty()) // For express tickets
            {
                while (dataExpress != null)
                {
                    Cus = (Customer) dataExpress;
                    Cus.CustomerPrint();
                    dataExpress= expressTicketInfo.getNext();
                }
            }

            // Show Payment datas.
            Payment PayN = null;
            Payment PayQ = null;
            Object dataPaymentN = paymentNormalQueue.getFront();
            Object dataPaymentQ = paymentExpressQueue.getFront();
```

```
                System.out.println("");
                System.out.println("------------------------------------------------------------------
-----------------------------");
                System.out.print("\n\t\t\t\t\tPayment Info\n");
                System.out.println("------------------------------------------------------------------
-----------------------------");

                PayN = (Payment) dataPaymentN;
                PayQ = (Payment) dataPaymentQ;

                if (!paymentNormalQueue.isEmpty()) // For normal tickets
                {
                    PayN.PaymentPrint();
                }
                else if (!expressTicketInfo.isEmpty()) // For express tickets
                {
                    PayQ.PaymentPrint();
                }
                System.out.println("------------------------------------------------------------------
-----------------------------\n");
                System.out.println("Thank you for using this program!\n");
                break; // Out from loop
            }
            else if (menuChoice == 5) {
                // Exit program
                System.out.println("Thank you for using this program!\n");
                break; // Out from loop
            }
            displayMenu(); // Continue display
        }
        input.close();
    }



    // functions
    private static void displayMenu() {
        // Menu list for welcoming
        if (menuListCount == 1)
        {
            menuChoice2= JOptionPane.showInputDialog(null,
            "Main Menu\n"+
            "1- Add ticket\n"+
            "2- Remove ticket\n"+
```

```
            "3- Modify ticket\n"+
            "4- Display ticket\n"+
            "5- Exit program\n"+
            "Input either 1, 2, 3, 4 or 5", "Welcome to Genting SkyWorlds Theme Park!",
JOptionPane.QUESTION_MESSAGE);
            menuChoice = Integer.parseInt(menuChoice2);
        }
        else if (menuListCount >= 2)
        {
            // Menu list after welcoming
            menuChoice2= JOptionPane.showInputDialog(null,
            "Main Menu\n"+
            "1- Add ticket\n"+
            "2- Remove ticket\n"+
            "3- Modify ticket\n"+
            "4- Display ticket\n"+
            "5- Exit program\n"+
            "Input either 1, 2, 3, 4 or 5", "Do you want to do anything else?",
JOptionPane.QUESTION_MESSAGE);
            menuChoice = Integer.parseInt(menuChoice2);
        }
    }

    // Random number for ticket ids.
    private static int GenerateRandom () {
        int min = 1000;
        int max = 9999;

        //Generate random int value from 1000 to 9999
        int random = (int)Math.floor(Math.random()*(max-min+1)+min);
        return random;
    }
}
```

Customer.java

```
import java.io.PrintStream;

// Customer Info
public class Customer {
    private String icNum;
    private String name;
    private int age; // Baby - <2 years, Children - 2-17years, Adult - 18-59years, Senior
Adult - 60+years
    private int ticketID;

    // Default Constructor
    public Customer() {
        icNum = "";
        name = "";
        age = -1;
        ticketID = -1;
    }
    // Normal Constructor
    public Customer(String icNum, String name, int age, int ticketID) {
        this.icNum = icNum;
        this.name = name;
        this.age = age;
        this.ticketID = ticketID;
    }
    // Group Setter
    public void setCustomer(String icNum, String name, int age, int ticketID) {
        this.icNum = icNum;
        this.name = name;
        this.age = age;
        this.ticketID = ticketID;
    }
    // Getters
    public String getIC() {
        return this.icNum;
    }
    public String getName() {
        return this.name;
    }
    public int getAge() {
        return this.age;
    }
    public int getTicketID(){
        return this.ticketID;
```

```
    }
    // Setters
    public void setIC(String newICNum) {
        this.icNum = newICNum;
    }
    public void setName(String newName) {
        this.name = newName;
    }
    public void setAge(int newAge) {
        this.age = newAge;
    }
    public void setTicketID(int newTicketID){
        this.ticketID = newTicketID;
    }
    // Printer
    public PrintStream CustomerPrint() {
        return System.out.printf("Ticket ID: %d | IC Number: %-20s | Name: %-20S | Age: %d
%n", ticketID, icNum, name, age);
    }
}
```

Payment.java

```
import java.io.PrintStream;

public class Payment {
    private double totalPayment;
    private String ticketType;
    private String dateTicket;
    private String datePurchased;

    // Default Constructor
    public Payment() {
        totalPayment = -1;
        ticketType = "";
        dateTicket = "";
        datePurchased = "";
    }
    // Normal Constructor
    public Payment(double totalPayment, String ticketType, String dateTicket, String
datePurchased) {
        this.totalPayment = totalPayment;
        this.ticketType = ticketType;
        this.dateTicket = dateTicket;
        this.datePurchased = datePurchased;
    }

    // Group Setter
    public void setPayment(double totalPayment, String ticketType, String dateTicket, String
datePurchased) {
        this.totalPayment = totalPayment;
        this.ticketType = ticketType;
        this.dateTicket = dateTicket;
        this.datePurchased = datePurchased;
    }
    // Getter Constructor
    public double getTotalPayment() {return totalPayment;}
    public String getTicketType () {return ticketType;}
    public String getDateTicket () {return dateTicket;}
    public String getDatePurchased () {return datePurchased;}

    // Setter Constructor
    public void setTotalPayment(double totalPayment) {
        this.totalPayment = totalPayment;
    }
    public void setTicketType(String ticketType) {
```

```
        this.ticketType = ticketType;
    }
    public void setDateTicket(String dateTicket) {
        this.dateTicket = dateTicket;
    }
    public void setDatePurchased(String datePurchased) {
        this.datePurchased = datePurchased;
    }

    // Printer
    public PrintStream PaymentPrint() {
        return System.out.printf("Total Payment: RM%.2f | Ticket Type: %S | Date Ticket: %s |
Date Purchased: %s %n", totalPayment, ticketType, dateTicket, datePurchased);
    }
}
```

Node.java

```
public class Node {

    Object data;
    Node next;

    Node(Object data)
    {
        this.data = data;
    }
}
```

LinkedList.java

```
public class LinkedList
{
    private Node first;
    private Node last;
    private Node current;

    public LinkedList()
    {
        first = null;
        last = null;
        current = null;
    }
    public boolean isEmpty()
    {
        return (first==null);
    }
    public void insertAtFront(Object data)
    {
        Node newNode = new Node(data);
        if (isEmpty())
        {
            first = newNode;
            last = newNode;
        }
        else
        {
            newNode.next = first;
            first = newNode;
        }
    }
    public void insertAtBack(Object data)
    {
        Node newNode = new Node(data);
        if (isEmpty())
        { first = newNode;
        last = newNode;
        }
        else
        {
            last.next = newNode;
            last = newNode;
        }
    }
```

```
public Object removeFromFront()
{
    Object removeItem = null;
    if (isEmpty())
    {
        return removeItem;
    }
    removeItem = first.data;
    if ( first == last)
    {
        first = null;
        last = null;
    }
    else
        first = first.next;
    return removeItem; }

public Object removeFromBack()
{
    Object removeItem = null;
    if (isEmpty())
    {
        return removeItem;
    }
    removeItem = last.data;
    if ( first == last)
    {
        first = null;
        last = null;
    }
    else
    {
        current = first;
        while (current.next != last)
            current = current.next;
        last = current;
        last.next = null;
    }
    return removeItem; }

public Object getFirst()
{
    if (isEmpty())
        return null;
```

```
        else
        {
            current = first;
            return current.data;
        }
    }

    public Object getNext()
    {
        if (current == last)
            return null;
        else
        {
            current = current.next;
            return current.data;
        }
    }

    public void clear() {
        first = current = last = null;
    }

    public Object set(int index, Object e)
    {
        if (index < 0 || index > 0)
            return null;
        current = first;
        for(int i=0; i<index; i++) {
            current = current.next;
        }
        Node temp = current;
        current.data = e;
        return temp;
    }
}
```

Queue.java

```java
public class Queue extends LinkedList {

    public Queue() {}

    public void enqueue(Object data) {
        insertAtBack(data);
    }

    public Object dequeue() {
        return removeFromFront();
    }

    public Object getFront() {
        return getFirst();
    }

    public Object getEnd() {
        Object data = removeFromBack();
        insertAtBack(data);
        return data;
    }
}
```