Cawangan Perlis
Kampus Arau

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES
UNIVERSITI TEKNOLOGI MARA CAWANGAN PERLIS

FUNDAMENTALS OF DATA STRUCTURES
(CSC248)

DIPLOMA IN COMPUTER SCIENCE

SEMESTER MARCH 2023 – AUGUST 2023

GROUP PROJECT: BERJAYA HOTEL MANAGEMENT SYSTEM

PREPARED BY:

GROUP MEMBERS	GROUP MEMBERS	MATRIC NO
	SITI NURALIA NATASHYA BINTI HAMDAN	2022539307
	NUR HANIS NATASYA BINTI GHAZALI	2022791679
	NURAINAA BALQIS BINTI MOHD ADLY HAFEZ HEDAYAT	2022770417
	MADIHAH BINTI MOKHTAR	2022141799
GROUP	RCS1103B	

PREPARED FOR: MADAM HANISAH BINTI AHMAD

SUBMISSION DATE
21st JULY 2023

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Project Background

Berjaya Hotel is a hotel and resort in Langkawi; This organizations plans to organize their guest booking information and develop an application system for the hotel. Below is the information given by Berjaya Hotel related to issues in keeping their monthly booking record:

They have a list of the room available at the hotel and guest details to be recorded. The room to be book is in three categories (S- Superior, P- Premier and T- Premier Suites). They record their room with the following information which are the room ID, room category and breakfast option. Other than that, they will record their guest's information which is booking ID, name, contact number and number of guests. Also, a payment details is amount of payment, payment methods and payment status (D- Paid and N- Unpaid). Each payment transaction will be recorded, and payment status will be updated. After the data has been collected and stored, at the end of the month, they need to process the information and display it in a form of report for their sales automatically.

After the discussion, we analyse their requirements, and these are the processes we propose to Berjaya Hotel to be applied in the application.

The application will be able to:

1. Store the room and guest's information in the application
2. Remove certain records of the guest after the guest meet the end of their stay (date before 07/07/2023).
3. Search and display the guest's details which didn't make the payment yet.
4. Search and display the guest's details for their upcoming booking (date after 07/07/2023).
5. Update the guest's payment status after the guest made booking payment.
6. Split the record for the guest who already paid and the guest that did not make the payment yet.
7. Calculate the total payment made by each customer.
8. Calculate the total of sum payment received from each payment status category.

Object Propose: Guests

Attributes : Guest Name, Guest's Phone Number, Number of Guest, Booking ID, Booking Date, Total Day, Room Number, Room Category (Superior, Premier or Premier Suites), Breakfast option, Payment date, Payment status and Payment method.

Methods : accessor method, mutator method, processor method and display method.

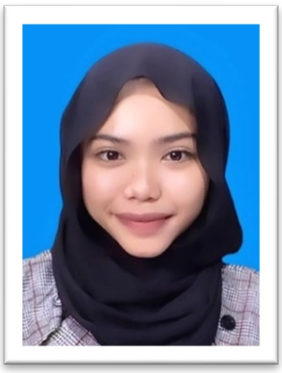
1.2 PROJECT OBJECTIVES

i. To ease company to manage customer information.

ii. To accelerate the company immediately updating the customer information to booking list.

iii. To make it easier and fast to calculate total payment for company.

1.3 GROUP MEMBERS AND DISTRIBUTION OF WORKS

PROFILE PICTURES	DETAILS
	**SITI NURALIA NATASHYA BINTI HAMDAN** - Project Manager - Report Writer
	NURAINAA BALQIS BINTI MOHD ADLY HAFEZ HEDAYAT - Programmer 2 - Report Writer
	MADIHAH BINTI MOKHTAR - Report Writer - Assistant programmer
	NUR HANIS NATASYA BINTI GHAZALI - Programmer 1 - Report Writer

2.0 SOURCE CODE

Class HotelBooking

```
public class HotelBooking {

    // ATTRIBUTES
    private String guestName;
    private String phoneNum;
    private int numOfGuest;
    private int bookingID;
    private String bookingDate;
    private int totalDay;
    private int roomNo;
    private char roomCategory;
    private boolean breakfast;
    private String paymentDate;
    private char paymentStatus;
    private String paymentMethod;

    // NORMAL CLASS
    public HotelBooking(String gn, String pn, int ng, int bi, String bd,
            int td, int r, char rc, boolean b, String pd, char ps, String pm) {
        guestName = gn;
        phoneNum = pn;
        numOfGuest = ng;
        bookingID = bi;
        bookingDate = bd;

        totalDay = td;

        roomNo = r;

        roomCategory = rc;

        breakfast = b;

        paymentDate = pd;

        paymentStatus = ps;

        paymentMethod = pm;

    }


    // SETTERS

    public void setGuestName(String gn) {

        guestName = gn;

    }


    public void setPhoneNum(String pn) {

        phoneNum = pn;

    }


    public void setNumOfGuest(int ng) {

        numOfGuest = ng;

    }


    public void setBookingID(int bi) {

        bookingID = bi;

    }
```

```
public void setBookingDate(String bd) {
    bookingDate = bd;
}

public void setTotalDay(int td) {
    totalDay = td;
}

public void setRoomNo(int r) {
    roomNo = r;
}

public void setRoomCategory(char rc) {
    roomCategory = rc;
}

public void setBreakfast(boolean b) {
    breakfast = b;
}

public void setPaymentDate(String pd) {
    paymentDate = pd;
}

public void setPaymentStatus(char ps) {
    paymentStatus = ps;
}

public void setPaymentMethod(String pm) {
    paymentMethod = pm;
}

// GETTERS
public String getGuestName() {
    return guestName;
}

public String getPhoneNum() {
    return phoneNum;
}

public int getNumOfGuest() {
    return numOfGuest;
}

public int getBookingID() {
    return bookingID;
}

public String getBookingDate() {
    return bookingDate;
```

```
}

public int getTotalDay() {
    return totalDay;
}

public int getRoomNo() {
    return roomNo;
}

public char getRoomCategory() {
    return roomCategory;
}

public boolean hasBreakfast() {
    return breakfast;
}

public String getPaymentDate() {
    return paymentDate;
}

public char getPaymentStatus() {
    return paymentStatus;
}
```

```
public String getPaymentMethod() {

    return paymentMethod;

}


// PROCESSOR

public double calculateTotalPayment() {

    double roomPrice;


    if (roomCategory == 'S') {

        roomPrice = 350;

    } else if (roomCategory == 'P') {

        roomPrice = 600;

    } else {

        roomPrice = 1350;

    }


    double totalPayment = roomPrice * totalDay;


    if (breakfast) {

        totalPayment += numOfGuest * 30;

    }


    return totalPayment;

}


// PRINTER
```

```
    public String toString() {
        return  "----------------------------------" + "\n" +
                        "            Hotel Booking Details                " + "\n" +
                        "----------------------------------" + "\n" +
                    " Guest Name           : " + guestName + "\n" +
                    " Phone Number       : " + phoneNum + "\n" +
                    " Number of Guest  : " + numOfGuest + "\n" +
                    " Booking ID             : " + bookingID + "\n" +
                    " Booking Date         : " + bookingDate + "\n" +
                    " Total Day of Stay: " + totalDay + "\n" +
                    " Room Number        : " + roomNo + "\n" +
                    " Room Category      : " + roomCategory + "\n" +
                    " Breakfast?              : " + breakfast + "\n" +
                    " Payment Date         : " + paymentDate + "\n" +
                    " Payment Status   : " + paymentStatus + "\n" +
                    " Payment Method   : " + paymentMethod + "\n" +
                    " Total Payment        : RM " + calculateTotalPayment() + "\n" +
                    "----------------------------------";
    }
}
```

Class LinkedList

```
public class LinkedList

{

   Node first;

   Node current;

   Node last;


   //default constructor

   public LinkedList()

   {

      first=current=last = null;

   }


  // check whether list is empty

   public boolean isEmpty()

   {

      return (first == null);

   }


 // insert at the front of list

   public void insertAtFront(Object insertItem)

   {

      Node newNode = new Node(insertItem);

    //create new node with value received


      if (isEmpty())
   {
      first = newNode;
      last = newNode;
   }
   else
   {
      newNode.next = first;
      first = newNode;
   }
}



// insert at the end of list
public void insertAtBack(Object insertItem)
{
   Node newNode = new Node(insertItem);

   if(isEmpty())
   {
      first = newNode;
      last = newNode;
   }
   else
   {
      last.next = newNode;
      last = newNode;
```

```
     }
   }



// delete element from front
  public Object removeFromFront()
  {
     Object removeItem = null;

     if(isEmpty())
     {
        return removeItem;
     }

     removeItem = first.data;//point to first.data

     if(first == last)
     {
        first = null;
        last = null;
     }
     else
        first = first.next;
    return removeItem;
  }
```

```
// delete element from second
public Object removeFromSecond1()
{
   Object removeItem = null;

   if(isEmpty())
   {
      return removeItem;
   }

   //point to second node data
   removeItem = first.next.data;

   if(first == last)
   {
      first = null;
      last = null;
   }

   else
      first = first.next.next;
  return removeItem;
}

// delete element from second
public Object removeFromSecond()
```

```
 {
   Object removeItem = null;


   if(isEmpty())
   {
      return removeItem;
   }


   if(first.next==null)
   {
      return null;


   }


   else{
      //point to second node data
      current = first.next;
      first.next = current.next;
      removeItem = current.data;
  }
  return removeItem;


 }


// delete element from back
 public Object removeFromBack()
```

```
{
  Object removeItem = null;


  if(isEmpty())
  {
    return removeItem;
  }


  removeItem = last.data;//store data should be remove


  if (first == last)
  {
    first = null;
    last = null;
  }
  else
  {
    current = first;
    while(current.next != last)
      current = current.next;
    last = current;
    last.next = null;
  }
  return removeItem;
}
```

```
 // get the first node
   public Object getFirst()
   {
      if(isEmpty())
         return null;
      else
      {
         current = first;
         return current.data;
      }
   }

 //Return element of the next node pointed by current node
   public Object getNext()
   {
      if(current == last)
         return null;
      else
      {
         current = current.next;
         return current.data;
      }
   }

}//end class LinkedList user define
```

Class Node

```
public class Node{
  Object data;
  Node next;
  //default constructor
  public Node(){
    data=null;
    next=null;
  }
  //normal contructor
  public Node(Object data){
    this.data=data;
    next=null;
  }
 }
```

Class LinkedListMain

```
import java.io.*;

import java.util.*;

import javax.swing.*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.awt.Font;


public class LinkedListMain {

  private static final String BOOKING_FILE_PATH = "D:\\SEM3\\PROJECT\\Booking.txt";


  public static void main(String[] args) {

    java.util.LinkedList<HotelBooking> hotelBookings = new java.util.LinkedList<>();


    // Read data from "Booking.txt" file and populate the linked list

    try {

      BufferedReader inFile = new BufferedReader(new
FileReader(BOOKING_FILE_PATH));

      String line;

      while ((line = inFile.readLine()) != null) {

        String[] bookingData = line.split(";");

        HotelBooking newBooking = new HotelBooking(

            bookingData[0], bookingData[1], Integer.parseInt(bookingData[2]),

            Integer.parseInt(bookingData[3]), bookingData[4],
Integer.parseInt(bookingData[5]),

            Integer.parseInt(bookingData[6]), bookingData[7].charAt(0),
                    Boolean.parseBoolean(bookingData[8]), bookingData[9],
bookingData[10].charAt(0),
                    bookingData[11]
                );
                hotelBookings.add(newBooking);
            }
            inFile.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error reading data from file: " +
e.getMessage());
        }

      int choice;
                    boolean exitProgram = false;

                    do {
                        String[] options =
                        {
                            "Exit",
                            "Add Booking",
                            "Remove Booking",
                            "Display Unpaid Bookings",
                            "Display Paid Bookings",
                            "Display Upcoming Bookings",
                            "Update Payment Status",
                            "Calculate Total Payments"
                        };


                        choice = JOptionPane.showOptionDialog(
                            null,
                            " --- Hotel Management System ---\n 1. Add Booking\n 2. Remove
Booking\n 3. Display Unpaid Bookings\n"
                    + " 4. Display Paid Bookings\n 5. Display Upcoming Bookings\n 6.
Update Payment Status\n 7. Calculate Total Payments\n 0. Exit",
                            "Hotel Management System",
                            JOptionPane.DEFAULT_OPTION,
                            JOptionPane.PLAIN_MESSAGE,
                            null,
                            options,
                            options[0]
                        );


                        // User clicked the 'Cancel' button or closed the dialog
                        if (choice == JOptionPane.CLOSED_OPTION) {
                            exitProgram = true;
                        } else if (choice == 1) {
                            addBooking(hotelBookings);
                            updateBookingFile(hotelBookings);
                        } else if (choice == 2) {
                            removeBooking(hotelBookings);
                            updateBookingFile(hotelBookings);
                        } else if (choice == 3) {
```

```
                        displayUnpaidBookings(hotelBookings);
                    } else if (choice == 4) {
                        displayPaidBookings(hotelBookings);
                    } else if (choice == 5) {
                        displayUpcomingBookings(hotelBookings);
                    } else if (choice == 6) {
                        updatePaymentStatus(hotelBookings);
                        updateBookingFile(hotelBookings);
                    } else if (choice == 7) {
                        calculateTotalPayments(hotelBookings);
                    } else if (choice == 0) {
                        exitProgram = true;
                        JOptionPane.showMessageDialog(null, "Exiting Hotel Management
System...");
                    } else {
                        JOptionPane.showMessageDialog(null, "Invalid choice. Please try
again.");
                    }
            } while (!exitProgram);
   }


        //Method for add booking
   public static void addBooking(java.util.LinkedList<HotelBooking> hotelBookings) {
      JOptionPane.showMessageDialog(null, "---- Add Booking ----");


      String guestName = JOptionPane.showInputDialog(null, "Guest Name:");
```

```
        String phoneNumber = JOptionPane.showInputDialog(null, "Phone Number:");
        int numOfGuests = Integer.parseInt(JOptionPane.showInputDialog(null, "Number of
Guests:"));
        int bookingID = Integer.parseInt(JOptionPane.showInputDialog(null, "Booking ID:"));
        String bookingDate = JOptionPane.showInputDialog(null, "Booking Date:");
        int totalDay = Integer.parseInt(JOptionPane.showInputDialog(null, "Total Day of
Stay:"));
        int roomNo = Integer.parseInt(JOptionPane.showInputDialog(null, "Room Number:"));
        char roomCategory = JOptionPane.showInputDialog(null, "Room Category:").charAt(0);
        boolean breakfast = Boolean.parseBoolean(JOptionPane.showInputDialog(null,
"Breakfast Included [true/false]:"));
        String paymentDate = JOptionPane.showInputDialog(null, "Payment Date:");
        char paymentStatus = JOptionPane.showInputDialog(null, "Payment
Status:").charAt(0);
        String paymentMethod = JOptionPane.showInputDialog(null, "Payment Method:");


        HotelBooking newBooking = new HotelBooking(guestName, phoneNumber,
numOfGuests, bookingID, bookingDate, totalDay,
                roomNo, roomCategory, breakfast, paymentDate, paymentStatus,
paymentMethod);


        hotelBookings.add(newBooking);
        JOptionPane.showMessageDialog(null, "Booking added successfully.");
    }


          //Method to remove booking by using booking ID
    public static void removeBooking(java.util.LinkedList<HotelBooking> hotelBookings) {
        JOptionPane.showMessageDialog(null, "---- Remove Booking ----");
        int bookingID = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the Booking
ID to remove:"));

        boolean removed = false;

        for (HotelBooking booking : hotelBookings) {
            if (booking.getBookingID() == bookingID) {
                hotelBookings.remove(booking);
                removed = true;
                JOptionPane.showMessageDialog(null, "Booking removed successfully.");
                break;
            }
        }

        if (!removed) {
            JOptionPane.showMessageDialog(null, "Booking ID not found.");
        }
    }

            //Method to display all the unpaid bookings
    public static void displayUnpaidBookings(java.util.LinkedList<HotelBooking>
hotelBookings) {
        JOptionPane.showMessageDialog(null, "--------- Unpaid Bookings ---------");

        boolean found = false;
        StringBuilder unpaidBookings = new StringBuilder();


        for (HotelBooking booking : hotelBookings) {
            if (booking.getPaymentStatus() == 'N') {
                unpaidBookings.append(booking).append("\n\n");
                found = true;
            }
        }


        if (!found) {
            unpaidBookings.append("No unpaid bookings found.");
        }


        JTextArea textArea = new JTextArea(unpaidBookings.toString(), 20, 50);
        textArea.setEditable(false);
        textArea.setFont(new Font("Courier New", Font.PLAIN, 12)); // Set the font to a fixed-
width font


        JScrollPane scrollPane = new JScrollPane(textArea);


        JOptionPane.showMessageDialog(null, scrollPane, "Unpaid Bookings",
JOptionPane.PLAIN_MESSAGE);
    }


    //Method to display all the paid bookings
    public static void displayPaidBookings(java.util.LinkedList<HotelBooking> hotelBookings)
{
        JOptionPane.showMessageDialog(null, "--------- Paid Bookings ---------");


        boolean found = false;

        StringBuilder paidBookings = new StringBuilder();


        for (HotelBooking booking : hotelBookings) {

            if (booking.getPaymentStatus() == 'D') {

                paidBookings.append(booking).append("\n\n");

                found = true;

            }

        }


        if (!found) {

            paidBookings.append("No paid bookings found.");

        }


        JTextArea textArea = new JTextArea(paidBookings.toString(), 20, 50);

        textArea.setEditable(false);

        textArea.setFont(new Font("Courier New", Font.PLAIN, 12)); // Set the font to a fixed-
width font


        JScrollPane scrollPane = new JScrollPane(textArea);


        JOptionPane.showMessageDialog(null, scrollPane, "Paid Bookings",
JOptionPane.PLAIN_MESSAGE);

    }
```

```
        //Method to display all the upcoming bookings
    public static void displayUpcomingBookings(java.util.LinkedList<HotelBooking>
hotelBookings) {
        JOptionPane.showMessageDialog(null, "-------- Upcoming Bookings --------");

        boolean found = false;
        String currentDate = getCurrentDate();

        // Format the specified date '20230707' for comparison
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        LocalDate specifiedDate = LocalDate.parse("20230707", formatter);

        StringBuilder upcomingBookings = new StringBuilder();

        for (HotelBooking booking : hotelBookings) {
            LocalDate bookingDate = LocalDate.parse(booking.getBookingDate(), formatter);
            // Check if booking date is after the specified date '20230707'
            if (bookingDate.isAfter(specifiedDate)) {
                upcomingBookings.append(booking).append("\n\n");
                found = true;
            }
        }

        if (!found) {
            upcomingBookings.append("No upcoming bookings found.");
        }


        JTextArea textArea = new JTextArea(upcomingBookings.toString(), 20, 50);
        textArea.setEditable(false);
        textArea.setFont(new Font("Courier New", Font.PLAIN, 12)); // Set the font to a fixed-
width font


        JScrollPane scrollPane = new JScrollPane(textArea);


        JOptionPane.showMessageDialog(null, scrollPane, "Upcoming Bookings",
JOptionPane.PLAIN_MESSAGE);
    }


          //Method to update the payment status
    public static void updatePaymentStatus(java.util.LinkedList<HotelBooking>
hotelBookings) {
        JOptionPane.showMessageDialog(null, "---- Update Payment Status ----");


        int bookingID = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the Booking
ID to update payment status:"));


        boolean found = false;


        for (HotelBooking booking : hotelBookings) {
            if (booking.getBookingID() == bookingID) {
                char paymentStatus = JOptionPane.showInputDialog(null, "Enter the new
Payment Status (D or N):").toUpperCase()
```

```
                    .charAt(0);

                booking.setPaymentStatus(paymentStatus);
                JOptionPane.showMessageDialog(null, "Payment status updated successfully.");
                found = true;
                break;
            }
        }

        if (!found) {
            JOptionPane.showMessageDialog(null, "Booking ID not found.");
        }
    }

            //Method to calculate the total payments
    public static void calculateTotalPayments(java.util.LinkedList<HotelBooking>
hotelBookings) {
        JOptionPane.showMessageDialog(null, "----- Calculate Total Payments -----");

        int totalPaymentD = 0; // Total payments received from category D
        int totalPaymentN = 0; // Total payments received from category N

        for (HotelBooking booking : hotelBookings) {
            if (booking.getPaymentStatus() == 'D') {
                double totalPayment = booking.calculateTotalPayment();
                totalPaymentD += totalPayment;
            } else {
                double totalPayment = booking.calculateTotalPayment();
                totalPaymentN += totalPayment;
            }
        }
        JOptionPane.showMessageDialog(null, "Total Payment Received (D): RM " +
totalPaymentD + "\nTotal Payment Received (N): RM " + totalPaymentN);
    }


            //Method to format the date
    public static String getCurrentDate() {
        LocalDate currentDate = LocalDate.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        return currentDate.format(formatter);
    }


            //Method to update the booking file
    public static void updateBookingFile(java.util.LinkedList<HotelBooking> hotelBookings) {
        try {
            PrintWriter writer = new PrintWriter(BOOKING_FILE_PATH);

            for (HotelBooking booking : hotelBookings) {
                writer.println(booking.toString());
            }

            writer.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error updating booking file: " +
e.getMessage());
        }
    }

    // LinkedList class to manage the linked list of HotelBooking nodes
    public static class HotelBookingLinkedList {
        private Node head; // Head of the linked list

        public HotelBookingLinkedList() {
            head = null;
        }

        // Method to add a new HotelBooking node to the linked list
        public void addBooking(HotelBooking newBooking) {
            Node newNode = new Node(newBooking);
            if (head == null) {
                head = newNode;
            } else {
                Node currentNode = head;
                while (currentNode.next != null) {
                    currentNode = currentNode.next;
                }
                currentNode.next = newNode;
            }
```

```
}


// Method to remove a HotelBooking node from the linked list based on booking ID
public void removeBooking(int bookingID) {
    if (head == null) {
        System.out.println("Linked list is empty.");
        return;
    }

    if (((HotelBooking) head.data).getBookingID() == bookingID) {
        head = head.next;
        return;
    }

    Node prevNode = head;
    Node currentNode = head.next;
    while (currentNode != null) {
    if (((HotelBooking) currentNode.data).getBookingID() == bookingID) {
        prevNode.next = currentNode.next;
        return;
    }
        prevNode = currentNode;
        currentNode = currentNode.next;
    }

    System.out.println("Booking ID not found.");
```

```
        }


        // Method to display all unpaid bookings in the linked list
        public void displayUnpaidBookings() {
            Node currentNode = head;
            boolean found = false;


            while (currentNode != null) {
                HotelBooking booking = (HotelBooking) currentNode.data;
                if (booking.getPaymentStatus() == 'N') {
                    System.out.println(booking);
                    found = true;
                }
                currentNode = currentNode.next;
             }


            if (!found) {
                System.out.println("No unpaid bookings found.");
            }
        }
    }
}
```

Class Queue

```
//import java.util.*;

public class Queue extends LinkedList

{

  public Queue() { } // constructor

  public void enqueue( Object elem)

  { insertAtBack (elem); }


  public Object dequeue ( )

  { return removeFromFront(); }


  public Object getFront()

  { return getFirst(); }


  public Object getEnd()

  {  Object O = removeFromBack();

    insertAtBack(O);

    return O;

  }


  public String toString(){

    String print="";

    Object obj;

    LinkedList temp = new LinkedList();

    while(!this.isEmpty())

    {
      obj = this.dequeue();
      print = print + obj +" ";
      temp.insertAtBack(obj);
    }
    while(!temp.isEmpty())
      this.enqueue(temp.removeFromFront());

    return print;
  }//end toString()
} // end Queue
```

Class HotelQueue

```
import java.util.NoSuchElementException;

public class HotelQueue {
    private static class Node {
        private HotelBooking data;
        private Node next;

        public Node(HotelBooking data) {
            this.data = data;
        }
    }

    private Node head;
    private Node tail;
```

```
public void enqueue(HotelBooking data) {
    Node newNode = new Node(data);
    if (tail == null) {
        head = newNode;
        tail = newNode;
    } else {
        tail.next = newNode;
        tail = newNode;
    }
}

public HotelBooking dequeue() {
    if (head == null) {
        throw new NoSuchElementException();
    }

    HotelBooking data = head.data;
    head = head.next;
    if (head == null) {
        tail = null;
    }
    return data;
}

public boolean isEmpty() {
    return head == null;
}

// Additional method to get all the bookings as an array
public HotelBooking[] toArray() {
    HotelBooking[] bookingsArray = new HotelBooking[size()];
    Node current = head;
    int index = 0;
    while (current != null) {
        bookingsArray[index++] = current.data;
        current = current.next;
    }
    return bookingsArray;
}

// Additional method to remove a specific booking
public void removeBooking(HotelBooking booking) {
    Node current = head;
    Node prev = null;
    while (current != null) {
        if (current.data.equals(booking)) {
            if (prev == null) {
                head = current.next;
            } else {
                prev.next = current.next;
            }
```

```
                if (current.next == null) {
                    tail = prev;
                }
                return;
            }
            prev = current;
            current = current.next;
        }
    }

    // Additional method to get the size of the queue
    public int size() {
        int count = 0;
        Node current = head;
        while (current != null) {
            count++;
            current = current.next;
        }
        return count;
    }
}
```

Class HotelQueueMain

```
import java.io.*;

import java.util.*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import javax.swing.JScrollPane;

import java.awt.Font;


public class HotelQueueMain {

    private static final String BOOKING_FILE_PATH = "D:\\SEM3\\PROJECT\\Booking[2].txt";


    public static void main(String[] args) {

        HotelQueue hotelBookings = new HotelQueue();


        // Read data from "Booking.txt" file and populate the queue

        try {

            BufferedReader inFile = new BufferedReader(new
FileReader(BOOKING_FILE_PATH));

            String line;

            while ((line = inFile.readLine()) != null) {

                String[] bookingData = line.split(";");

                HotelBooking newBooking = new HotelBooking(

                        bookingData[0], bookingData[1], Integer.parseInt(bookingData[2]),
                        Integer.parseInt(bookingData[3]), bookingData[4],
Integer.parseInt(bookingData[5]),
                        Integer.parseInt(bookingData[6]), bookingData[7].charAt(0),
                        Boolean.parseBoolean(bookingData[8]), bookingData[9],
bookingData[10].charAt(0),
                        bookingData[11]
                );
                hotelBookings.enqueue(newBooking);
            }
            inFile.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error reading data from file: " +
e.getMessage());
        }

        int choice;
                boolean exitProgram = false;

                do {
                    String choiceString = JOptionPane.showInputDialog(getMenu());
                    if (choiceString == null) {
                        // User clicked the 'Cancel' button or closed the dialog
                        exitProgram = true;
                        JOptionPane.showMessageDialog(null, "Exiting Hotel Management
System...");
                    } else {
                        try {
```

```
                choice = Integer.parseInt(choiceString);


                if (choice == 1) {
                    addBooking(hotelBookings);
                    updateBookingFile(hotelBookings);
                } else if (choice == 2) {
                    removeBooking(hotelBookings);
                    updateBookingFile(hotelBookings);
                } else if (choice == 3) {
                    updatePaymentStatus(hotelBookings);
                    updateBookingFile(hotelBookings);
                } else if (choice == 4) {
                    displayUnpaidBookings(hotelBookings);
                } else if (choice == 5) {
                    displayPaidBookings(hotelBookings);
                } else if (choice == 6) {
                    displayUpcomingBookings(hotelBookings);
                } else if (choice == 7) {
                    calculateTotalPayments(hotelBookings);
                } else if (choice == 0) {
                    exitProgram = true;
                    JOptionPane.showMessageDialog(null, "Exiting Hotel Management
System...");
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid choice. Please try
again.");
```

```
                    }
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Invalid input. Please enter a
number.");
                }
            }
        } while (!exitProgram);
    }

        //Method for JOPane display
    public static String getMenu() {
        return "---- Hotel Management System ----\n" +
                "1. Add Booking\n" +
                "2. Remove Booking\n" +
                "3. Update Payment Status\n" +
                "4. Display Unpaid Bookings\n" +
                "5. Display Paid Bookings\n" +
                "6. Display Upcoming Bookings\n" +
                "7. Calculate Total Payments\n" +
                "0. Exit\n" +
                "--------------------------------";
    }

        //Method for add booking
    public static void addBooking(HotelQueue hotelBookings) {
        System.out.println("---- Add Booking ----");
        // Get input for new booking using JOptionPane
        String guestName = JOptionPane.showInputDialog("Guest Name:");
        String phoneNumber = JOptionPane.showInputDialog("Phone Number:");
        int numOfGuests = Integer.parseInt(JOptionPane.showInputDialog("Number of
Guests:"));
        int bookingID = Integer.parseInt(JOptionPane.showInputDialog("Booking ID:"));
        String bookingDate = JOptionPane.showInputDialog("Booking Date:");
        int totalDay = Integer.parseInt(JOptionPane.showInputDialog("Total Day of Stay:"));
        int roomNo = Integer.parseInt(JOptionPane.showInputDialog("Room Number:"));
        char roomCategory = JOptionPane.showInputDialog("Room Category:").charAt(0);
        boolean breakfast = Boolean.parseBoolean(JOptionPane.showInputDialog("Breakfast
Included [true/false]:"));
        String paymentDate = JOptionPane.showInputDialog("Payment Date:");
        char paymentStatus = JOptionPane.showInputDialog("Payment Status:").charAt(0);
        String paymentMethod = JOptionPane.showInputDialog("Payment Method:");

        // Create a new HotelBooking object
        HotelBooking newBooking = new HotelBooking(
            guestName, phoneNumber, numOfGuests, bookingID, bookingDate, totalDay,
            roomNo, roomCategory, breakfast, paymentDate, paymentStatus, paymentMethod
        );

        // Enqueue the new booking to the queue
        hotelBookings.enqueue(newBooking);
        JOptionPane.showMessageDialog(null, "Booking added successfully.");
```

```
}


        //Method to remove booking by using booking ID
public static void removeBooking(HotelQueue hotelBookings) {
    System.out.println("---- Remove Booking ----");
    String input = JOptionPane.showInputDialog("Enter the Booking ID to remove:");
    if (input == null) {
        return; // User clicked the 'Close' button or closed the dialog
    }

    try {
        int bookingID = Integer.parseInt(input);
        boolean removed = false;
        int initialSize = hotelBookings.size();

        // Create a temporary queue to store non-matching bookings
        HotelQueue tempQueue = new HotelQueue();

        for (int i = 0; i < initialSize; i++) {
            HotelBooking booking = hotelBookings.dequeue();
            if (booking.getBookingID() == bookingID) {
                removed = true;
                JOptionPane.showMessageDialog(null, "Booking removed successfully.");
            } else {
                tempQueue.enqueue(booking);
            }
            }

            // Copy the remaining bookings back to the original queue
            while (!tempQueue.isEmpty()) {
                hotelBookings.enqueue(tempQueue.dequeue());
            }

            if (!removed) {
                JOptionPane.showMessageDialog(null, "Booking ID not found.");
             }
              }
                    catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Invalid input. Please enter a valid
Booking ID.");
        }
    }

            //Method to update the payment status
    public static void updatePaymentStatus(HotelQueue hotelBookings) {
        System.out.println("---- Update Payment Status ----");

        String bookingIDString = JOptionPane.showInputDialog("Enter the Booking ID to
update payment status:");
        if (bookingIDString == null) {
            return; // User clicked the 'Close' button or closed the dialog
        }
        try {
            int bookingID = Integer.parseInt(bookingIDString);
            boolean found = false;
            int initialSize = hotelBookings.size();

            HotelQueue tempQueue = new HotelQueue();

            for (int i = 0; i < initialSize; i++) {
                HotelBooking booking = hotelBookings.dequeue();
                if (booking.getBookingID() == bookingID) {
                    String paymentStatusInput = JOptionPane.showInputDialog("Enter the new
Payment Status (D or N):");
                    if (paymentStatusInput == null || paymentStatusInput.isEmpty()) {
                        JOptionPane.showMessageDialog(null, "Invalid payment status input.");
                        return;
                    }
                    char paymentStatus = paymentStatusInput.toUpperCase().charAt(0);
                    booking.setPaymentStatus(paymentStatus);
                    JOptionPane.showMessageDialog(null, "Payment status updated
successfully.");
                    found = true;
                }
                tempQueue.enqueue(booking);
            }
            while (!tempQueue.isEmpty()) {
                hotelBookings.enqueue(tempQueue.dequeue()); // Enqueue all bookings back into
hotelBookings
            }

            if (!found) {
                JOptionPane.showMessageDialog(null, "Booking ID not found.");
            }
            }
            catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Invalid input. Please enter a valid
Booking ID.");
        }
    }

            //Method to display all the unpaid bookings
    public static void displayUnpaidBookings(HotelQueue hotelBookings) {
        StringBuilder unpaidBookings = new StringBuilder();
        boolean found = false;

        HotelQueue tempQueue = new HotelQueue();

        while (!hotelBookings.isEmpty()) {
            HotelBooking booking = hotelBookings.dequeue();
            if (booking.getPaymentStatus() == 'N') {
                unpaidBookings.append(booking.toString()).append("\n");
                found = true;
            }
            tempQueue.enqueue(booking);
        }

        while (!tempQueue.isEmpty()) {
            hotelBookings.enqueue(tempQueue.dequeue());
        }

        if (found) {
             JTextArea textArea = new JTextArea(unpaidBookings.toString(), 20, 50);
                    textArea.setEditable(false);
                                    textArea.setFont(new Font("Courier New", Font.PLAIN, 12)); // Set the
font to a fixed-width font
                    JScrollPane scrollPane = new JScrollPane(textArea);

                    JOptionPane.showMessageDialog(null, scrollPane, "Unpaid Bookings",
JOptionPane.PLAIN_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "No unpaid bookings found.");
        }
    }

    //Method to display all the paid bookings
    public static void displayPaidBookings(HotelQueue hotelBookings) {
        StringBuilder paidBookings = new StringBuilder();
        boolean found = false;


        HotelQueue tempQueue = new HotelQueue();


        while (!hotelBookings.isEmpty()) {
            HotelBooking booking = hotelBookings.dequeue();
            if (booking.getPaymentStatus() == 'D') {
                paidBookings.append(booking.toString()).append("\n");
                found = true;
            }
            tempQueue.enqueue(booking);
        }


        while (!tempQueue.isEmpty()) {
            hotelBookings.enqueue(tempQueue.dequeue());
        }


        if (found) {
             JTextArea textArea = new JTextArea(paidBookings.toString(), 20, 50);
                    textArea.setEditable(false);
                                textArea.setFont(new Font("Courier New", Font.PLAIN, 12)); // Set the
font to a fixed-width font
                    JScrollPane scrollPane = new JScrollPane(textArea);


                    JOptionPane.showMessageDialog(null, scrollPane, "Paid Bookings",
JOptionPane.PLAIN_MESSAGE);
```

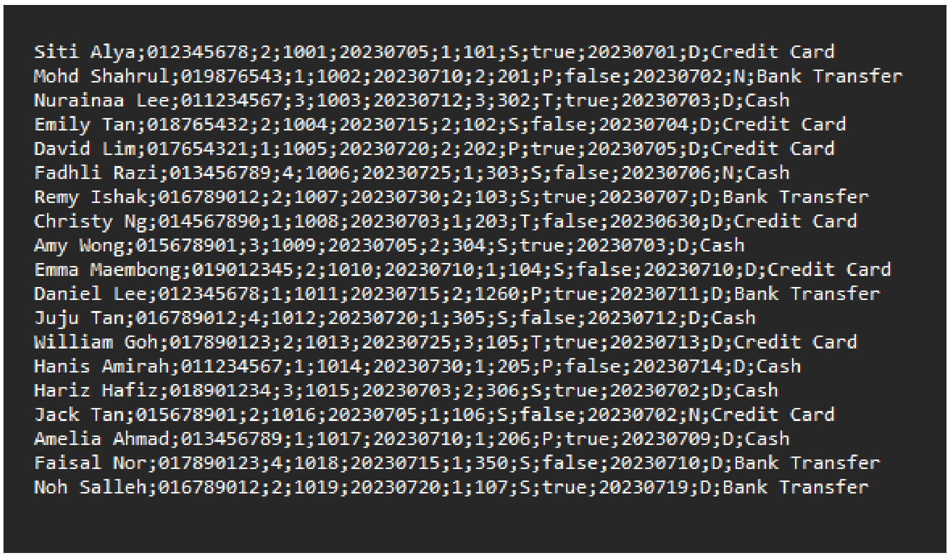
```
    } else {
        JOptionPane.showMessageDialog(null, "No paid bookings found.");
    }
}

        //Method to display all the upcoming bookings
public static void displayUpcomingBookings(HotelQueue hotelBookings) {
    StringBuilder upcomingBookings = new StringBuilder();
    boolean found = false;
    String currentDate = getCurrentDate();

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
    LocalDate specifiedDate = LocalDate.parse("20230707", formatter);

    HotelQueue tempQueue = new HotelQueue();

    while (!hotelBookings.isEmpty()) {
        HotelBooking booking = hotelBookings.dequeue();
        LocalDate bookingDate = LocalDate.parse(booking.getBookingDate(), formatter);
        if (bookingDate.isAfter(specifiedDate)) {
            upcomingBookings.append(booking.toString()).append("\n");
            found = true;
        }
        tempQueue.enqueue(booking);
    }
```

```
        while (!tempQueue.isEmpty()) {
            hotelBookings.enqueue(tempQueue.dequeue());
        }

        if (found) {
            JTextArea textArea = new JTextArea(upcomingBookings.toString(), 20, 50);
                    textArea.setEditable(false);
                                    textArea.setFont(new Font("Courier New", Font.PLAIN, 12)); // Set the
font to a fixed-width font
                    JScrollPane scrollPane = new JScrollPane(textArea);

                    JOptionPane.showMessageDialog(null, scrollPane, "Upcoming Bookings",
JOptionPane.PLAIN_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "No upcoming bookings found.");
      }
    }

            //Method to calculate the total payments
    public static void calculateTotalPayments(HotelQueue hotelBookings) {
        StringBuilder paymentSummary = new StringBuilder();

        double totalPaymentD = 0; // Total payments received from category D
        double totalPaymentN = 0; // Total payments received from category N

        HotelQueue tempQueue = new HotelQueue(); // Temporary queue to store the
bookings
```

```
        while (!hotelBookings.isEmpty()) {
            HotelBooking booking = hotelBookings.dequeue();
            if (booking.getPaymentStatus() == 'D') {
                totalPaymentD += booking.calculateTotalPayment();
            } else if (booking.getPaymentStatus() == 'N') {
                totalPaymentN += booking.calculateTotalPayment();
            }
            tempQueue.enqueue(booking);
        }

        while (!tempQueue.isEmpty()) {
            hotelBookings.enqueue(tempQueue.dequeue()); // Enqueue all bookings back into
hotelBookings
        }

        paymentSummary.append("Total Payment Received (D): RM
").append(totalPaymentD).append("\n");
        paymentSummary.append("Total Payment Received (N): RM
").append(totalPaymentN);
        JOptionPane.showMessageDialog(null, "---- Calculate Total Payments ----\n" +
paymentSummary.toString());
    }

            //Method to format the date
    public static String getCurrentDate() {
        LocalDate currentDate = LocalDate.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");

        return currentDate.format(formatter);

    }


          //Method to update the booking file

    public static void updateBookingFile(HotelQueue hotelBookings) {

        try {

            PrintWriter writer = new PrintWriter(BOOKING_FILE_PATH);


            HotelBooking[] bookingsArray = hotelBookings.toArray();

            for (HotelBooking booking : bookingsArray) {

                writer.println(booking.toString());

            }


            writer.close();

        } catch (IOException e) {

            JOptionPane.showMessageDialog(null, "Error updating booking file: " +
e.getMessage());

        }

    }

}
```

3.0 SAMPLE INPUT AND OUTPUT

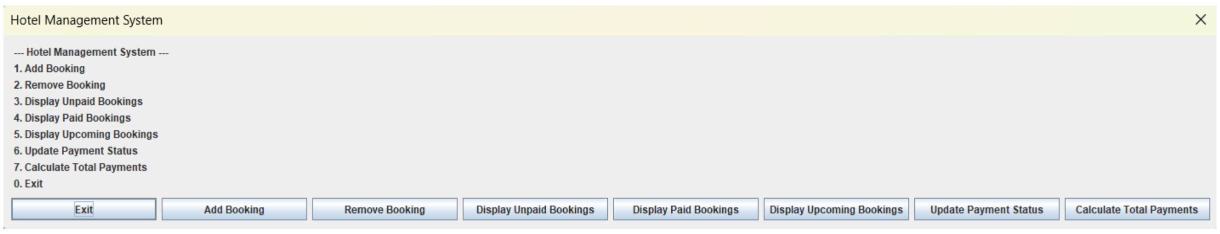
ORIGINAL TEXT FILE



```
Siti Alya;012345678;2;1001;20230705;1;101;S;true;20230701;D;Credit Card
Mohd Shahrul;019876543;1;1002;20230710;2;201;P;false;20230702;N;Bank Transfer
Nurainaa Lee;011234567;3;1003;20230712;3;302;T;true;20230703;D;Cash
Emily Tan;018765432;2;1004;20230715;2;102;S;false;20230704;D;Credit Card
David Lim;017654321;1;1005;20230720;2;202;P;true;20230705;D;Credit Card
Fadhli Razi;013456789;4;1006;20230725;1;303;S;false;20230706;N;Cash
Remy Ishak;016789012;2;1007;20230730;2;103;S;true;20230707;D;Bank Transfer
Christy Ng;014567890;1;1008;20230703;1;203;T;false;20230630;D;Credit Card
Amy Wong;015678901;3;1009;20230705;2;304;S;true;20230703;D;Cash
Emma Maembong;019012345;2;1010;20230710;1;104;S;false;20230710;D;Credit Card
Daniel Lee;012345678;1;1011;20230715;2;1260;P;true;20230711;D;Bank Transfer
Juju Tan;016789012;4;1012;20230720;1;305;S;false;20230712;D;Cash
William Goh;017890123;2;1013;20230725;3;105;T;true;20230713;D;Credit Card
Hanis Amirah;011234567;1;1014;20230730;1;205;P;false;20230714;D;Cash
Hariz Hafiz;018901234;3;1015;20230703;2;306;S;true;20230702;D;Cash
Jack Tan;015678901;2;1016;20230705;1;106;S;false;20230702;N;Credit Card
Amelia Ahmad;013456789;1;1017;20230710;1;206;P;true;20230709;D;Cash
Faisal Nor;017890123;4;1018;20230715;1;350;S;false;20230710;D;Bank Transfer
Noh Salleh;016789012;2;1019;20230720;1;107;S;true;20230719;D;Bank Transfer
```

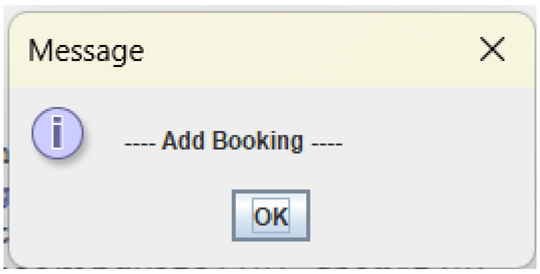
SAMPLE INPUT LINKEDLISTMAIN

Menu



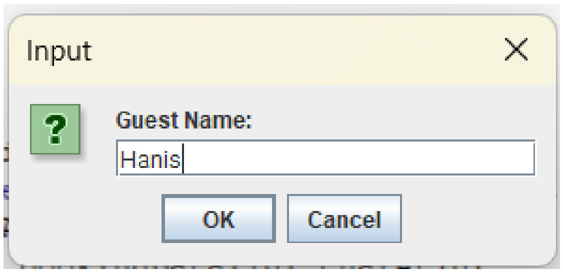
```
Hotel Management System
--- Hotel Management System ---
1. Add Booking
2. Remove Booking
3. Display Unpaid Bookings
4. Display Paid Bookings
5. Display Upcoming Bookings
6. Update Payment Status
7. Calculate Total Payments
0. Exit
```

AddBooking (Choice 1)



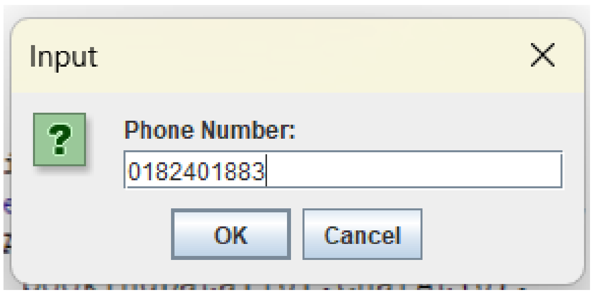
Message

---- Add Booking ----



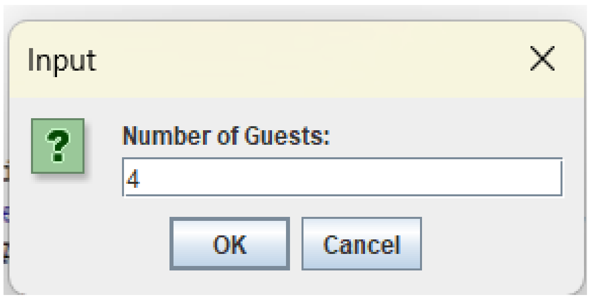
Input

Guest Name: Hanis


Input

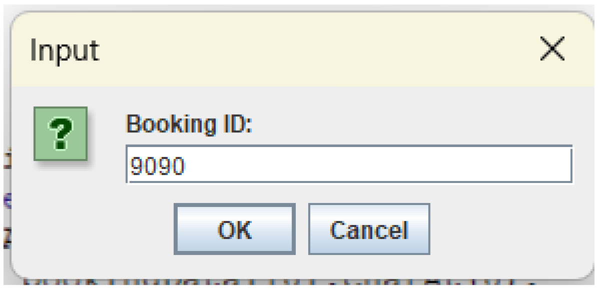
Phone Number: 0182401883

OK Cancel


Input

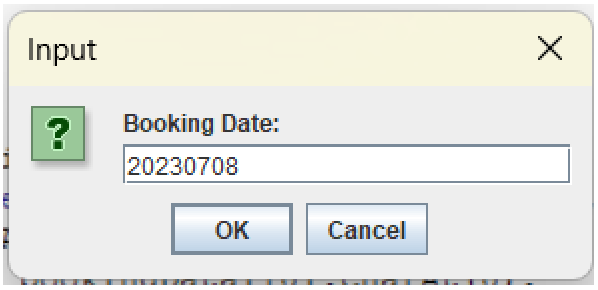
Number of Guests: 4

OK Cancel


Input

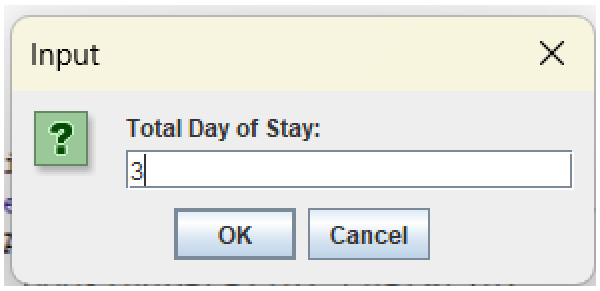
Booking ID: 9090

OK Cancel


Input

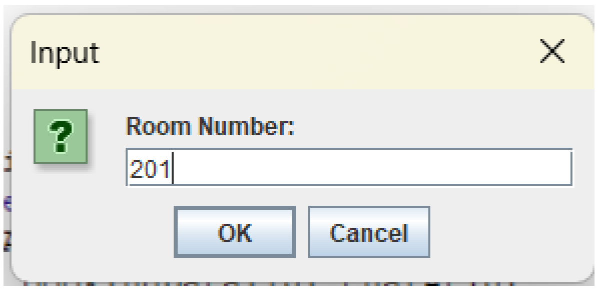
Booking Date: 20230708

OK Cancel


Input

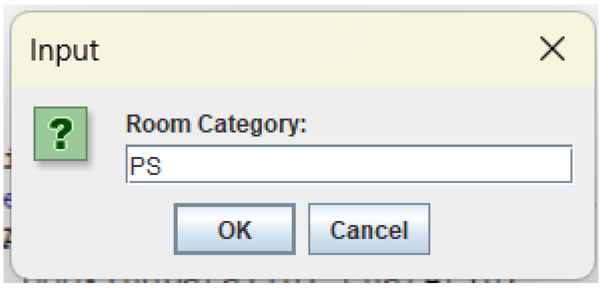
Total Day of Stay: 3

OK Cancel


Input

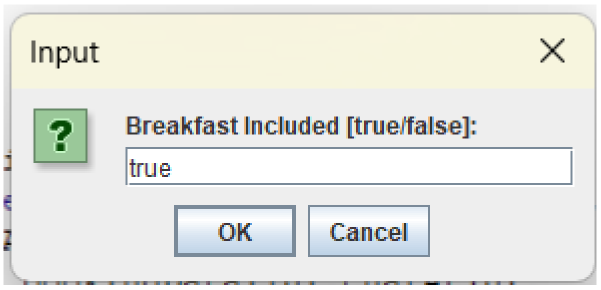
Room Number: 201

OK Cancel


Input

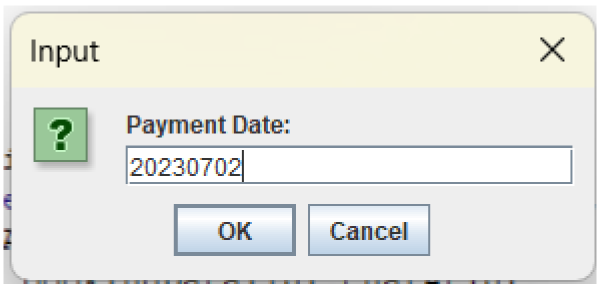
Room Category: PS

OK Cancel


Input

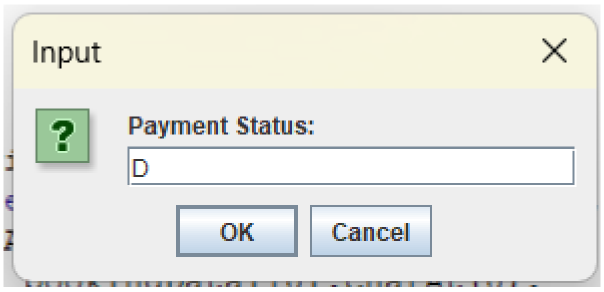
Breakfast Included [true/false]: true

OK Cancel


Input

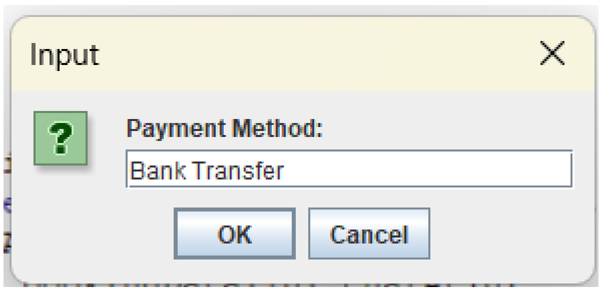
Payment Date: 20230702

OK Cancel


Input

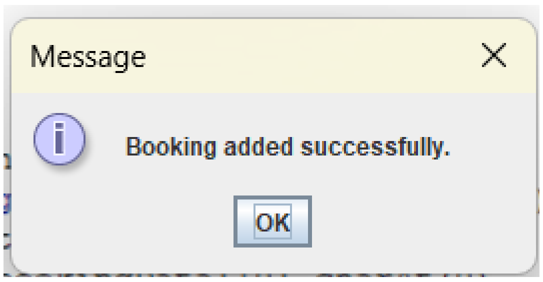
Payment Status: D

OK Cancel


Input

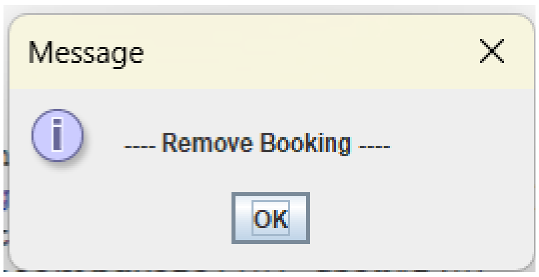
Payment Method:

Bank Transfer

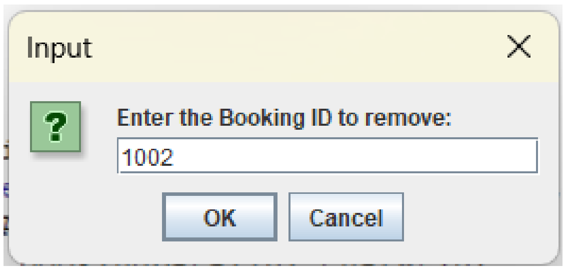

Message

Booking added successfully.

Remove Booking (Choice 2)

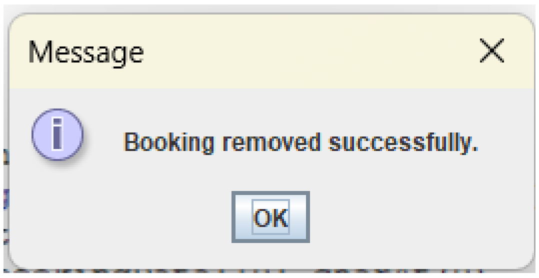

Message

---- Remove Booking ----


Input

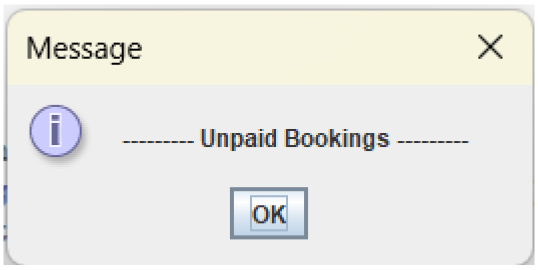
Enter the Booking ID to remove:

1002


Message

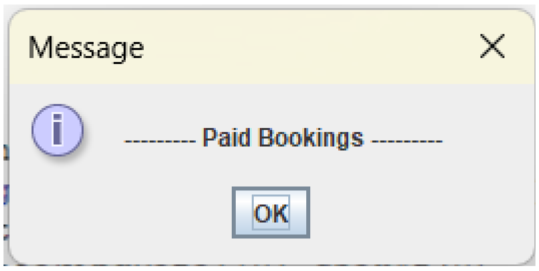
Booking removed successfully.

Display Unpaid Bookings (Choice 3)


Message

--------- Unpaid Bookings ---------

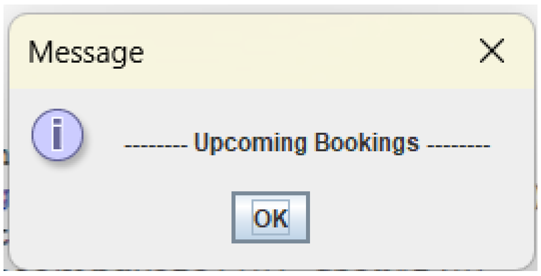
Display Paid Bookings (Choice 4)



Message

--------- Paid Bookings ---------

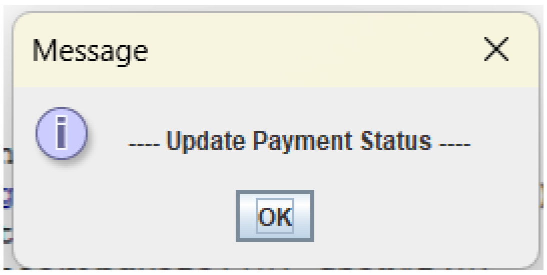
Display Upcoming Bookings (Choice 5)



Message

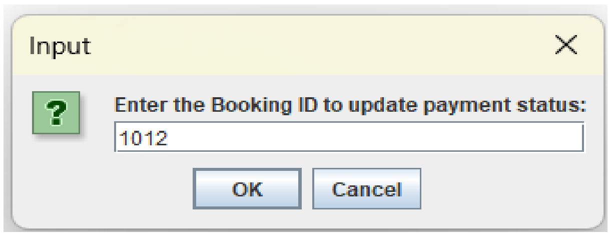
-------- Upcoming Bookings --------

Display Update Payment Status (Choice 6)



Message

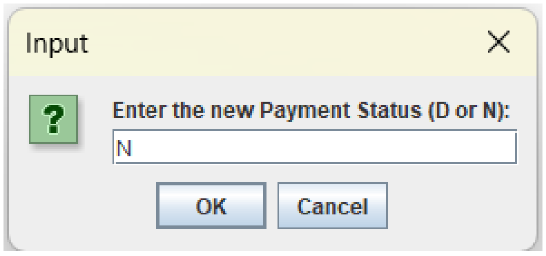
---- Update Payment Status ----



Input

Enter the Booking ID to update payment status:

1012

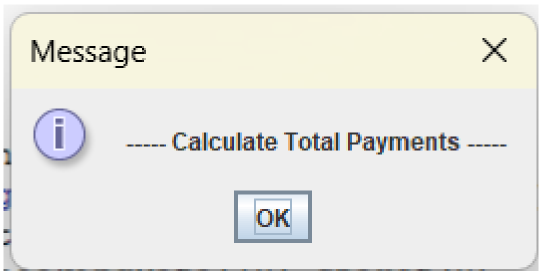


Input

Enter the new Payment Status (D or N):

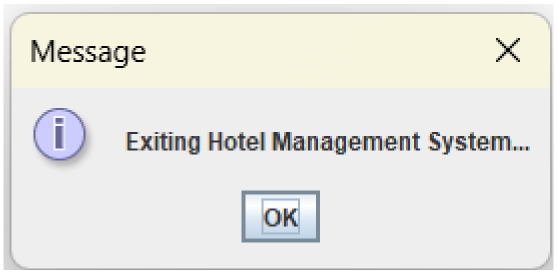
N

Display Calculate Total Payment (Choice 7)



```
----- Calculate Total Payments -----
```

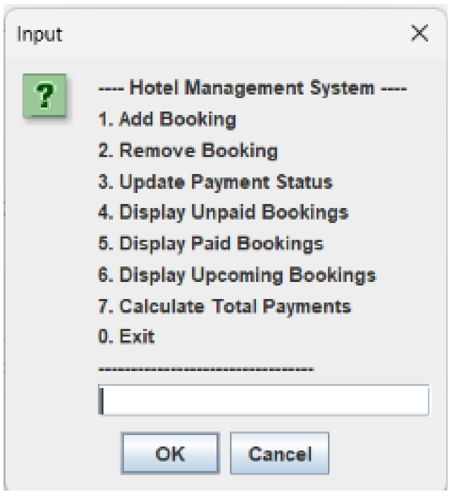
Display Exiting Hotel Management System (Choice 0)



```
Exiting Hotel Management System...
```

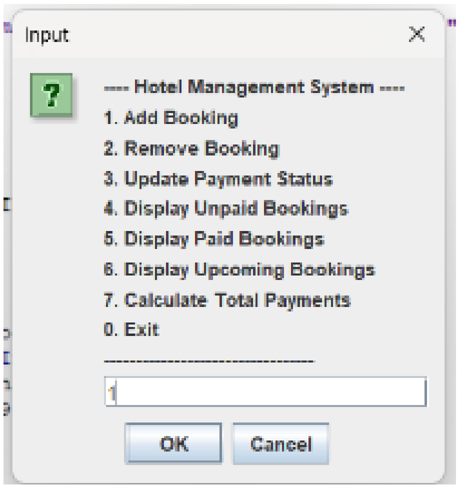
SAMPLE INPUT HOTELQUEUEMAIN

Menu

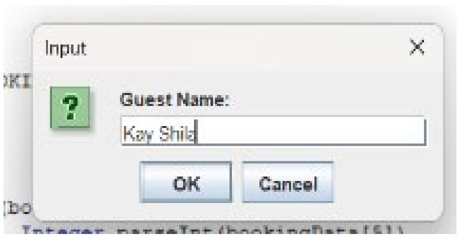


```
Input
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
--------------------------------
```

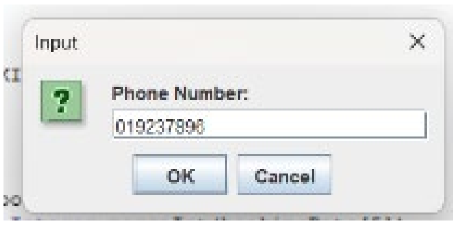
Add Booking (Choice 1)



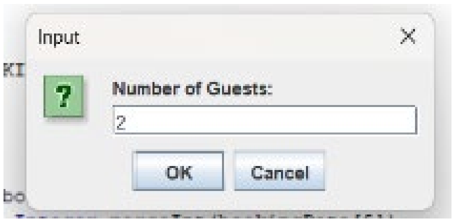
```
Input
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
--------------------------------
1
```



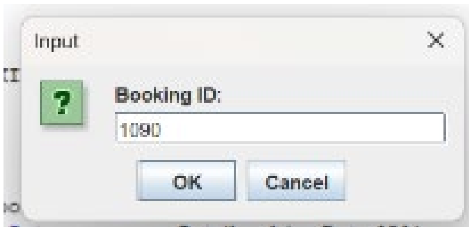
Input
Guest Name: Kay Shila



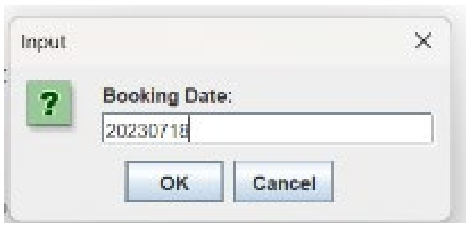
Input
Phone Number: 019237896



Input
Number of Guests: 2

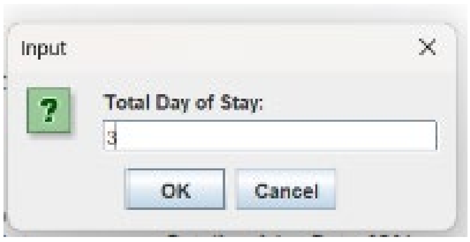


Input
Booking ID: 1090


Input

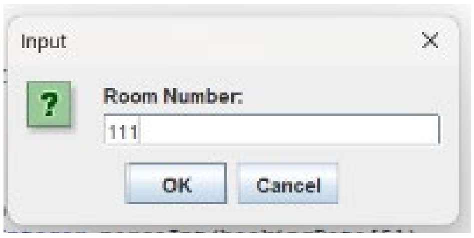
Booking Date:

20230718


Input

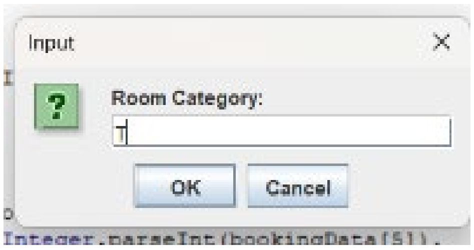
Total Day of Stay:

3


Input

Room Number:

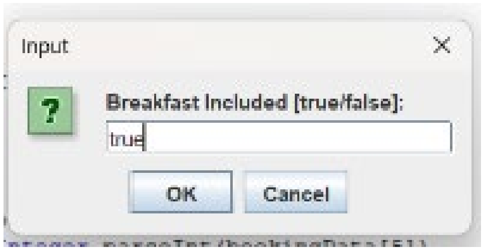
111


Input

Room Category:

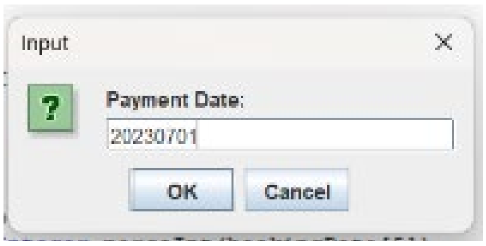
T

```
Integer.parseInt(bookingData[5]).
```


Input

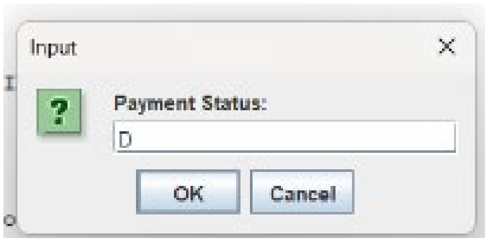
Breakfast Included [true/false]:

true


Input

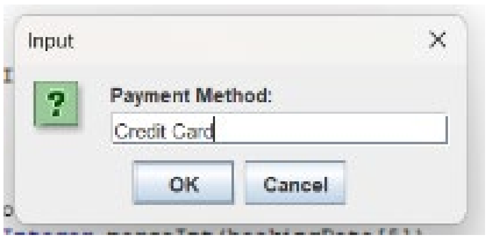
Payment Date:

20230701


Input

Payment Status:

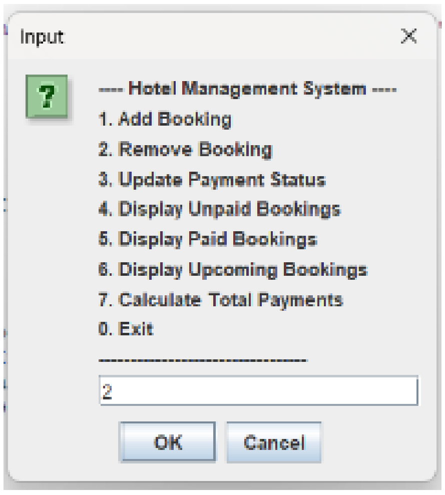
D


Input

Payment Method:

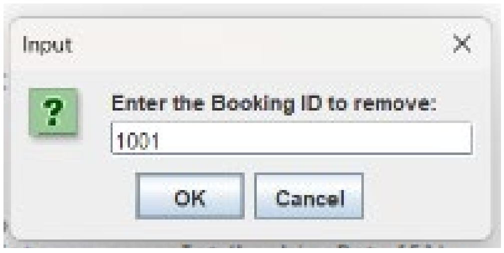
Credit Card

Remove booking (Choice 2)


Input

```
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
--------------------------------
```

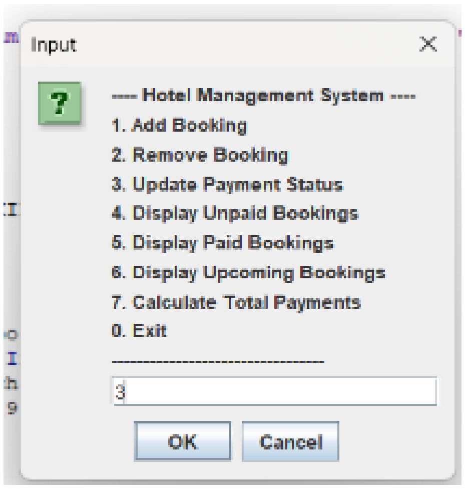
2


Input

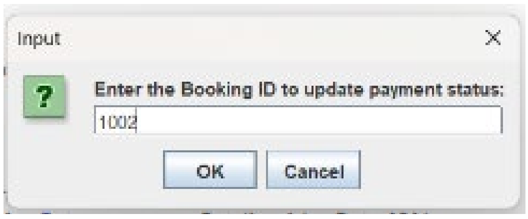
Enter the Booking ID to remove:

1001

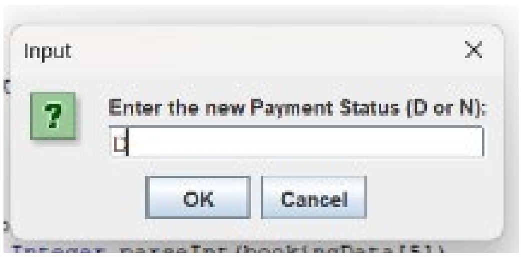
Update Payment Status (Choice 3)



```
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
---------------------------------
3
```

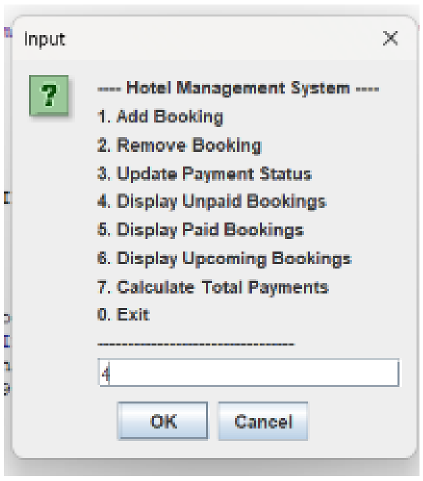


```
Enter the Booking ID to update payment status:
1002
```



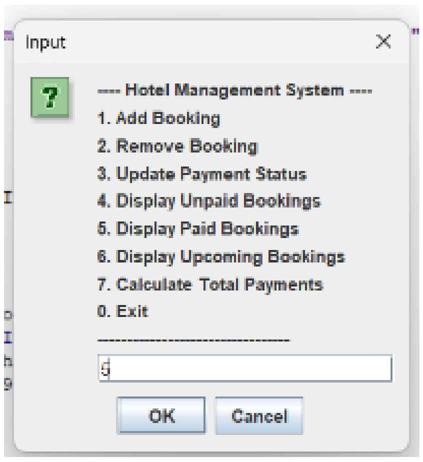
```
Enter the new Payment Status (D or N):
D
```

Display Unpaid Booking (Choice 4)



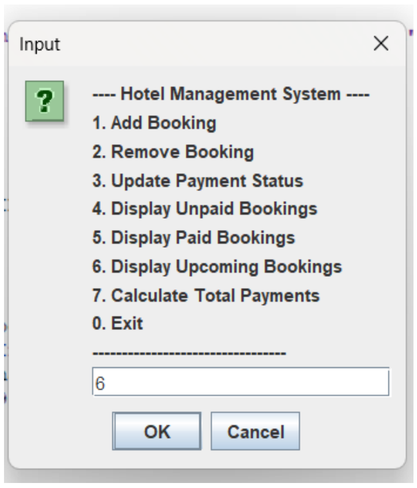
```
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
---------------------------------
4
```

Display Paid Booking (Choice 5)


Input

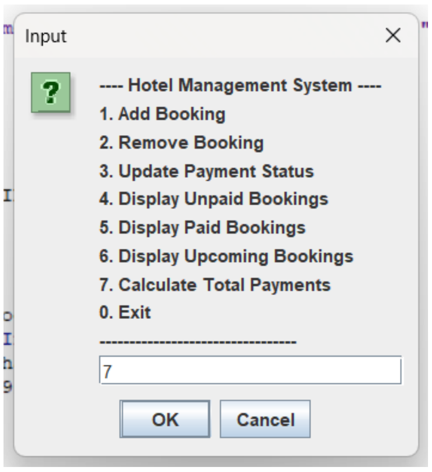
```
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
---------------------------------
5
```

Display Upcoming Booking (Choice 6)


Input

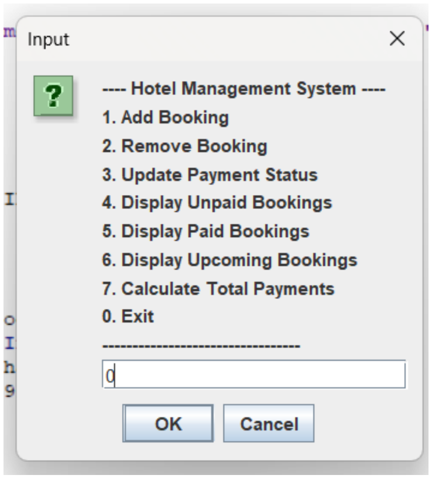
```
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
---------------------------------
6
```

Calculate Total Payment (Choice 7)



```
Input
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
--------------------------------
7
```

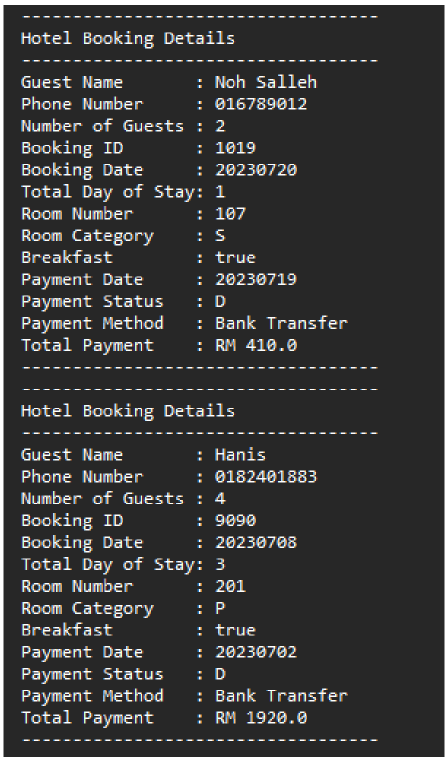
Exit Management System (Choice 0)



```
Input
---- Hotel Management System ----
1. Add Booking
2. Remove Booking
3. Update Payment Status
4. Display Unpaid Bookings
5. Display Paid Bookings
6. Display Upcoming Bookings
7. Calculate Total Payments
0. Exit
--------------------------------
0
```

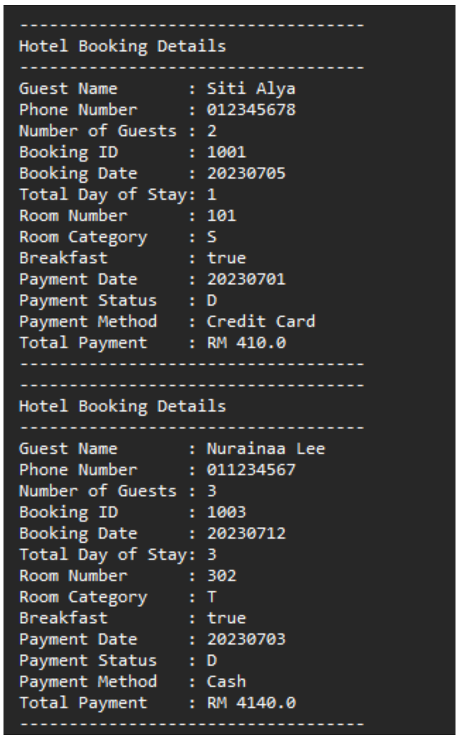
SAMPLE OUTPUT LINKEDLISTMAIN

Display Add Booking (Choice 1)



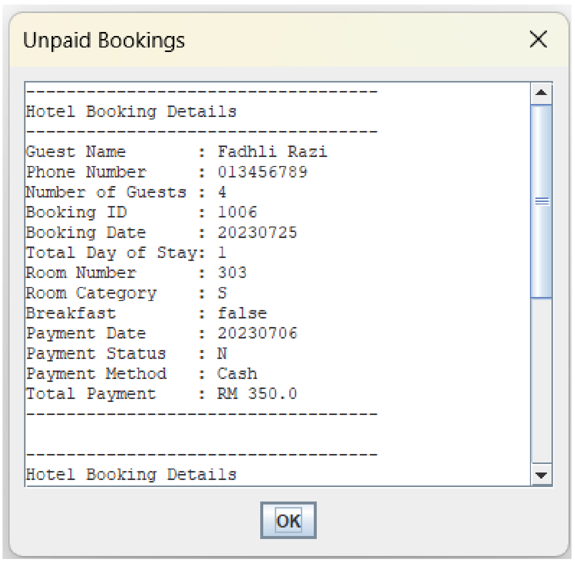
```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name       : Noh Salleh
Phone Number     : 016789012
Number of Guests : 2
Booking ID       : 1019
Booking Date     : 20230720
Total Day of Stay: 1
Room Number      : 107
Room Category    : S
Breakfast        : true
Payment Date     : 20230719
Payment Status   : D
Payment Method   : Bank Transfer
Total Payment    : RM 410.0
-----------------------------------
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name       : Hanis
Phone Number     : 0182401883
Number of Guests : 4
Booking ID       : 9090
Booking Date     : 20230708
Total Day of Stay: 3
Room Number      : 201
Room Category    : P
Breakfast        : true
Payment Date     : 20230702
Payment Status   : D
Payment Method   : Bank Transfer
Total Payment    : RM 1920.0
-----------------------------------
```

Display Remove Booking (Choice 2)

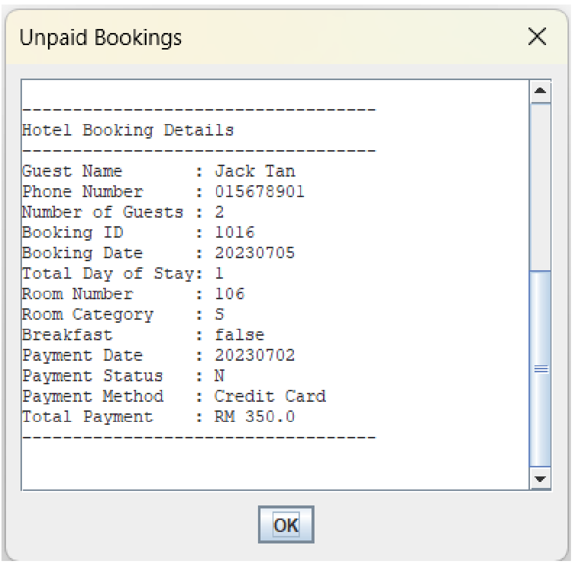


```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name       : Siti Alya
Phone Number     : 012345678
Number of Guests : 2
Booking ID       : 1001
Booking Date     : 20230705
Total Day of Stay: 1
Room Number      : 101
Room Category    : S
Breakfast        : true
Payment Date     : 20230701
Payment Status   : D
Payment Method   : Credit Card
Total Payment    : RM 410.0
-----------------------------------
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name       : Nurainaa Lee
Phone Number     : 011234567
Number of Guests : 3
Booking ID       : 1003
Booking Date     : 20230712
Total Day of Stay: 3
Room Number      : 302
Room Category    : T
Breakfast        : true
Payment Date     : 20230703
Payment Status   : D
Payment Method   : Cash
Total Payment    : RM 4140.0
-----------------------------------
```

Display Unpaid Bookings (Choice 3)

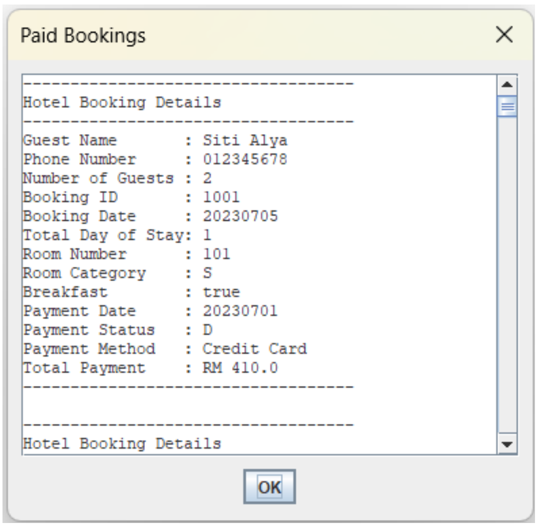


```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name        : Fadhli Razi
Phone Number      : 013456789
Number of Guests : 4
Booking ID        : 1006
Booking Date      : 20230725
Total Day of Stay: 1
Room Number       : 303
Room Category     : S
Breakfast         : false
Payment Date      : 20230706
Payment Status    : N
Payment Method    : Cash
Total Payment     : RM 350.0
------------------------------------

------------------------------------
Hotel Booking Details
```

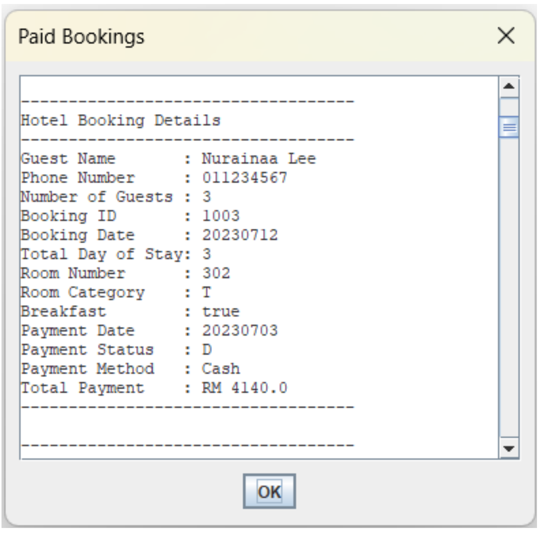


```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name        : Jack Tan
Phone Number      : 015678901
Number of Guests : 2
Booking ID        : 1016
Booking Date      : 20230705
Total Day of Stay: 1
Room Number       : 106
Room Category     : S
Breakfast         : false
Payment Date      : 20230702
Payment Status    : N
Payment Method    : Credit Card
Total Payment     : RM 350.0
------------------------------------
```

Display Paid Bookings (Choice 4)

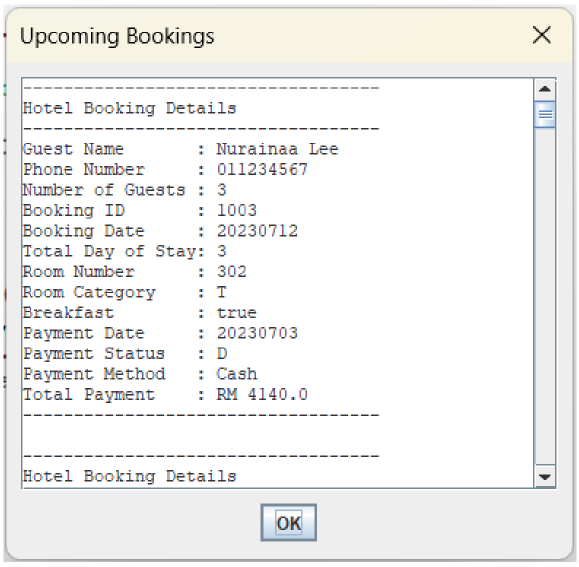


```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name        : Siti Alya
Phone Number      : 012345678
Number of Guests : 2
Booking ID        : 1001
Booking Date      : 20230705
Total Day of Stay: 1
Room Number       : 101
Room Category     : S
Breakfast         : true
Payment Date      : 20230701
Payment Status    : D
Payment Method    : Credit Card
Total Payment     : RM 410.0
------------------------------------

------------------------------------
Hotel Booking Details
```

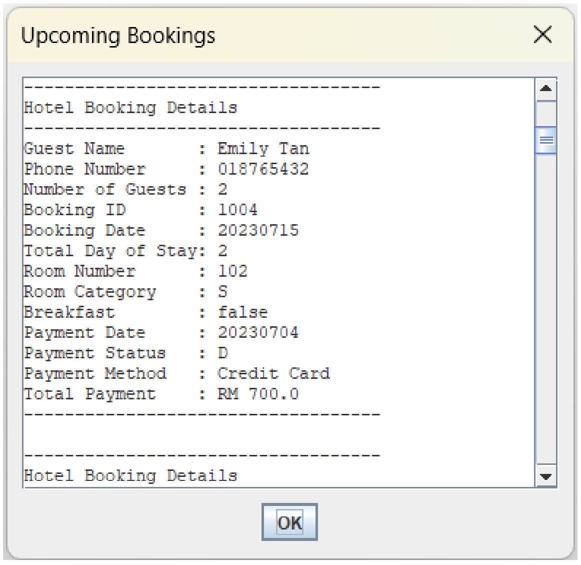


```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name        : Nurainaa Lee
Phone Number      : 011234567
Number of Guests : 3
Booking ID        : 1003
Booking Date      : 20230712
Total Day of Stay: 3
Room Number       : 302
Room Category     : T
Breakfast         : true
Payment Date      : 20230703
Payment Status    : D
Payment Method    : Cash
Total Payment     : RM 4140.0
------------------------------------

------------------------------------
```

Display Upcoming Bookings (Choice 5)

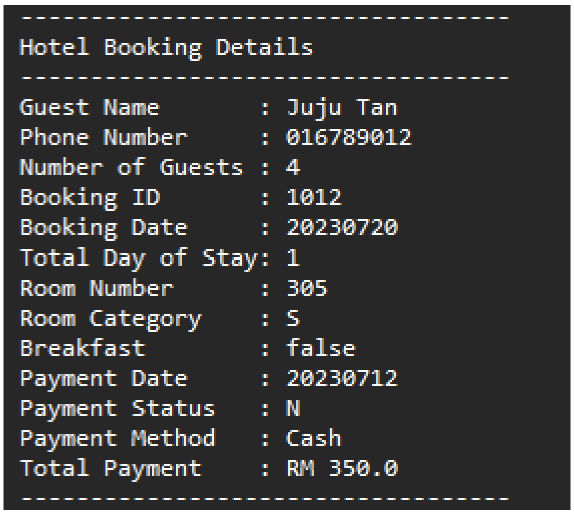


```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Nurainaa Lee
Phone Number      : 011234567
Number of Guests : 3
Booking ID        : 1003
Booking Date      : 20230712
Total Day of Stay: 3
Room Number       : 302
Room Category     : T
Breakfast         : true
Payment Date      : 20230703
Payment Status    : D
Payment Method    : Cash
Total Payment     : RM 4140.0
-----------------------------------

-----------------------------------
Hotel Booking Details
```



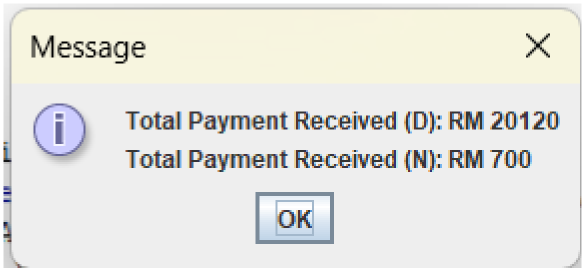
```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Emily Tan
Phone Number      : 018765432
Number of Guests : 2
Booking ID        : 1004
Booking Date      : 20230715
Total Day of Stay: 2
Room Number       : 102
Room Category     : S
Breakfast         : false
Payment Date      : 20230704
Payment Status    : D
Payment Method    : Credit Card
Total Payment     : RM 700.0
-----------------------------------

-----------------------------------
Hotel Booking Details
```

Display Updated Status Payment (Choice 6)



```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Juju Tan
Phone Number      : 016789012
Number of Guests : 4
Booking ID        : 1012
Booking Date      : 20230720
Total Day of Stay: 1
Room Number       : 305
Room Category     : S
Breakfast         : false
Payment Date      : 20230712
Payment Status    : N
Payment Method    : Cash
Total Payment     : RM 350.0
-----------------------------------
```

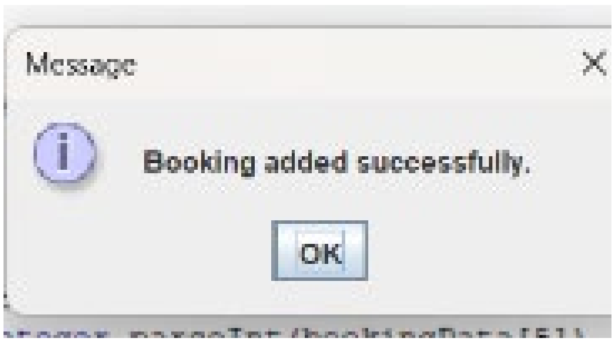
Display Calculate Total Payments (Choice 7)



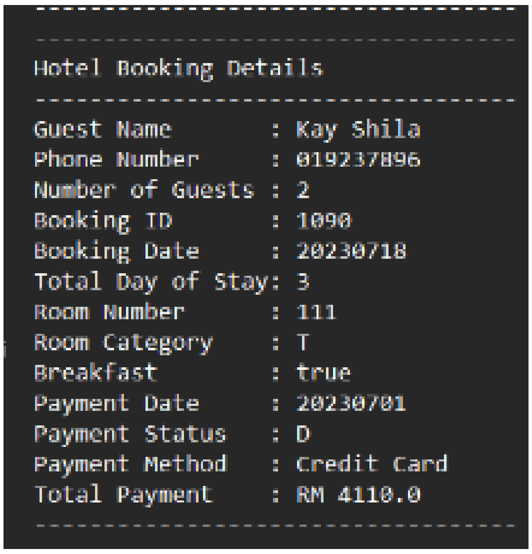
Total Payment Received (D): RM 20120
Total Payment Received (N): RM 700

<u>SAMPLE OUTPUT HOTELQUEUEMAIN</u>

Add Booking (Choice 1)

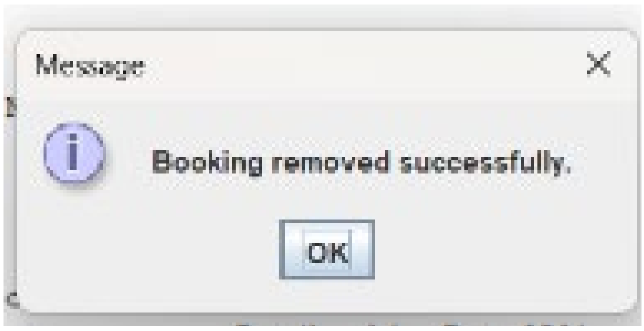


```
Message
Booking added successfully.
OK
```



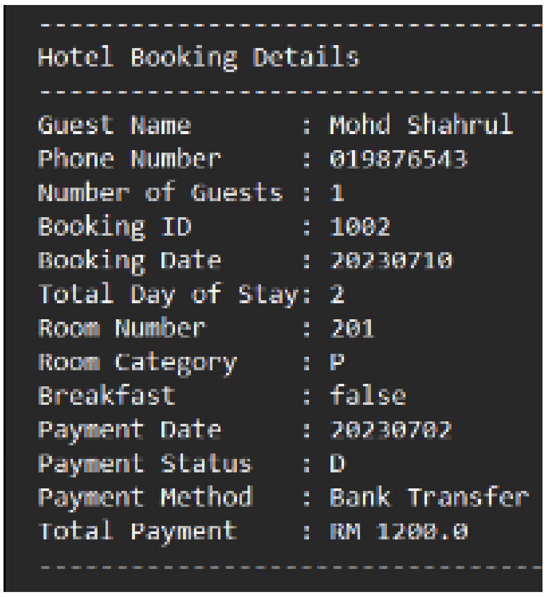
```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Kay Shila
Phone Number      : 019237896
Number of Guests : 2
Booking ID        : 1090
Booking Date      : 20230718
Total Day of Stay: 3
Room Number       : 111
Room Category     : T
Breakfast         : true
Payment Date      : 20230701
Payment Status    : D
Payment Method    : Credit Card
Total Payment     : RM 4110.0
-----------------------------------
```

Remove Booking (Choice 2)

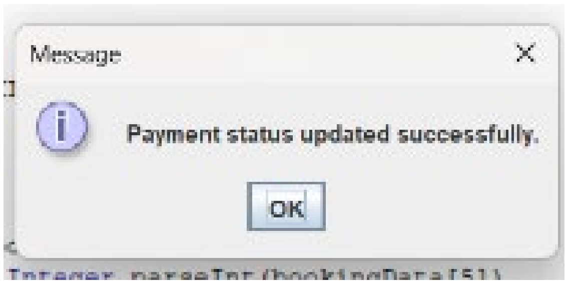


```
Message
Booking removed successfully.
OK
```

Update Payment Status (Choice 3)



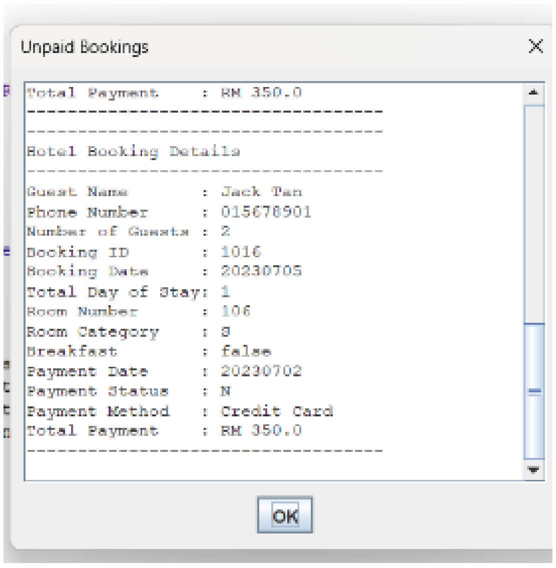
```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Mohd Shahrul
Phone Number      : 019876543
Number of Guests  : 1
Booking ID        : 1002
Booking Date      : 20230710
Total Day of Stay: 2
Room Number       : 201
Room Category     : P
Breakfast         : false
Payment Date      : 20230702
Payment Status    : D
Payment Method    : Bank Transfer
Total Payment     : RM 1200.0
-----------------------------------
```



Message

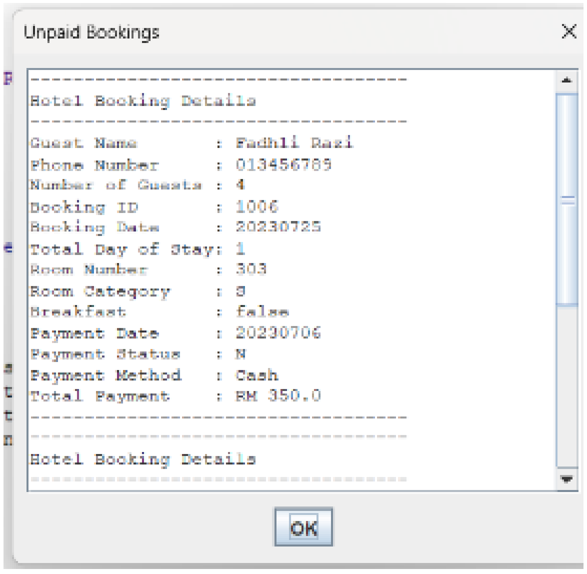
Payment status updated successfully.

Display Unpaid Bookings (Choice 4)



Unpaid Bookings

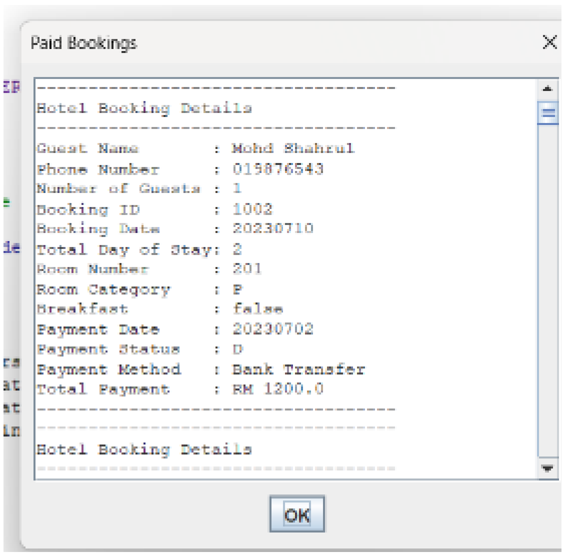
```
Total Payment     : RM 350.0
-----------------------------------
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Jack Tan
Phone Number      : 015678901
Number of Guests  : 2
Booking ID        : 1016
Booking Date      : 20230705
Total Day of Stay: 1
Room Number       : 106
Room Category     : S
Breakfast         : false
Payment Date      : 20230702
Payment Status    : N
Payment Method    : Credit Card
Total Payment     : RM 350.0
-----------------------------------
```



Unpaid Bookings

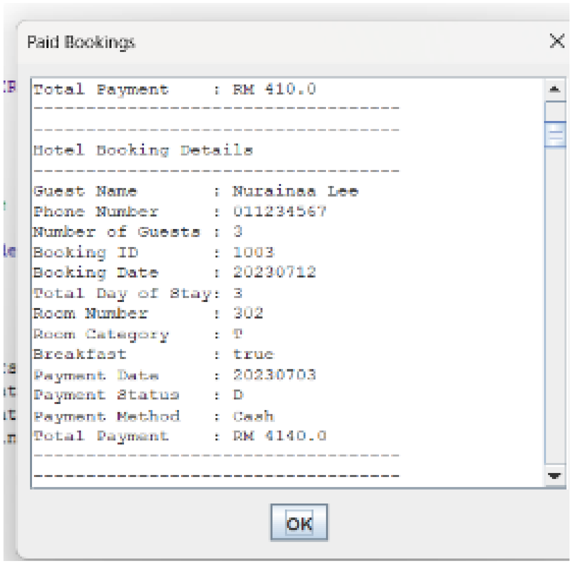
```
-----------------------------------
Hotel Booking Details
-----------------------------------
Guest Name        : Fadhli Razi
Phone Number      : 013456789
Number of Guests  : 4
Booking ID        : 1006
Booking Date      : 20230725
Total Day of Stay: 1
Room Number       : 303
Room Category     : S
Breakfast         : false
Payment Date      : 20230706
Payment Status    : N
Payment Method    : Cash
Total Payment     : RM 350.0
-----------------------------------
-----------------------------------
Hotel Booking Details
-----------------------------------
```

Display Paid Booking (Choice 5)


Paid Bookings

```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name          : Mohd Shahrul
Phone Number        : 015876543
Number of Guests    : 1
Booking ID          : 1002
Booking Date        : 20230710
Total Day of Stay: 2
Room Number         : 201
Room Category       : P
Breakfast           : false
Payment Date        : 20230702
Payment Status      : D
Payment Method      : Bank Transfer
Total Payment       : RM 1200.0
------------------------------------
------------------------------------
Hotel Booking Details
------------------------------------
```

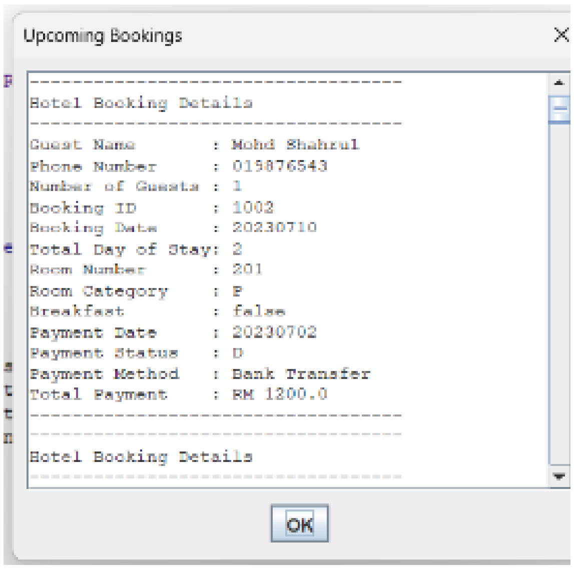
OK


Paid Bookings

```
Total Payment       : RM 410.0
------------------------------------
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name          : Nurainaa Lee
Phone Number        : 011234567
Number of Guests    : 3
Booking ID          : 1003
Booking Date        : 20230712
Total Day of Stay: 3
Room Number         : 302
Room Category       : T
Breakfast           : true
Payment Date        : 20230703
Payment Status      : D
Payment Method      : Cash
Total Payment       : RM 4140.0
------------------------------------
------------------------------------
```

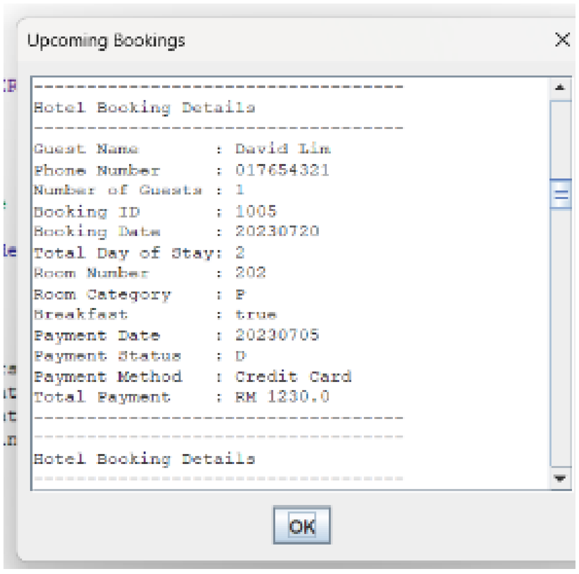
OK

Display Upcoming Bookings (Choice 6)


Upcoming Bookings

```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name          : Mohd Shahrul
Phone Number        : 015876543
Number of Guests    : 1
Booking ID          : 1002
Booking Date        : 20230710
Total Day of Stay: 2
Room Number         : 201
Room Category       : P
Breakfast           : false
Payment Date        : 20230702
Payment Status      : D
Payment Method      : Bank Transfer
Total Payment       : RM 1200.0
------------------------------------
------------------------------------
Hotel Booking Details
------------------------------------
```

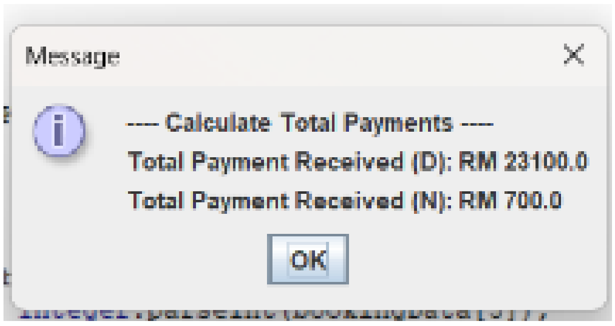
OK


Upcoming Bookings

```
------------------------------------
Hotel Booking Details
------------------------------------
Guest Name          : David Lim
Phone Number        : 017654321
Number of Guests    : 1
Booking ID          : 1005
Booking Date        : 20230720
Total Day of Stay: 2
Room Number         : 202
Room Category       : P
Breakfast           : true
Payment Date        : 20230705
Payment Status      : D
Payment Method      : Credit Card
Total Payment       : RM 1230.0
------------------------------------
------------------------------------
Hotel Booking Details
------------------------------------
```

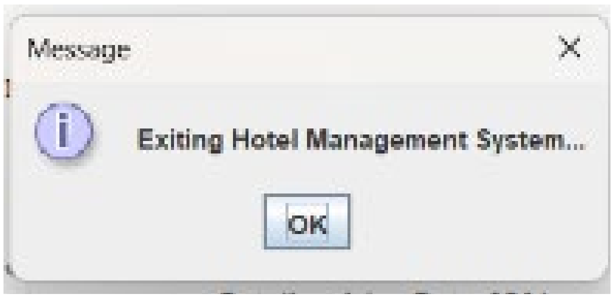
OK

Calculate Total Payments (Choice 7)



```
---- Calculate Total Payments ----
Total Payment Received (D): RM 23100.0
Total Payment Received (N): RM 700.0
```

Exit System (Choice 0)



```
Exiting Hotel Management System...
```

4.0 CONCLUSION

As a result, the linked list method is easier to use and more user-friendly compared to the queue method. When we run the program using the linked list method, there is no issues, and our data remains intact. However, if we run the program using the queue method, there might be issues, and our data could potentially be lost when we run certain choice in menu, the data in text file all lost.