



اَوْنَبُوْ سَيِّدِيْ تِيَكُوْ لُوْ كِيْ مَبَارَا
UNIVERSITI
TEKNOLOGI
MARA

Cawangan Perak
Kampus Tapah

GROUP PROJECT (FINAL REPORT)

COURSE CODE : CSC186
COURSE NAME : OBJECT ORIENTED PROGRAMMING
PROGRAM : DIPLOMA IN COMPUTER SCIENCE
CODE PROGRAM : CDCS110
GROUP NO : A4CDCSC1102A

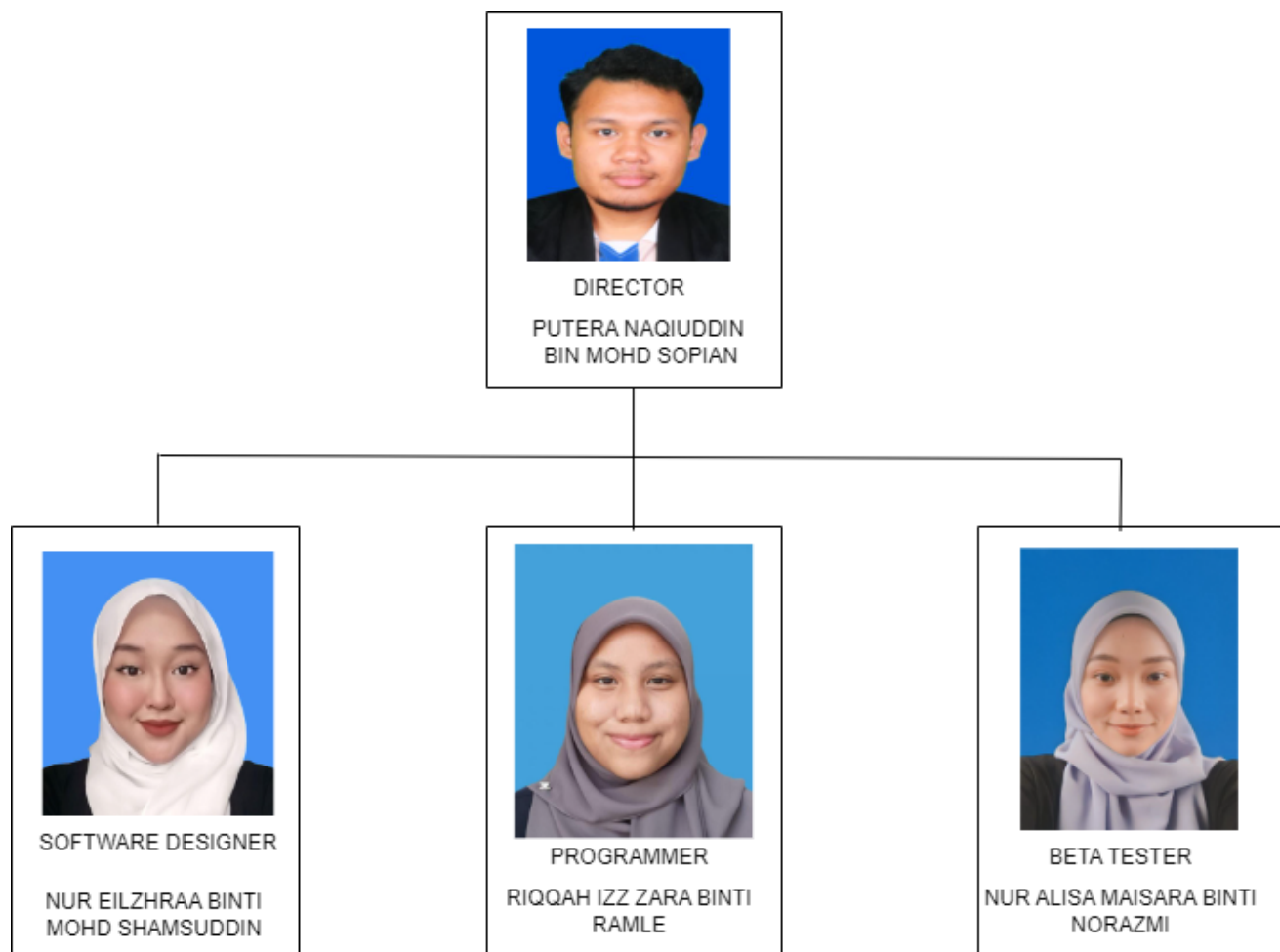
PROGRAM TITLE: CINEMA TICKET SYSTEM





Matric No.	Name
2022866226	PUTERA NAQUIDDIN BIN MOHD SOPIAN
2022802242	NUR EILZHRAA BINTI MOHD SHAMSUDDIN
2022450646	NUR ALISA MAISARA BINTI NORAZMI
2022877958	RIQQAHA IZZ ZARA BINTI RAMLE

TABLE OF CONTENT

NO	CONTENTS	PAGES
1.	ORGANIZATIONAL STRUCTURE	3
2.	INTRODUCTION	5
3.	OBJECTIVES	5
4.	SYSTEM'S SCOPE	5
5.	UML DIAGRAM	6
6.	USE CASE DIAGRM	7
7.	INPUT AND OUTPUT FILE	8
8.	CLASS DEFINITION	10
9.	CLASS DEFINITION OF INHERITENCE	29
10.	CLASS APPLICATION	36
11.	DISPLAY INFORMATION AND SAMPLE IN TERFACE	51
12.	REFERENCE	55

1.0 ORGANIZATIONAL STRUCTURE



POSITIONS	ROLES
 Director	<ul style="list-style-type: none"> - Direct the team to ensure goal are achieved without flaw. - Assist in writing program to ensure the project are within the required rubric.
 Software Designer	<ul style="list-style-type: none"> - Mainly work on software to create quality design and systems. - Ensure the software are user-friendly.
 Programmer	<ul style="list-style-type: none"> - Mainly lead in writing program based on the instruction of director. - Making sure the writing program are suitable with the software.
 Beta Tester	<ul style="list-style-type: none"> - Main focus on finding errors and flaw in writing program before the programmer proceed onto the next task. - Assist in making sure the software are user-friendly.

2.0 INTRODUCTION

Greetings from the freshly updated website from our company called AEPI Cinema Group, Cinema Ticket system is a quicker, cleaner, and slightly more personalized website that was created specifically to improve your booking experience. Our company relies on four people that are a part of this efficient system. In this system, customers can view any movie show's material whenever they want and can purchase any movie ticket as necessary. This will make it simpler for customers to buy movie tickets online without having to wait in queue for extremely long. It will also prevent customers from being unable to buy the ticket because of unforeseen circumstances. The subtotal and total are calculated automatically by the programme. For example, (**Type of ticket + Movie price + Foods (if any) – Membership (if any) = Total**). The order details, including the customer's name, and billing instructions, are securely kept in the database at the time the customer decides to book the ticket. When purchasing a ticket, the option to order a combo is also available. When you visit our website for the first time, you must register as a new user. After that, your information is permanently kept in our database, and you may use it to book movie tickets whenever you choose.

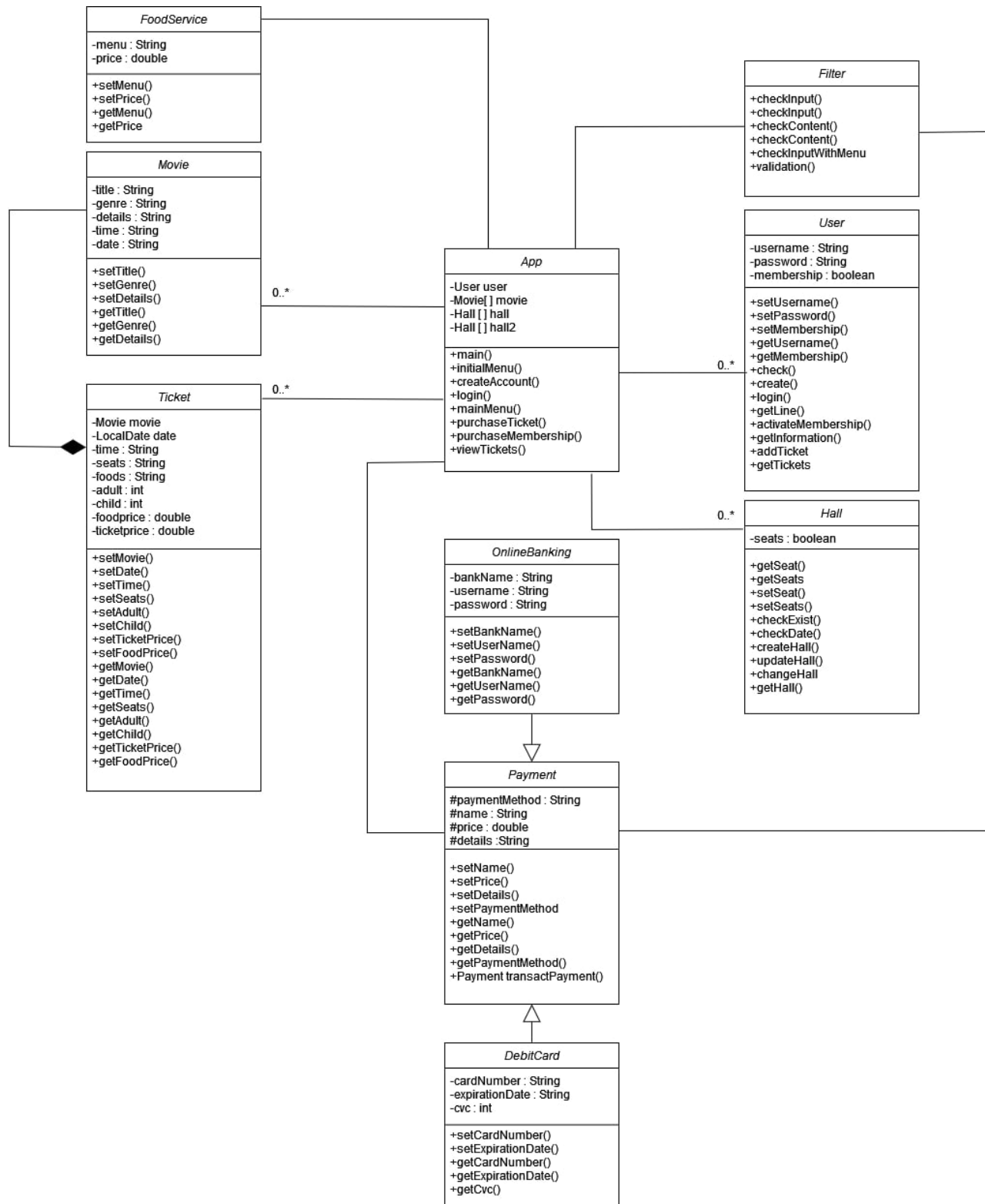
3.0 OBJECTIVES

- Can make payment in two ways which is online banking and debit card.
- Customers can get a discount when they purchase membership.
- To calculate when a customer buys a movie ticket based on the number of persons in their party, including children and adults, as well as when the customer wants to include food in their purchase.
- To print receipt when the customers purchase their order.

4.0 SYSTEM'S SCOPE

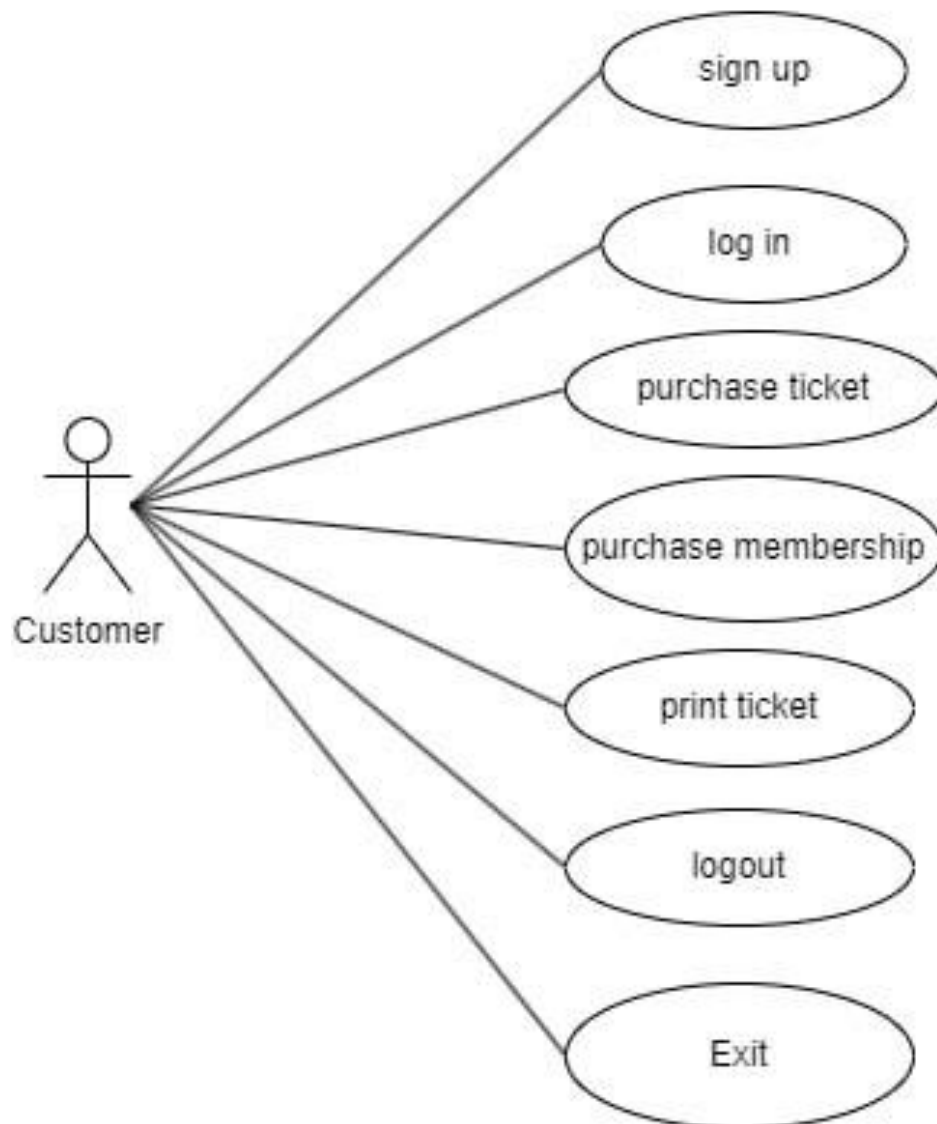
- For payment, customers can make payment in two ways which is online banking and debit card.
- Each purchase has its own purchase code, which customers must show when they go to the movie theatre.
- Customer can get 30% off when they purchased membership.

5.0 UML DIAGRAM

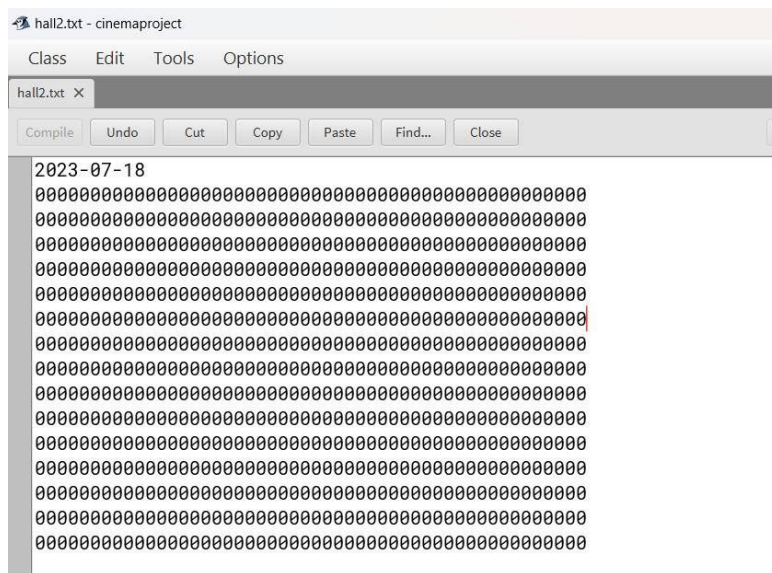
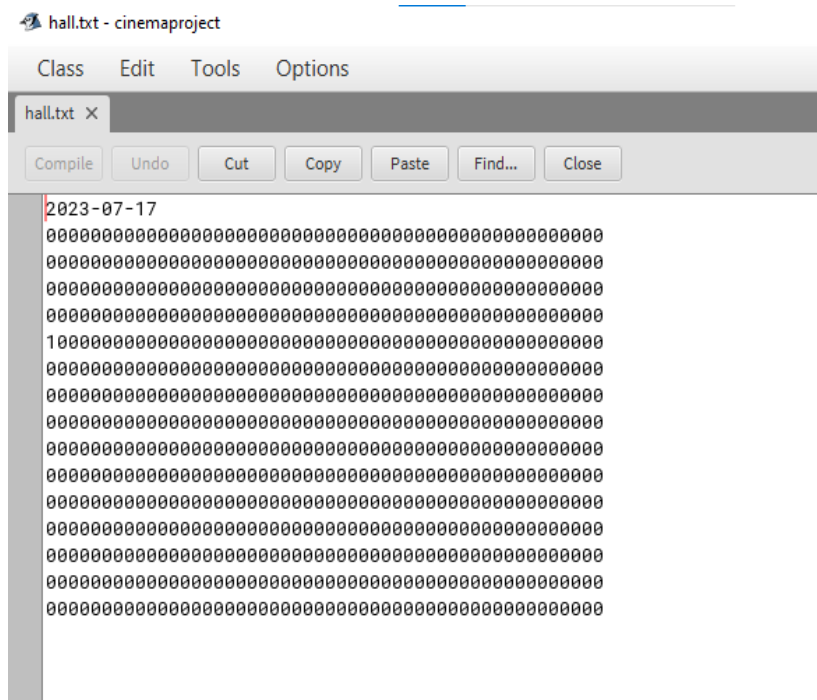


6.0 USE CASE DIAGRAM

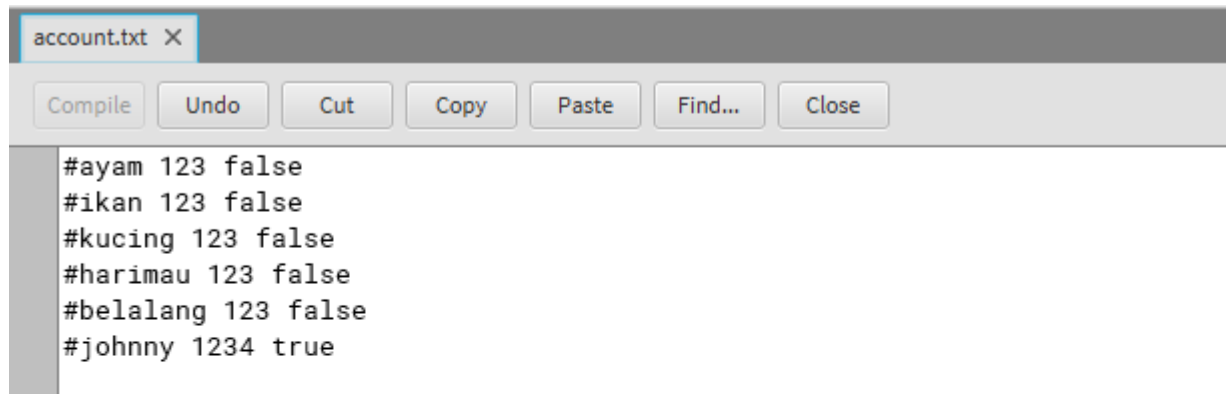
CINEMA TICKET SYSTEM



7.0 INPUT FILE

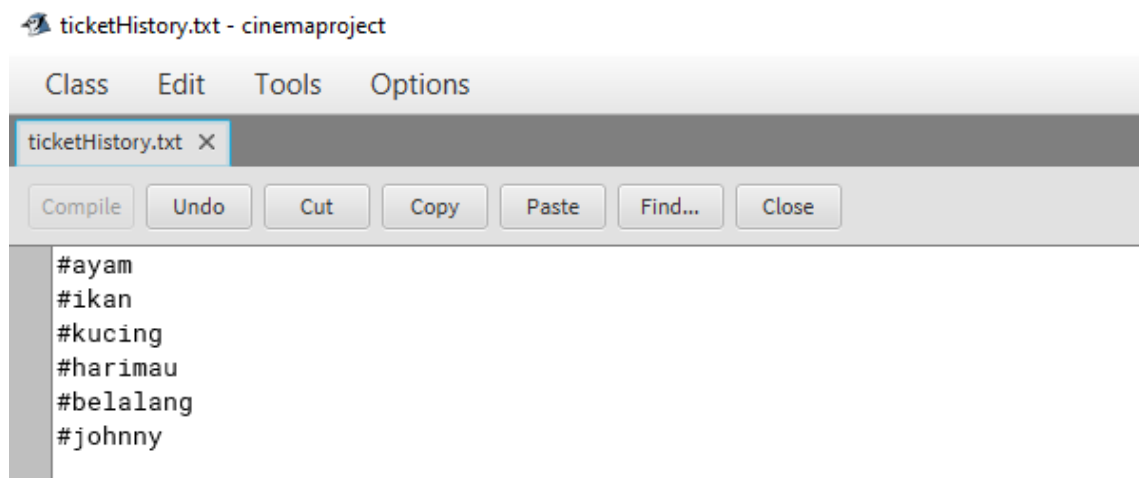


OUTPUT FILE



A screenshot of a text editor window titled "account.txt". The window has a menu bar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The text content is as follows:

```
#ayam 123 false
#ikan 123 false
#kucing 123 false
#harimau 123 false
#belalang 123 false
#johnny 1234 true
```



A screenshot of a text editor window titled "ticketHistory.txt - cinemaproject". The window has a menu bar with buttons for "Class", "Edit", "Tools", and "Options". Below the menu bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The text content is as follows:

```
#ayam
#ikan
#kucing
#harimau
#belalang
#johnny
```

8.0 CLASS DEFINITION

User Class

```
import java.io.*;
import java.util.*;
import javax.swing.*;

public class User {

    private String username, password;
    private boolean membership;

    public User() {
        username = null;
        password = null;
        membership = false;
    }

    public User(String username, String password, boolean
membership) {
        this.username = username;
        this.password = password;
        this.membership = membership;
    }

    public User(User u) {
        username = u.username;
        password = u.password;
        membership = u.membership;
    }

    public void setUsername(String username) { this.username =
username; }
    public void setPassword(String password) { this.password =
password; }
    public void setMembership(boolean membership) {
this.membership = membership; }

    public String getUsername() {return username;}
    public String getPassword() {return password;}
    public boolean getMembership() {return membership;}

    public String toString() {
        return ("Username: "+username+"\nPassword:
"+password+"\nMembership: "+membership);
    }
}
```

```

public static boolean check(String username) throws IOException {
    try {
        BufferedReader reader = new BufferedReader(new
FileReader("account.txt"));
        Boolean exist = false;
        String line = null;

        while ((line = reader.readLine()) != null){
            StringTokenizer st = new StringTokenizer(line);
            String token = null;
            if (st.hasMoreTokens()) token = st.nextToken();
else continue;
            if (token.equals("#"+username)){
                exist = true;
                break;
            }
        }
        reader.close();
        return exist;

    } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return
false;}
}

    public static void create(String username, String password)
throws IOException {

        try {
            if (User.check(username) == false){
                PrintWriter writer = new PrintWriter(new
FileWriter("account.txt", true));

                writer.println("#" +username+" "+password+"
"+false);

                writer.close();

                PrintWriter writer2 = new PrintWriter(new
FileWriter("ticketHistory.txt", true));
                writer2.println("#"+username);
                writer2.close();
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "Account
already exist");
        }
    }
}

```

```

        } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error");}
    }
    public static boolean check(String username) throws
IOException {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("account.txt"));
            Boolean exist = false;
            String line = null;

            while ((line = reader.readLine()) != null){
                StringTokenizer st = new StringTokenizer(line);
                String token = null;
                if (st.hasMoreTokens()) token = st.nextToken();
else continue;
                if (token.equals("#"+username)){
                    exist = true;
                    break;
                }
            }
            reader.close();
            return exist;

        } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return
false;}
    }

    public static void create(String username, String password)
throws IOException {

        try {
            if (User.check(username) == false){
                PrintWriter writer = new PrintWriter(new
FileWriter("account.txt", true));

                writer.println("#" +username+" "+password+"
"+false);
                writer.close();

                PrintWriter writer2 = new PrintWriter(new
FileWriter("ticketHistory.txt", true));
                writer2.println("#"+username);
                writer2.close();
            }

```

```

        else {
            JOptionPane.showMessageDialog(null, "Account
already exist");
        }
    } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error");}
    }

    public static boolean login(String username, String password)
throws IOException {

        try {
            BufferedReader reader = new BufferedReader(new
FileReader("account.txt"));
            Boolean login = false;

            if (User.check(username) == true){
                String line = null;

                while ((line = reader.readLine()) != null){
                    StringTokenizer st = new
StringTokenizer(line);
                    String token = null;
                    if (st.hasMoreTokens()) token =
st.nextToken(); else continue;
                    if (token.equals("#"+username)){
                        if (st.hasMoreTokens() &&
st.nextToken().equals(password)) {
                            login = true;
                            break;
                        } else
{JOptionPane.showMessageDialog(null, "Incorrect password");}
                    }
                }
            } else { JOptionPane.showMessageDialog(null,
"Incorrect username"); }
            reader.close();
            return login;
        } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return
false;}
    }

    public static int getLine(String username) {
        try {

```

```

        int count = 0;

        if (!User.check(username)) {
            reader.close();
            return 0;
        }

        String line;

        while ((line = reader.readLine()) != null) {
            count++;
            String[] tokens = line.split(" ");
            if (tokens.length > 0 && tokens[0].equals("#" +
username)) {
                reader.close();
                return count;
            }
        }

        reader.close();
        return 0;
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "Input Error");
        return 0;
    }
}

public static void activateMembership(String username) {
    try {
        if (!User.check(username)) {return; }
        BufferedReader reader = new BufferedReader(new
FileReader("account.txt"));
        StringBuilder sb = new StringBuilder();

        String line;

        int count = 0;
        int target = User.getLine(username);

        while ((line = reader.readLine()) != null){
            count++;
            String[] token = line.split(" ");
            if (count == target) {
                token[2] = "true";
            }
        }
    }
}

```

```

        sb.append(token[0] + " " + token[1] + " " +
token[2]).append("\n");
    }

    reader.close();

    PrintWriter writer = new PrintWriter(new
FileWriter("account.txt"));
    writer.print(sb.toString());
    writer.close();

    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "Input Error");
    }
}

public static String getInformation(String username){
    try {
        BufferedReader reader = new BufferedReader(new
FileReader("account.txt"));

        if (User.check(username) != true) {reader.close();
return null;}

        String line = null;
        int count = 0;
        int target = User.getLine(username);

        while ((line = reader.readLine()) != null){
            count++;
            if (count == target) {return line;}
        }
        reader.close();
    } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return null;}
    return null;
}

public static void addTicket(String username, String code)
throws IOException {
    try {
        if (!User.check(username)) { return; }
        BufferedReader reader = new BufferedReader(new
FileReader("ticketHistory.txt"));
    }
}

```

```

        StringBuilder sb = new StringBuilder();

        String line = null;
        int count = 0;
        int target = User.getLine(username);

        while ((line = reader.readLine()) != null){
            count++;
            if (count == target) {
                sb.append(line + " " + code + "\n");
            }
            else {
                sb.append(line + "\n");
            }
        }

        reader.close();

        PrintWriter writer2 = new PrintWriter(new
FileWriter("ticketHistory.txt"));
        writer2.print(sb.toString());
        writer2.close();

    } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return;}
    }

    public static String getTickets(String username) throws
IOException {
        try {
            if (!User.check(username)) { return null; }
            BufferedReader reader = new BufferedReader(new
FileReader("ticketHistory.txt"));

            String line = null;
            int count = 0;
            int target = User.getLine(username);

            while ((line = reader.readLine()) != null){
                count++;
                if (count == target) {return line;}
            }
            reader.close();

```



```

        } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return null;}
        return null;
    }
}

```

Hall Class

```

import java.io.*;
import java.util.*;
import java.time.*;
import javax.swing.*;
public class Hall {
    private boolean[] seats = new boolean[50];

    public Hall() {
        for (int i = 0; i < seats.length; i++){
            seats[i] = false;
        }
    }

    public Hall(boolean[] seats){
        for (int i = 0; i < this.seats.length; i++){
            this.seats[i] = seats[i];
        }
    }

    public Hall(Hall h){
        for (int i = 0; i < seats.length; i++){
            seats[i] = h.seats[i];
        }
    }

    public boolean getSeat(int i){return seats[i-1];}
    public boolean[] getSeats() {return seats;}

    public void setSeat(int i, boolean taken){seats[i-1] = taken;}
    public void setSeats(boolean[] seats){
        for (int i = 0; i < this.seats.length; i++){
            this.seats[i] = seats[i];
        }
    }

    public static boolean checkExist(String file) throws
IOException{
        try {

```

```

        BufferedReader reader = new BufferedReader(new
FileReader(file+".txt"));
        String line = reader.readLine();
        reader.close();
        if (line == null || line.isEmpty()) return false; else
return true;
    }
    catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return
false;}
    }

    public static boolean checkDate(String file) throws
IOException{
        try {
            BufferedReader reader = new BufferedReader(new
FileReader(file+".txt"));
            if (Hall.checkExist(file) != true) return false;
            String line = reader.readLine();
            String date = null;
            if (file == "hall") date = LocalDate.now().toString();
else date = LocalDate.now().plusDays(1).toString();
            return (line.equals(date));
        } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return
false;}
    }

    public static void createHall(String file) throws IOException{
        try {
            String date = null;
            if (file == "hall") date = LocalDate.now().toString();
else date = LocalDate.now().plusDays(1).toString();
            if (Hall.checkExist(file)) return;

            PrintWriter writer = new PrintWriter(new
FileWriter(file+".txt"));

            writer.println(date);
            for (int i = 0; i < 15; i++){
                String line = "";
                for (int j = 0; j < 50; j++){
                    line += "0";
                }
            }
        }
    }

```

```

        writer.println(line);
    }
    writer.close();
} catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error");}
}

    public static void updateHall(String file, boolean[] seats,
int row) throws IOException {
    try {
        if (!Hall.checkExist(file)) return;
        BufferedReader reader = new BufferedReader(new
FileReader(file+".txt"));
        String currentHall = "";
        int count = 0;
        row += 1;
        String line = null;

        while ((line = reader.readLine()) != null){
            count++;
            if (count != row) currentHall += line; else {
                String UpdatedHall = "";
                for (int i=0; i < seats.length; i++){
                    if (seats[i]) UpdatedHall += "1"; else
UpdatedHall += "0";
                }
                currentHall += UpdatedHall;
            }
            currentHall += "\n";
        }

        reader.close();
        PrintWriter writer = new PrintWriter(new
FileWriter(file+".txt"));
        writer.println(currentHall);
        writer.close();
    } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error");}
}

    public static void changeHall() throws IOException {
        try {
            String date = LocalDate.now().plusDays(1).toString();
            // ... (rest of the code)

```

```

        if (Hall.checkDate("hall")) return;

        BufferedReader reader = new BufferedReader(new
FileReader("hall2.txt"));
        String line;
        StringBuilder tempHall = new StringBuilder();

        while ((line = reader.readLine()) != null) {
            tempHall.append(line).append("\n");
        }
        reader.close();

        PrintWriter writer = new PrintWriter(new
FileWriter("hall.txt"));
        writer.println(tempHall);
        writer.close();

        writer = new PrintWriter(new FileWriter("hall2.txt"));
        writer.close();
        Hall.createHall("hall2");

    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "Input Error");
    }
}

public static boolean[] getHall(String file, int row) throws
IOException {
    try {
        boolean[] seats = new boolean[50];
        if (!Hall.checkExist(file)) return seats;
        BufferedReader reader = new BufferedReader(new
FileReader(file + ".txt"));
        int count = 0;
        row+=1;
        String output = null;
        String line = null;

        while ((line = reader.readLine()) != null){
            count++;
            if (count == row){
                output = new String(line);
            }
        }
    }
}

```

```

        reader.close();
        if (output == null) return seats;
        char[] data = output.toCharArray();
        if (data.length != 50) return seats;
        for (int i = 0; i < data.length; i++){
            if (data[i] == '1') seats[i] = true; else seats[i]
= false;
        }
        return seats;
    } catch (IOException ex)
{JOptionPane.showMessageDialog(null, "Input Error"); return new
boolean[50];}
    }
}

```

FoodService class

```

import javax.swing.*;
public class Foodservice
{
    private String menu;
    private double price;

    public Foodservice()
    {
        menu = null;
        price = 0.0;
    }

    public Foodservice(String menu,double price)
    {
        this.menu = menu;
        this.price = price;
    }

    public Foodservice(Foodservice c)
    {
        this.menu = c.menu;
        this.price = c.price;
    }

    public void setMenu(String menu){ menu = menu; }
    public void setPrice(double price) { price = price; }
}

```

```

        public String getMenu(){return menu;}
        public double getPrice(){return price;}

        public String toString(){
            return (menu +" - RM"+ price);
        }
    }
}

```

Ticket class

```

import java.time.*;

public class Ticket{
    private Movie movie;
    private LocalDate date;
    private String time, seats, foods;
    private int adult, child;
    private double foodprice, ticketprice;

    public Ticket(){
        movie = null;
        date = null; time = null;
        seats = null; foods = null;
        adult = 0; child = 0; foodprice = 0; ticketprice = 0;
    }

    public Ticket(Movie movie, LocalDate date, String time, String
seats, int adult, int child, String foods, double ticketprice,
double foodprice){
        this.movie = new Movie(movie);
        this.date = date; this.time = time;
        this.seats = seats; this.adult = adult; this.child =
child;
        this.foods = foods;
        this.ticketprice = ticketprice;
        this.foodprice = foodprice;
    }

    public Ticket(Ticket t){
        movie = new Movie(t.movie); date = t.date; time = t.time;
seats = t.seats;
        adult = t.adult; child = t.child; foods = t.foods; ticketprice
= t.ticketprice; foodprice = t.foodprice;
    }
}

```

```

        public void setMovie(Movie movie) {this.movie = new
Movie(movie);}
        public void setDate(LocalDate date) {this.date = date;}
        public void setTime(String time) {this.time = time;}
        public void setSeats(String seats) {this.seats = seats;}
        public void setAdult(int adult) {this.adult = adult;}
        public void setChild(int child) {this.child = child;}
        public void setTicketPrice(double price) {this.ticketprice =
price;}
        public void setFoodPrice(double price) {this.foodprice =
price;}

        public Movie getMovie(){return movie;}
        public LocalDate getDate(){return date;}
        public String getTime(){return time;}
        public String getSeats(){return seats;}
        public int getAdult(){return adult;}
        public int getChild(){return child;}
        public double getTicketPrice() {return ticketprice;}
        public double getFoodPrice() {return foodprice;}

        public String toString(){
            String output = "";
            output += "Movie: " + movie.getTitle() + "\n";
            output += "Date: " + date.toString() + "\n";
            output += "Time: " + time + "\n";
            output += "Seats: " + seats + "\n";
            output += "Ticket for " + adult + " adult, " + child + "
child\n";
            output += "Add-on: " +foods + "\n";
            output += "Add-on Price: RM" + foodprice;
            output += "\nTicket Price: RM" + ticketprice;
            output += "\nTotal Price: RM" + (foodprice+ticketprice);
            return output;
        }
    }

Movie class
public class Movie
{
    String title;
    String genre;

```

```

String details;
String time;
String date;
//default
public Movie(){
    title = null;
    genre = null;
}
//normal
public Movie(String m, String g, String d){
    title = m;
    genre = g;
    details = d;
}
//copy
public Movie(Movie m){
    title = m.title;
    genre = m.genre;
    details = m.details;
}
//setters
public void setTitle(String m){ title = m;}
public void setGenre(String g){ genre = g;}
public void setDetails(String d) { details = d;}

//getters
public String getTitle(){ return title; }
public String getgenre(){ return genre; }
public String getDetails() { return details; }

public String toString() {
    return ("Movie Title: [ " +title + " ]\nGenre: " + genre
+" \nDetails: "+details);
}
}

```

Filter class

```

import javax.swing.*.*;

public class Filter {
    public static double checkInput(String input,String name){
        if (name != null) {name = "("+name+"")";}
        double value = 0;
        boolean valid = false;
    }
}

```



```

        while (!valid){
            try {
                value = Double.parseDouble(input);

                valid = true;
            }
            catch (NumberFormatException ex) {
                input = JOptionPane.showInputDialog(null,"Invalid
input. Please enter a valid number
"+name,null,JOptionPane.PLAIN_MESSAGE);
            }
            catch (NullPointerException ex) {
                input = JOptionPane.showInputDialog(null,"Invalid
input. Please enter a valid number
"+name,null,JOptionPane.PLAIN_MESSAGE);
            }
            catch (IllegalArgumentException e) {
                JOptionPane.showMessageDialog(null,
e.getMessage());
                return 0;
            }
        }

        return value;
    }

    public static double checkInput(String input){
        double value = 0;
        boolean valid = false;

        while (!valid){
            try {
                value = Double.parseDouble(input);

                valid = true;
            }
            catch (NumberFormatException ex) {
                input = JOptionPane.showInputDialog(null,"Invalid
input. Please enter a valid
number",null,JOptionPane.PLAIN_MESSAGE);
            }
            catch (NullPointerException ex) {

```

```

        input = JOptionPane.showInputDialog(null,"Invalid input.
Please enter a valid number ",null,JOptionPane.PLAIN_MESSAGE);
    }
    catch (IllegalArgumentException e) {
        JOptionPane.showMessageDialog(null, e.getMessage());
        return 0;
    }
}

return value;
}

public static String checkContent(String args){
    boolean valid = true;
    do {
        valid = true;
        if (args == null) return args;
        if (args.isEmpty()) {valid = false;}
        if (args.matches(".*[ \\]\\].*")) {valid = false;}

        if (!valid) {
            args = JOptionPane.showInputDialog(null,"Please
enter input without space or backslash\nPress cancel to return
instead of leaving input empty",null,JOptionPane.PLAIN_MESSAGE);
        }
    } while (!valid);

    return args;
}

public static String checkContent(String args, String name){
    boolean valid = true;
    do {
        valid = true;
        if (args == null) return args;
        if (args.isEmpty()) {valid = false;}
        if (args.matches(".*[ \\]\\].*")) {valid = false;}

        if (!valid) {
            args = JOptionPane.showInputDialog(null,"("+name+")

```

```

Please enter input without space or backslash\nPress cancel to
return instead of leaving input
empty",null,JOptionPane.PLAIN_MESSAGE);
        if (args == null) {valid = true;}
    }
    } while (!valid);

    return args;
}

public static double checkInputWithMenu(String input,String
menu){
    double value = 0;
    boolean valid = false;

    while (!valid){
        try {
            value = Double.parseDouble(input);

            valid = true;
        }
        catch (NumberFormatException ex) {
            if (input.equalsIgnoreCase("done")) return -556;
            input =
JOptionPane.showInputDialog(null,menu+"\nInvalid input. Please
enter a valid number",null,JOptionPane.PLAIN_MESSAGE);
        }
        catch (NullPointerException ex) {
            return 0;
        }
        catch (IllegalArgumentException e) {
            JOptionPane.showMessageDialog(null,
e.getMessage());
            return 0;
        }
    }

    return value;
}

```

```

public static String validation(String args, String name){
    boolean valid = true;
    do {
        valid = true;
        if (args == null) return args;
        if (args.isEmpty()) {valid = false;}

        if (!valid) {
            args =
JOptionPane.showInputDialog(null, "("+name+") Please enter
appropriate input", null, JOptionPane.PLAIN_MESSAGE);
            if (args == null) {valid = true;}
        }
    } while (!valid);

    return args;
}
}

```

9.0 CLASS DEFINITION OF INHERITANCE

```
import javax.swing.*;

public class Payment
{
    protected String paymentMethod;
    protected String name;
    protected double price;
    protected String details;

    public Payment()
    {
        name = null;
        paymentMethod = null;
        price = 0;
        details = null;
    }

    public Payment(String name, String method, double price,
String details){
        this.name = name;
        this.paymentMethod = method;
        this.price = price;
        this.details = details;
    }

    public Payment(Payment p){
        name = p.name;
        paymentMethod = p.paymentMethod;
        price = p.price;
    }
}
```

```

        details = p.details;
    }

    public void setName(String name) {this.name = name;}
    public void setPrice(double price) {this.price = price;}
    public void setDetails(String details) {this.details =
details;}

    public void setPaymentMethod(String
paymentMethod){this.paymentMethod = paymentMethod;}

    public String getName() {return name;}
    public double getPrice() {return price;}
    public String getDetails() {return details;}
    public String getPaymentMethod() {return paymentMethod;}

    public String toString()
    {   String output = "====Payment processed
successfully!====";
        output += "\nCustomer's Name : " + name;
        output += "\nDetails: " + details;
        output += "\nTotal Price:RM " + price;
        return output;
    }

    public static Payment transactPayment(String info, double
price){
        boolean complete = false;
        String[] options2 = {"Online Banking", "Debit Card",
"Cancel"};
        Payment payment = null;

        int choice = JOptionPane.showOptionDialog(
            null,

```

```

        "[ CINEMA TICKET SYSTEM ]\nChoose payment method",
        "Payment Method",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.PLAIN_MESSAGE,
        null,
        options2,
        options2[0]);

    String name;
    switch (choice){
        case -1:
            break;
        case 0:
            name =
Filter.validation(JOptionPane.showInputDialog("Enter your
name"), "Name");
            if (name == null) break;
            String bank =
Filter.validation(JOptionPane.showInputDialog("Enter bank name"),
"Bank Name");
            if (bank == null) break;
            String username =
Filter.checkContent(JOptionPane.showInputDialog("Enter
username"), "Username");
            if (username == null) break;
            String password =
Filter.checkContent(JOptionPane.showInputDialog("Enter
password"), "Username");
            if (password == null) break;
            payment = new OnlineBanking(name, options2[0],
price, info, bank, username, password);
            break;

```

```

        case 1:
            name =
Filter.validation(JOptionPane.showInputDialog("Enter your name"),
"Name");

            if (name == null) break;
            String cardNum =
Filter.validation(JOptionPane.showInputDialog("Enter card
number"), "Card Number");

            if (cardNum == null) break;
            String expDate =
Filter.validation(JOptionPane.showInputDialog("Enter card
expiration date"), "Expiration Date");

            if (expDate == null) break;
            int cvc =
(int)Filter.checkInput(JOptionPane.showInputDialog("Enter card
cvc"));

            if (cvc == 0) break;
            payment = new DebitCard(name, options2[1], price,
info, cardNum, expDate, cvc);
            break;
        case 2:
            break;
        default:
            break;
    }
    return payment;
}

}

//Online banking class
public class OnlineBanking extends Payment
{
    private String bankName;
    private String username;

```



```

        private String password;
        public OnlineBanking()
        {
            super();
            this.bankName = null;
            this.username = null;
            this.password = null;
        }
        public OnlineBanking(String name, String method, double
price, String details,String bankName,String username, String
password)
        {
            super(name,method,price,details);
            this.bankName = bankName;
            this.username = username;
            this.password = password;
        }
        public OnlineBanking(OnlineBanking bank)
        {
            super(bank);
            bankName = bank.bankName;
            username = bank.username;
            password = bank.password;
        }

        public void setBankName (String bName){bankName = bName;}
        public void setUsername (String uName){username = uName;}
        public void setPassword (String pass) {password = pass;}

        public String getBankName() { return bankName;}
        public String getUsername(){return username;}
        public String getPassword(){return password;}

```

```

        public String toString() {
            String output = "====Payment processed
successfully!====";
            output += "\nCustomer's Name : " + name;
            output += "\nDetails: " + details;
            output += "\nTotal Price: RM " + price;
            output += "\nPayment Method: " + paymentMethod;
            output += "\nPaid with Bank: " + bankName;
            return output;
        }
    }

    //DebitCard class
    public class DebitCard extends Payment
    {
        private String cardNumber;
        private String expirationDate;
        private int cvc;

        public DebitCard(){
            super();
            cardNumber = null;
            expirationDate = null;
            cvc = 0;
        }

        public DebitCard(String name, String method, double price,
String details, String cardNumber, String expirationDate, int cvc)
        {
            super(name, method, price, details);
            this.cardNumber = cardNumber;
            this.expirationDate = expirationDate;

```

```

        this.cvc = cvc;
    }

    public DebitCard(DebitCard card){
        super(card);
        cardNumber = card.cardNumber;
        expirationDate = card.expirationDate;
        cvc = card.cvc;
    }

    public void setCardNumber(String num) {cardNumber = num;}
    public void setExpirationDate(String date)
{expirationDate = date;}
    public void setCvc(int cvc) {this.cvc = cvc;}

    public String getCardNumber(){return cardNumber;}
    public String getExpirationDate(){return expirationDate;}
    public int getCvc() {return cvc;}

    public String toString()
    {
        String output = "====Payment processed
successfully!====";
        output += "\nCustomer's Name : " + name;
        output += "\nDetails: " + details;
        output += "\nTotal Price:RM " + price;
        output += "\nPayment Method: " + paymentMethod;
        return output;
    }

}

```

10.0 CLASS APPLICATION

```
import javax.swing.*;
import java.io.*;
import java.time.*;
import java.util.*;

public class App {
    private static User user = new User();
    private static Movie[] movie = new Movie[5];
    private static Hall[] hall = new Hall[15];
    private static Hall[] hall2 = new Hall[15];

    public static void main(String[] args) throws
IOException{
        //initialize stuffs
        movie[0] = new Movie("EL CAMINO","Western, Crime
film, Drama, Crime-Drama",
        "After the events captured in the finale of Breaking
Bad, Jesse is now on the run, and a massive police manhunt for him
is in operation.\n" +
        "He has a plan to get out of Albuquerque, but he'll
need heaps of cash to do it. Luckily, he knows where some may be
stashed.");
        movie[1] = new Movie("POLIS EVO","Action, Comedy,
Adventure, Crime film",
        "Big city cop and small-town-big-cop must learn to
work together in order to take down Malaysia's biggest drug
lord.");
        movie[2] = new Movie("JURASSIC WORLD DOMINION",
"Action, Adventure, Science fiction, Monster",
        "Four years after the destruction of Isla Nublar,
Biosyn
```

```

        operatives attempt to track down Maisie Lockwood,\n"+
        "while Dr Ellie Sattler investigates a genetically
engineered swarm of giant insects.");

        movie[3] = new Movie("SUZUME","Anime, Science
fiction, Adventure, Animation",
        "A modern action adventure road story where a 17-
year-old girl named Suzume" +
        " helps a mysterious young man\nclose doors from the
other side that are releasing disasters all over in Japan");

        movie[4] = new Movie("ALITA: BATTLE ANGEL","Action,
Science fiction, Cyberpunk, Romance",
        "In a bleak world, Dr. Ido finds Alita, a deadly
cyborg with no memory."+
        "\nAs the only hope for ending destruction, Alita
embraces her dual nature: an angel from heaven and an angel of
death.");

        Hall.createHall("hall");
        Hall.createHall("hall2");
        Hall.changeHall();

        for (int i = 0; i < 15; i++){
            hall[i] = new Hall(Hall.getHall("hall",i+1));
            hall2[i] = new Hall(Hall.getHall("hall2",i+1));
        }
        initialMenu();
    }

    public static void initialMenu() throws IOException {
        createAccount();
    }

```

```

        if (login() != true) {initialMenu(); return; }

        JOptionPane.showMessageDialog(null,"Welcome, User
        "+user.getUsername());

        mainMenu();
    }

    public static void createAccount() throws IOException {
        int choice = JOptionPane.showOptionDialog(null,
            "Do you want to create new account?\nPress NO
to immediately login", "Confirmation",
            JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE,null,null,null);
        if (choice == 0){
            String username =
JOptionPane.showInputDialog(null,"Enter username","Account
creation", JOptionPane.PLAIN_MESSAGE);
            username =
Filter.checkContent(username,"Username");
            if (username == null) { createAccount(); return;}

String password = JOptionPane.showInputDialog(null,"Enter
password","Account creation", JOptionPane.PLAIN_MESSAGE);
            password = Filter.checkContent(password,"Password");
            if (password == null) { createAccount(); return;}

            User.create(username, password);
        } else if (choice == -1) {System.exit(0);}

    }

```

```

    public static boolean login() throws IOException {
        boolean login = false;
        String username;
        String password;
        do {
            username = JOptionPane.showInputDialog(null, "Enter
username", "Login", JOptionPane.PLAIN_MESSAGE);
            username = Filter.checkContent(username, "Username");
            if (username == null) {
JOptionPane.showMessageDialog(null, "Action canceled"); break;}
            password =
JOptionPane.showInputDialog(null, "Enter password", "Login",
JOptionPane.PLAIN_MESSAGE);
            password =
Filter.checkContent(password, "Password");
            if (password == null) {
JOptionPane.showMessageDialog(null, "Action canceled"); break;}
            login = User.login(username, password);
        }while (!login);
        if (login) {
            String info = User.getInformation(username);
            if (info == null) {return false;}
            String[] infos = info.split(" ");
            user = new User(username, infos[1],
Boolean.parseBoolean(infos[2]));
        }
        return login;
    }

    public static void mainMenu() throws IOException {

        String output = "[ CINEMA TICKET SYSTEM ]";
        output += "\n\n1. <Purchase ticket>";
    }

```

```

        output += "\n2. <Purchase membership>";
        output += "\n3. <Log out>";
        output += "\n4. <Exit>";

        String[] options = { "Purchase Ticket", "Purchase
Membership", "View Tickets" , "Log out", "Exit" };

        int choice = JOptionPane.showOptionDialog(
            null,
            "[ CINEMA TICKET SYSTEM ]\nChoose an
option:",
            "Main Menu",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.PLAIN_MESSAGE,
            null,
            options,
            options[0]
        );
        switch (choice) {
            case 0:
                purchaseTicket();
                break;
            case 1:
                purchaseMembership();
                break;
            case 2:
                viewTickets();
                break;
            case 3:
                JOptionPane.showMessageDialog(null, "Log out
succeed. Heading to Initial Menu");
                initialMenu();
        }
    }
}

```



```

        break;

        case 4:
            break;
        default:
            mainMenu();
            break;
    }
    return;
}

public static void purchaseTicket() throws IOException{
    String[] options = new String[6];
    for (int i = 0; i < movie.length; i++){
        options[i] = movie[i].getTitle();
    }
    options[5] = "Back";

    int choice = JOptionPane.showOptionDialog(
        null,
        "[ CINEMA TICKET SYSTEM ]\nSelect a movie:",
        "Ticket Purchase",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.PLAIN_MESSAGE,
        null,
        options,
        options[0]);
    if (choice == -1 || choice == 5) {mainMenu();
return;}

        String[] options2 = { LocalDate.now().toString(),
LocalDate.now().plusDays(1).toString(), "Back" };
        LocalDate[] dates = { LocalDate.now(),
LocalDate.now().plusDays(1)};

```

```

        int choice2 = JOptionPane.showOptionDialog(
            null,
            "[ CINEMA TICKET SYSTEM
]\n"+movie[choice].toString(),
            "Ticket Purchase",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.PLAIN_MESSAGE,
            null,
            options2,
            options2[0]);

        if ( choice2 == -1 ) {mainMenu(); return;}
        if ( choice2 == 2 ) {purchaseTicket(); return;}

        String[] options3 = { "11:00 AM", "01:00 PM", "03:00
PM", "Back" };
        int[] options3_int = { 11, 13, 15 };

        int choice3 = JOptionPane.showOptionDialog(
            null,
            "[ CINEMA TICKET SYSTEM
]\n"+movie[choice].toString(),
            "Ticket Purchase",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.PLAIN_MESSAGE,
            null,
            options3,
            options3[0]);

        if (choice3 == -1) {mainMenu(); return;}
        if (choice3 == 3) {purchaseTicket(); return;}
        int rowHall = (choice3 + 1) + (choice*3);

```

```

        boolean[] seats = new boolean[50];
        boolean[] hallSeats = hall[rowHall-1].getSeats();
        boolean[] hall2Seats = hall2[rowHall-1].getSeats();

        if (choice2 == 0) {
            for (int i = 0; i < hallSeats.length; i++){
                seats[i] = hallSeats[i];
            }
        } else {
            for (int i = 0; i < hallSeats.length; i++){
                seats[i] = hall2Seats[i];
            }
        }
        String selectedSeat = null;
        int intSeat = 0;
        int totalseat = 0;
        String seatDetails = "";

        do {
            String output = "";
            selectedSeat = null;
            intSeat = 0;
            for (int i = 0; i < seats.length; i++){
                if (!seats[i]){
                    if (i<9) output += "[ S0" +(i+1)+" ]";
else output += "[ S" +(i+1)+" ]";
                }
                else {
                    output += "[--- ---]";
                }
                if (i%10 == 9) output += "\n";
            }
        }

```

```

        selectedSeat = JOptionPane.showInputDialog(null,"[ CINEMA
TICKET SYSTEM ]\nEnter number to choose seat:\nType \"done\" to
proceed\n"+output+"\nNOTE: Seats are facing
downward\n",null,JOptionPane.PLAIN_MESSAGE);

        if (selectedSeat != null) {if
(selectedSeat.equalsIgnoreCase("done")) break; }

        intSeat =
(int)Filter.checkInputWithMenu(selectedSeat,"[ CINEMA TICKET
SYSTEM ]\nEnter number to choose seat:\nType \"done\" to
proceed\n"+output +"\nNOTE: Seats are facing downward\n");

        if (intSeat > 0 && seats[intSeat-1]) continue;
        if (intSeat > 0 && intSeat <= 50){
            seats[intSeat-1] = true;
            totalseat++;
            seatDetails += " S"+intSeat;
        }
    } while (selectedSeat != null && intSeat > 0 &&
intSeat <= 50 );

    if (totalseat == 0)
{JOptionPane.showMessageDialog(null,"Invalid response, you didn't
select a seat!");}

    if (intSeat == -556 &&
(selectedSeat.equalsIgnoreCase("done") == false)) {selectedSeat =
"done";}

    if (selectedSeat == null (intSeat <= 0 &&
(selectedSeat.equalsIgnoreCase("done") == false)) intSeat > 50 ||
totalseat == 0 ) {mainMenu(); return;}

    int adult = 0;
    int child = 0;
    do {

```

```

        adult = (int)
Filter.checkInput(JOptionPane.showInputDialog(null,"Please enter
ticket amount for adult(RM12)", "Ticket
Amount",JOptionPane.PLAIN_MESSAGE),"Adult ticket");

        child = (int)
Filter.checkInput(JOptionPane.showInputDialog(null,"Please enter
ticket amount for child(RM8)", "Ticket
Amount",JOptionPane.PLAIN_MESSAGE),"Child ticket");

        if ((adult+child)!= totalseat)
JOptionPane.showMessageDialog(null,"You got "+totalseat+" ticket
selected! Please enter appropriately");

        } while ((adult+child)!= totalseat || (adult == 0 &&
child == 0));

        if (adult == 0 && child == 0){mainMenu(); return;}
        double ticketPrice = adult*12 + child*8;
        if (user.getMembership()) ticketPrice -=
ticketPrice*.3;

        String[] name = {"Caramel popcorn", "Regular popcorn",
"Salted popcorn", "Cheezy Nachos", "Cheezy Sausage",
                        "Coca-cola", "Sprite", "Fanta Grape",
"Milo", "Mineral water"};

        double[] price = {10, 8, 8, 10, 10,
                        6, 6, 6, 5, 3};

        Foodservice[] menu = new Foodservice[10];
        String menuOutput = "[ CINEMA TICKET SYSTEM ]\nEnter
number to choose add-on\nType \"done\" to proceed\n";

        for (int i = 0; i < menu.length; i++){
            menu[i] = new Foodservice(name[i], price[i]);
            menuOutput += (i+1) + ". " + menu[i].toString()
+"\n";
        }

```

```

double itemPrice = 0;
    String selectedItem = null;
    String itemDetails = "";
    int intItem = 0;

    do {
        selectedItem = null;
        intItem = 0;
        selectedItem = JOptionPane.showInputDialog(null,
menuOutput, "Item Add-on",JOptionPane.PLAIN_MESSAGE);
        if (selectedItem != null &&
selectedItem.equalsIgnoreCase("done")) break;
        intItem = (int)
Filter.checkInputWithMenu(selectedItem, menuOutput);
        if (intItem > 0 && intItem <= 10){
            if (!itemDetails.isEmpty()) itemDetails +=
"\n        ";
            itemPrice += menu[intItem-1].getPrice();
            itemDetails += "*" + menu[intItem-1].getMenu();
        }
    } while (selectedItem != null  intItem <= 0  intSeat
> 10);

    if (intItem == -556 &&
(selectedItem.equalsIgnoreCase("done")== false)){selectedItem =
"done";}

    if (selectedItem == null  ( intItem <= 0 &&
(selectedItem.equalsIgnoreCase("done") == false )  intItem > 10))
{mainMenu(); return;}

    if (user.getMembership()) itemPrice -= itemPrice*.3;
    LocalDate date;
    if (choice2 == 0) date = LocalDate.now(); else date =

```

```

        LocalDate.now().plusDays(1);

        Ticket ticket = new
Ticket(movie[choice],date,options3[choice3],seatDetails,adult,chil
d,itemDetails,ticketPrice,itemPrice);

        String[] confirmation = {"Confirm", "Cancel"};

        int accept = JOptionPane.showOptionDialog(
            null,
            "[ CINEMA TICKET SYSTEM]\n" +
ticket.toString(),
            "Ticket Purchase Confirmation",
            JOptionPane.DEFAULT_OPTION,
            JOptionPane.PLAIN_MESSAGE,
            null,
            confirmation,
            confirmation[0]);

        if (accept != 0) {mainMenu(); return;}
        Payment payment = null;
        payment = payment.transactPayment("Ticket
Purchasement",ticketPrice+itemPrice);
        if (payment == null) {mainMenu(); return;}

        if (choice2 == 0) {
            hall[rowHall-1].setSeats(seats);
            Hall.updateHall("hall",seats,rowHall);
        } else {
            hall2[rowHall-1].setSeats(seats);
            Hall.updateHall("hall2",seats,rowHall);
        }
    }

```

```

        int length = 8;

        Random random = new Random();

        StringBuilder codeBuilder = new StringBuilder();
        for (int i = 0; i < length; i++) {
            boolean isDigit = random.nextBoolean();

            if (isDigit) {
                int digit = random.nextInt(10);
                codeBuilder.append(digit);
            } else {
                char character = (char) (random.nextInt(26) +
'A');

                codeBuilder.append(character);
            }
        }

        String code = codeBuilder.toString();
        JOptionPane.showMessageDialog(null, payment.toString());
        JOptionPane.showMessageDialog(null, "[ CINEMA TICKET
SYSTEM]\n" + ticket.toString() + "\nCode: " + code);

        User.addTicket(user.getUsername(), code);

        mainMenu();
        return;
    }

    public static void purchaseMembership() throws
IOException{
        String[] options = {"Purchase", "Cancel"};
        int choice = JOptionPane.showOptionDialog(

```



```

        null,

        "[ CINEMA TICKET SYSTEM ]\nDetails:
Membership Purchasement\nPurchasing membership will award you with
30% discount on total price!\nCost: RM30.00\n",

        "Membership Purchase",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.PLAIN_MESSAGE,
        null,
        options,
        options[0]);

        if (choice != 0){mainMenu(); return;}
        if (user.getMembership()) {
JOptionPane.showMessageDialog(null, "You already have
membership!"); mainMenu(); return;}

        Payment payment = null;
        payment = payment.transactPayment("Membership
Purchasement", 30);

        if (payment == null)
JOptionPane.showMessageDialog(null, "Purchase canceled");
        else {

JOptionPane.showMessageDialog(null, payment.toString());

        User.activateMembership(user.getUsername());
        user.setMembership(true);
        }
        mainMenu();
        return;
    }

    public static void viewTickets() throws IOException{
        String output = "";

```

```

        output = "[ CINEMA TICKET SYSTEM ]\n" + user.getUsername() +
" Ticket Code History\n\n";

        try {
            String[] tickets =
user.getTickets(user.getUsername()).split(" ");
            for (int i = 1; i < tickets.length; i++){
                output += tickets[i] + "\n";
            }

            if (tickets.length == 1) output += "No ticket
bought\n";

        } catch (NullPointerException ex) {
            output += "No ticket bought\n";
        }

        JOptionPane.showMessageDialog(null, output);

        mainMenu();
        return;
    }
}

```

11.0 DISPLAY INFORMATION AND SAMPLE INTERFACE

1. System log in

Customer who already have an account can continue straight to log in process, while for those who don't have it, they need to sign up first.

The image shows three sample interface windows for the login process. The first window, titled 'Confirmation', contains a green question mark icon and the text 'Do you want to create new account? Press NO to immediately login'. It has two buttons: 'Yes' and 'No'. The second window, titled 'Login', contains the text 'Enter username' and a text input field, with 'OK' and 'Cancel' buttons below it. The third window, also titled 'Login', contains the text 'Enter password' and a text input field, with 'OK' and 'Cancel' buttons below it.

2. Main Menu

Customers will be given an option whether they want to purchase a ticket or to purchase membership.

The image shows a 'Main Menu' window titled '[CINEMA TICKET SYSTEM]'. It contains the text 'Choose an option:' followed by five buttons: 'Purchase Ticket', 'Purchase Membership', 'View Tickets', 'Log out', and 'Exit'.

3. Purchase Ticket

The customer will be given various of movie's choices to choose from and which date and time they want to watch the movie.

The image shows two sample interface windows for the ticket purchase process. The first window, titled 'Ticket Purchase', contains the text '[CINEMA TICKET SYSTEM]' and 'Select a movie:'. It has six buttons: 'EL CAMINO', 'POLIS EVO', 'JURASSIC WORLD DOMINION', 'SUZUME', 'ALITA: BATTLE ANGEL', and 'Back'. The second window, also titled 'Ticket Purchase', contains the text '[CINEMA TICKET SYSTEM]', 'Movie Title: [POLIS EVO]', 'Genre: Action, Comedy, Adventure, Crime film', and 'Details: Big city cop and small-town-big-cop must learn to work together in order to take down Malaysia's biggest drug lord.' It has three buttons: '2023-07-20', '2023-07-21', and 'Back'.

Ticket Purchase

[CINEMA TICKET SYSTEM]
Movie Title: [POLIS EVO]
Genre: Action, Comedy, Adventure, Crime film
Details: Big city cop and small-town-big-cop must learn to work together in order to take down Malaysia's biggest drug lord.

11:00 AM
01:00 PM
03:00 PM
Back

4. Seat, Ticket amount and Add-On

Customer needs to choose their seat and the amount of ticket that they want to purchase for adult and children. The customer also can add-on any foods if they want and the system will calculate the total.

[CINEMA TICKET SYSTEM]
Enter number to choose seat:
Type "done" to proceed
[S01][S02][S03][S04][S05][S06][S07][S08][S09][S10]
[S11][S12][S13][S14][S15][S16][S17][S18][S19][S20]
[S21][S22][S23][S24][S25][S26][S27][S28][S29][S30]
[S31][S32][S33][S34][S35][S36][S37][S38][S39][S40]
[S41][S42][S43][S44][S45][S46][S47][S48][S49][S50]
NOTE: Seats are facing downward

OK Cancel

Ticket Amount

Please enter ticket amount for adult(RM12)

OK Cancel

Ticket Amount

Please enter ticket amount for child(RM8)

OK Cancel

Item Add-on

[CINEMA TICKET SYSTEM]
Enter number to choose add-on
Type "done" to proceed
1. Caramel popcorn - RM10.0
2. Regular popcorn - RM8.0
3. Salted popcorn - RM8.0
4. Cheezy Nachos - RM10.0
5. Cheezy Sausage - RM10.0
6. Coca-cola - RM6.0
7. Sprite - RM6.0
8. Fanta Grape - RM6.0
9. Milo - RM5.0
10. Mineral water - RM3.0

OK Cancel

Ticket Purchase Confirmation

[CINEMA TICKET SYSTEM]
Movie: POLIS EVO
Date: 2023-07-20
Time: 11:00 AM
Seats: S1
Ticket for 1 adult, 0 child
Add-on: *Cheezy Sausage
Add-on Price: RM10.0
Ticket Price: RM12.0
Total Price: RM22.0
Confirm Cancel

5. Payment Method (Online Banking)

Customers need to choose whether they want to choose payment method using online banking or debit card. For online banking, they need to enter their name, password and their bank's name as to log in to the bank to make payment.

Payment Method

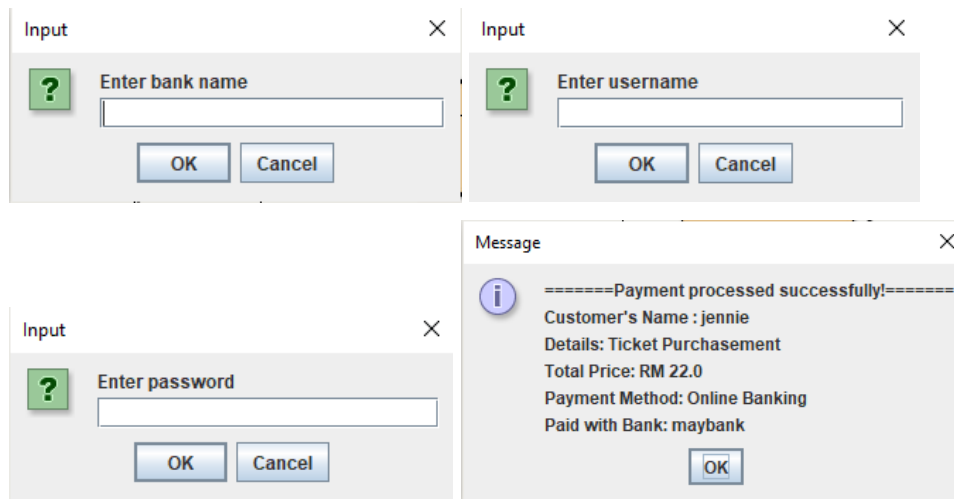
[CINEMA TICKET SYSTEM]
Choose payment method
Online Banking
Debit Card
Cancel

Input

? Enter your name

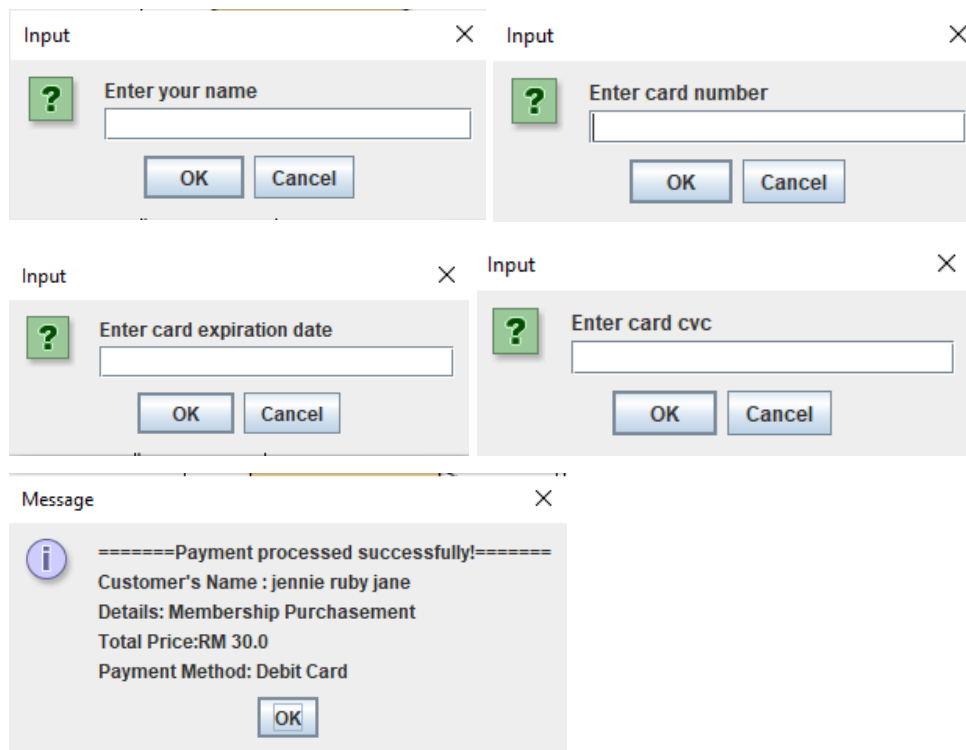
OK Cancel

52



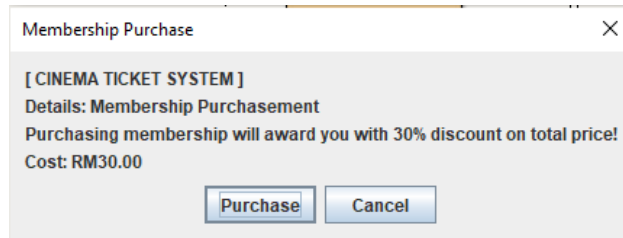
Debit Card

Customers also can pay using debit card where they need to enter their name, card's name, card's expiration date and the cvc number.



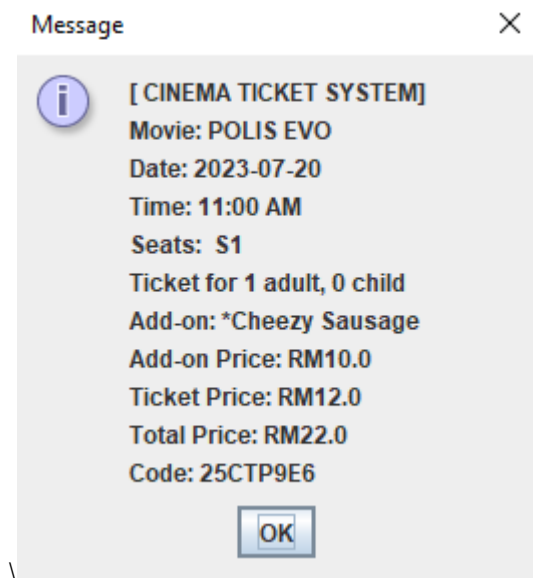
6. Membership purchase

Customers also can purchase membership to receive discount for their purchase. They also can make a payment for membership purchase whether using an online banking or debit card.



7. Receipt

The system will print the receipt of customer's purchase ticket.



12.0 REFERENCE

1. TEACHING AND LEARNING MODULE OBJECT ORIENTED PROGRAMMING USING JAVA BY NORMAH AHAMD, ITAZA AFIANI, NUR AZMINA ZAMANI.
2. <https://www.gsc.com.my/>