Name: Nguyen Phuong Le

ID: 104178943

# **Assignment 2**

## **State A:**

### **Customer:**

package assignment2;

public class Customer {

private String name;

private String phoneNumber;

// Constructor

public Customer(String name, String phoneNumber) {

this.name = name;

this.phoneNumber = phoneNumber;

}

//Get and set methods

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

}

### **Ticket:**

package assignment2;

import java.io.Serializable;

/\* The "Ticket" class contains information about travel tickets, including details such as the number of adults, and children, discounts, total price, and customers. \*/

public class Ticket implements Serializable {

private static final long ***serialVersionUID*** = 1L;

private int id;

private String tourName;

private int numberAdult;

private int numberChild;

private double discount;

private double total;

private String pickUp;

private String status;

// Transient make Customer not save in file

private transient Customer customer;

private double pAndD;

private String managerName;

// A complete constructor

public Ticket(int id, String tourName, int numberAdult, int numberChild, double discount, double total,

String pickUp, String status, Customer customer, double pAndD, String managerName) {

this.id = id;

this.tourName = tourName;

this.numberAdult = numberAdult;

this.numberChild = numberChild;

this.discount = discount;

this.total = total;

this.pickUp = pickUp;

this.status = status;

this.customer = customer;

this.pAndD = pAndD;

this.managerName = managerName;

}

//Get and set methods

public String getManagerName() {

return managerName;

}

public void setManagerName(String managerName) {

this.managerName = managerName;

}

public String getPickUp() {

return pickUp;

}

public void setPickUp(String pickUp) {

this.pickUp = pickUp;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getTourName() {

return tourName;

}

public void setTourName(String tourName) {

this.tourName = tourName;

}

public int getNumberAdult() {

return numberAdult;

}

public void setNumberAdult(int numberAdult) {

this.numberAdult = numberAdult;

}

public int getNumberChild() {

return numberChild;

}

public void setNumberChild(int numberChild) {

this.numberChild = numberChild;

}

public double getDiscount() {

return discount;

}

public void setDiscount(double discount) {

this.discount = discount;

}

public double getTotal() {

return total;

}

public void setTotal(double total) {

this.total = total;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

// Constructor 1

public Ticket(int id, String tourName, int numberAdult, int numberChild, double discount, double total,

String pickUp) {

this.id = id;

this.tourName = tourName;

this.numberAdult = numberAdult;

this.numberChild = numberChild;

this.discount = discount;

this.total = total;

this.pickUp = pickUp;

this.status = "Valid"; //assign value

}

// Constructor 2

// extended two more attributes "status" and "customer"

public Ticket(int id, String tourName, int numberAdult, int numberChild, double discount, double total,

String pickUp, String status, Customer customer, double pAndD) {

this.id = id;

this.tourName = tourName;

this.numberAdult = numberAdult;

this.numberChild = numberChild;

this.discount = discount;

this.total = total;

this.pickUp = pickUp;

this.status = status;

this.customer = customer;

this.pAndD = pAndD;

}

*@Override*

public String toString() {

return "Ticket - Id: " + id + ", Tour Name: " + tourName + ", Number Adult: " + numberAdult + ", Number Child:"

+ numberChild + ", Discount: " + discount + ", Total: " + total + ", Pick Up: " + pickUp + ", Status: "

+ status + ".";

}

//Get and set methods

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

public double getpAndD() {

return pAndD;

}

public void setpAndD(double pAndD) {

this.pAndD = pAndD;

}

}

### **ATour:**

package assignment2;

import java.util.HashMap;

import java.util.Map;

public abstract class ATour {

    private int id;

    private String name;

    // Price for adult and child

    private Map<String, Integer> price = new HashMap<>(); //Use "HashMap" to store adult and child ticket prices

    private int numberAvailability;

    //Get and set methods

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public Map<String, Integer> getPrice() {

        return price;

    }

    public void setPrice(Map<String, Integer> price) {

        this.price = price;

    }

    public int getNumberAvailability() {

        return numberAvailability;

    }

    public void setNumberAvailability(int numberAvailability) {

        this.numberAvailability = numberAvailability;

    }

// Constructor

    public ATour(int id, String name, Map<String, Integer> price) {

        this.id = id;

        this.name = name;

        this.price = price;

        // Maximum of number availability is 40

        this.numberAvailability = 40;

    }

    // Method calculation discount depend on number adult and child

    public double discountPercent(int numberAdult, int numberChild) {

        // 10% discount from the total ticket price is offered for 1 Adult and 1 Child or 2 Adults and 1 Child

        if((numberAdult == 1 && numberChild == 1) || (numberAdult == 2 && numberChild == 1)) {

            return 0.1;

        }

        // For families with 4 and above 15% discount is offered.

        else if(numberAdult + numberChild >= 4) {

            return 0.15;

        }

        // Else return 0

        return 0;

    }

    //Calculate discount amount

    public double disCount(int numberAdult, int numberChild) {

        return (price.get("adult") \* numberAdult + price.get("child") \* numberChild) \* this.discountPercent(numberAdult, numberChild);

    }

    //total amount after discount

    public double total(int numberAdult, int numberChild) {

        return price.get("adult") \* numberAdult + price.get("child") \* numberChild - disCount(numberAdult, numberChild);

    }

    public abstract String pickupPlace();

    public void printToString() {

        System.out.println(id + " : " + name);

    };

}

### **CityTour:**

package assignment2;

import java.util.Map;

// Inherit ATour class

public class CityTour extends ATour{

private String duration; //Describe the tour duration

//Get and set methods

public String getDuration() {

return duration;

}

public void setDuration(String duration) {

this.duration = duration;

}

// Constructor

public CityTour(int id, String name, String duration, Map<String, Integer> price) {

super(id, name, price);

this.duration = duration;

}

*@Override*

public String pickupPlace() {

// **TODO** Auto-generated method stub

return null;

}

}

### **AttractionTour:**

package assignment2;

import java.util.Map;

// Inherit ATour class

public class AttractionTour extends ATour {

private String inclution; //Store information about services included in the tour

// Constructor

public AttractionTour(int id, String name, String inclution, Map<String, Integer> price) {

super(id, name, price);

this.inclution = inclution;

}

//Get and set methods

public String getInclution() {

return inclution;

}

public void setInclution(String inclution) {

this.inclution = inclution;

}

*@Override*

public String pickupPlace() {

return "bus"; //"bus" string

}

}

### **StateA:**

package assignment2;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

public class StateA {

    public static List<ATour> tours = new ArrayList<>(); //Declare the "tours" list

    public static List<Ticket> tickets = new ArrayList<>(); //declare a "tickets" list to store booked tickets

    public static String[] listAttractionBus = {"Travel Australia CBD office", "Flinders Station", "Southern Cross Station", "Queen Victoria Market", "Melbourne Museum"};

    // Create Tour list

    void createTourList() {

        // Add city tour

        /\*  Map -> contains ticket price information for adults and children.

            Create a list of tours, including "CityTour" and "AttractionTour" \*/

        tours.add(new CityTour(1, "Melbourne City", "4 hours", Map.of("adult", 35, "child", 25)));

        tours.add(new CityTour(2, "Melbourne City & Yarra River Boat Cruise", "6 hours",

                Map.of("adult", 65, "child", 40)));

        tours.add(new CityTour(3, "Yarra River Cruise & Melbourne Zoo", "6 hours", Map.of("adult", 75, "child", 45)));

        tours.add(new CityTour(4, "Melbourne City & Melbourne Zoo", "7 hours", Map.of("adult", 65, "child", 40)));

        tours.add(new CityTour(5, "Melbourne City & Melbourne Aquarium", "6 hours", Map.of("adult", 75, "child", 45)));

        // Add attractions tour

        tours.add(new AttractionTour(6, "Great Ocean Road", "Lunch & Supper", Map.of("adult", 135, "child", 90)));

        tours.add(new AttractionTour(7, "Yarra Valley Wine Tasting ", "Lunch", Map.of("adult", 85, "child", 60)));

        tours.add(new AttractionTour(8, "Wilson Prom ", "Lunch & Evening Tea", Map.of("adult", 110, "child", 90)));

        tours.add(new AttractionTour(9, "Phillip Island & Penguin Parade", "Lunch & Supper",

                Map.of("adult", 160, "child", 120)));

    }

    // Display All City Tour

    void displayCityTour() {

        for (ATour tour : tours) { //Browse all tours in the "tours" list

            if (tour instanceof CityTour) { //Print tours of type "CityTour"

                tour.printToString();

            }

        }

    }

    // Display All Attraction Tour

    void displayAttractionTour() {

        for (ATour tour : tours) { //Browse all tours in the "tours" list

            if (tour instanceof AttractionTour) { //Print tours of type "AttractionTour"

                tour.printToString();

            }

        }

    }

    // Display Attraction Tour Bus

    void displayListAttractionBus() {

        for(int i = 1; i <= 5; i++) { //corresponding to 5 locations in the bus list

            System.out.println("" + i + ". " + listAttractionBus[i - 1]);

        }

    }

    // Calculate total of list tickets

    double getTotalTicket(List<Ticket> listTickets) {

        double total = 0; //total amount of all original tickets

        for(Ticket ticket : listTickets) {

            total += ticket.getTotal(); //Add the "total" value of each ticket to the "total" variable

        }

        return total;

    }

    // Reserve 1 Ticket

    Ticket reverseOneTicket(Scanner input, int idTicket) {

        System.out.println("\nWhich type of tour you wish to book");

        System.out.println("\tA: City Tour");

        System.out.println("\tB: Attractions");

        System.out.print("Please enter your choice: ");

        char choice = input.nextLine().charAt(0);

        System.out.println("Pick your choice");

        switch (choice) {

            case 'A':

                displayCityTour();

                break;

            case 'B':

                displayAttractionTour();

                break;

            default:

                System.out.println("Choice is invalid");

                return null;

        }

        //Find and book tours

                System.out.println("Which tour you wish to buy ticket/s for: ");

                int tourId = input.nextInt();

                input.nextLine();

                ATour tour = null;

                // Find tour with id

                for (ATour atour : tours) {

                    if (atour.getId() == tourId) {

                        tour = atour;

                    }

                }

                if (tour == null) {

                    System.out.println("Your tour is invalid");

                    return null;

                }

                // Get number adult and child

                System.out.println("Booking tickets for " + tour.getName());

                System.out.println("How many adult tickets - Maximum " + tour.getNumberAvailability());

                int numberAdult = input.nextInt();

                input.nextLine();

                System.out.println("How many child tickets - Maximum " + tour.getNumberAvailability());

                int numberChild = input.nextInt();

                input.nextLine();

                String pickUp = " ";

                if(tour.pickupPlace() == "bus") { //"bus" string

                    // Get board bus choice

                    displayListAttractionBus();

                    System.out.println("\nWhere do you wish to board the bus");

                    System.out.println("Please enter your choice:");

                    int busChoice = input.nextInt();

                    input.nextLine();

                    // Validate bus choice

                    if(busChoice <= 0 || busChoice > 5) {

                        System.out.println("Choice is invalid");

                        return null;

                    }

                    pickUp = listAttractionBus[busChoice - 1];

                }

                // Calculate discount and total

                double discount = tour.disCount(numberAdult, numberChild); //Discounts are calculated based on the number of adults and children participating in the tour.

                double total = tour.total(numberAdult, numberChild); //Calculate the total ticket price based on the number of adults and children.

                // Create new Ticket

                Ticket ticket = new Ticket(idTicket, tour.getName(), numberAdult, numberChild, discount,

                total, pickUp);

                // Decrease number availability of this tour

                tour.setNumberAvailability(tour.getNumberAvailability() - numberAdult - numberChild);

                return ticket;

    }

    // Reserve Ticket (can reserve multiple ticket) (A Choice)

    void reverseTicket(Scanner input) {

        List<Ticket> reserveTickets = new ArrayList<>();

        boolean multipleReserve = false;

        do {

            Ticket ticket = reverseOneTicket(input, tickets.size() + reserveTickets.size() + 1);

            if (ticket == null)

                return;

            reserveTickets.add(ticket);

            System.out.println("Reserve another tour/attraction ticket to this customer? Y/N");

            char choice = input.nextLine().charAt(0);

            switch (choice) {

                case 'Y':

                    multipleReserve = true;

                    break;

                case 'N':

                    multipleReserve = false;

                    break;

                default:

                    System.out.println("Choice is invalid");

                    return;

            }

        } while (multipleReserve);

        displayListTickets(reserveTickets);

        for (Ticket reserTicket : reserveTickets) {

            tickets.add(reserTicket);

        }

        return;

    }

    // Show availability of each tour (D choice)

    void displayListAvailability() {

        System.out.printf("%-40s %-20s%n", "\nTour", "Current Availability");

        for (ATour tour : tours) {

            System.out.printf("%-40s %-20d%n", tour.getName(), tour.getNumberAvailability());

        }

    }

    // Show sales of ticket (E and F choice)

    //Displays details of booked tickets.

    void displayListTickets(List<Ticket> listTickets) {

        System.out.printf("%-5s %-2s %-40s %-10s %-10s %-10s %-10s %10s%n", "Ticket", "Id", "Tour", "Adults", "Children",

                "Total", "Discount", "PickUp");

        System.out.println(

                "------------------------------------------------------------------------------------------------------------------------");

        for (Ticket reserTicket : listTickets) {

            System.out.printf("%-5s %-2d %-40s %3d %10d %13.2f %10.2f %27s%n", "------", reserTicket.getId(),

                    reserTicket.getTourName(), reserTicket.getNumberAdult(), reserTicket.getNumberChild(),

                    reserTicket.getTotal(), reserTicket.getDiscount(), reserTicket.getPickUp());

        }

        System.out.printf("%-10s %-10.2f%n", "------ Total: $", getTotalTicket(listTickets));

    }

    // Choice E

    //Show total number of tickets sold

    void displayAllSales() {

        System.out.println("Reversed Tickets");

        displayListTickets(tickets);

    }

    // Display all ticket by tour (Choice F)

    void displayReversedTicketOfTour(Scanner input) {

        displayCityTour();

        displayAttractionTour();

        System.out.print("Select the tour number: ");

        int idTour = input.nextInt();

        input.nextLine();

        ATour tour = null;

        for(ATour atour : tours) {

            if(atour.getId() == idTour) tour = atour;

        }

        if(tour == null) {

            System.out.println("Choice is invalid");

            return;

        }

        List<Ticket> listTickets = new ArrayList<>();

        for(Ticket reservedTicket: tickets) {

            if(reservedTicket.getTourName() == tour.getName()) {

                listTickets.add(reservedTicket);

            }

        }

        System.out.println("Reversed Tickets");

        displayListTickets(listTickets);

    }

    // Search Ticket (Choice B)

    void searchTicket(Scanner input) {

        System.out.println("Enter the ticket number");

        int idTicket = input.nextInt();

        input.nextLine();

        for(Ticket ticket : tickets) {

            if(ticket.getId() == idTicket) {

                System.out.println(ticket.toString());

                break;

            }

        }

        return;

    }

    // Refund ticket method

    void refundOneTicket(Ticket ticket) {

        ticket.setDiscount(0);

        ticket.setTotal(ticket.getTotal() \* 0.25); //25% of the amount will be lost when cancelling the ticket.

        ticket.setStatus("Canceled"); //change ticket status

        // Find the tour of ticket

        for(ATour tour: tours) {

            if(tour.getName().equals(ticket.getTourName())) {

                // Update number availability of this tour

                tour.setNumberAvailability(tour.getNumberAvailability() + ticket.getNumberAdult() + ticket.getNumberChild());

            }

        }

    }

    // Choice C

    void refundTicket(Scanner input) {

        System.out.println("Enter the ticket number for refund");

        int idTicket = input.nextInt();

        input.nextLine();

        Ticket reservedTicket = null;

        for(Ticket ticket : tickets) {

            if(ticket.getId() == idTicket) {

                reservedTicket = ticket;

            }

        }

        if(reservedTicket == null) {

            System.out.println("No ticket found");

            return;

        }

        reservedTicket.toString();

        // Calculate total refund ( 75% )

        double refundTotal = reservedTicket.getTotal() \* 0.75; ////75% of the amount the customer will receive after canceling the ticket

        System.out.println("Do you wish to cancel this ticket? By cancelling you will receive " + refundTotal + " as refund");

        System.out.println("Press Y to cancel or any key to abort cancellation");

        char agreeChoice = input.nextLine().charAt(0);

        if(agreeChoice == 'Y') {

            refundOneTicket(reservedTicket);

            System.out.println("Ticket cancelled. Your refund is $ " + refundTotal);

            return;

        } else {

            return;

        }

    }

    // Function create menu

    static void menu(Scanner input) {

        System.out.println("\nTravel Australia - Ticket Reservation");

        System.out.println("\tA: Reserver Ticket");

        System.out.println("\tB: Search Ticket");

        System.out.println("\tC: Refund Ticket");

        System.out.println("\tD: List Availability");

        System.out.println("\tE: List Sales");

        System.out.println("\tF: List Sales for a Tour");

        System.out.println("\tX: Exit");

        System.out.print("Please enter your choice: ");

        System.out.print("");

    }

    // Handle Client Choice

    int handleChoice(char choice, Scanner input) {

        switch (choice) {

            case 'A':

                reverseTicket(input);

                break;

            case 'B':

                searchTicket(input);

                break;

            case 'C':

                refundTicket(input);

                break;

            case 'D':

                displayListAvailability();

                break;

            case 'E':

                displayAllSales();

                break;

            case 'F':

                displayReversedTicketOfTour(input);

                break;

            case 'X':

                return 0;

            default:

                System.out.println("Your choice is invalid");

                break;

        }

        return 1;

    }

    // Main function

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        StateA stateA = new StateA();

        stateA.createTourList();

        // While loop to display menu always

        while (true) {

            menu(input);

            char choice = input.nextLine().charAt(0);

            int resultAfterChoice = stateA.handleChoice(choice, input);

            // If choice X, the while loop will be broken

            if (resultAfterChoice == 0)

                break;

        }

    }

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### **Junit test:**

#### **TicketTest:**

package test;

import static org.junit.jupiter.api.Assertions.\*;

import java.util.Map;

import org.junit.jupiter.api.Test;

import assignment2.ATour;

import assignment2.AttractionTour;

import assignment2.CityTour;

import assignment2.InternationalTour;

import assignment2.QueenslandTour;

class TicketTest {

*@Test*

public void queenland\_and\_international\_tour\_not\_have\_discount() {

ATour queenlandTour = new QueenslandTour(10, "Cairns", "Airfare + Accommodation", Map.*of*("adult", 850, "child", 700), 5);

ATour internationalTour = new InternationalTour(13, "Pattaya", "Airfare + Accommodation + Breakfast",

Map.*of*("adult", 1400, "child", 1150), 7);

int numberAdult = 2, numberChild = 2;

*assertEquals*(3100, queenlandTour.total(numberAdult, numberChild));

*assertEquals*(5100, internationalTour.total(numberAdult, numberChild));

*assertEquals*(0.0, queenlandTour.disCount(numberAdult, numberChild));

*assertEquals*(0.0, internationalTour.disCount(numberAdult, numberChild));

}

*@Test*()

public void city\_and\_attraction\_tour\_have\_discount() {

ATour cityTour = new CityTour(1, "Melbourne City", "4 hours", Map.*of*("adult", 35, "child", 25));

ATour attractionTour = new AttractionTour(1, "Great Ocean Road", "Lunch & Supper", Map.*of*("adult", 35, "child", 25));

int numberAdult = 2, numberChild = 2;

*assertEquals*(0.15, cityTour.discountPercent(numberAdult, numberChild));

*assertEquals*(0.15, attractionTour.discountPercent(numberAdult, numberChild));

*assertEquals*(18, cityTour.disCount(numberAdult, numberChild));

*assertEquals*(102, cityTour.total(numberAdult, numberChild));

*assertEquals*(18, attractionTour.disCount(numberAdult, numberChild));

*assertEquals*(102, attractionTour.total(numberAdult, numberChild));

}

*@Test*()

public void discount\_depend\_on\_number\_client() {

ATour cityTour = new CityTour(1, "Melbourne City", "4 hours", Map.*of*("adult", 35, "child", 25));

int numberAdult = 2, numberChild = 1, numberChild2 = 2;

*assertEquals*(0.1, cityTour.discountPercent(numberAdult, numberChild));

*assertEquals*(0.15, cityTour.discountPercent(numberAdult, numberChild2));

*assertEquals*(9.5, cityTour.disCount(numberAdult, numberChild));

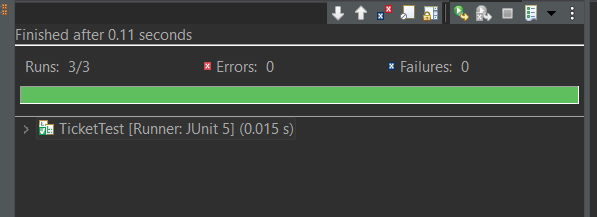
*assertEquals*(18, cityTour.disCount(numberAdult, numberChild2));

*assertEquals*(85.5, cityTour.total(numberAdult, numberChild));

*assertEquals*(102, cityTour.total(numberAdult, numberChild2));

}

}



#### **ATourTest:**

package test;

import static org.junit.jupiter.api.Assertions.\*;

import java.util.Map;

import org.junit.Assert;

import org.junit.jupiter.api.Test;

import assignment2.ATour;

import assignment2.AttractionTour;

import assignment2.CityTour;

import assignment2.InternationalTour;

import assignment2.QueenslandTour;

class ATourTest {

*@Test*()

public void test\_create\_city\_tour() {

ATour tour = new CityTour(1, "Melbourne City", "4 hours", Map.*of*("adult", 35, "child", 25));

// Test name, id, duration

Assert.*assertEquals*(tour.getId(), 1);

Assert.*assertEquals*(tour.getName(), "Melbourne City");

Assert.*assertEquals*(tour.pickupPlace(), null);

}

*@Test*()

public void test\_create\_attraction\_tour() {

ATour tour = new AttractionTour(1, "Great Ocean Road", "Lunch & Supper", Map.*of*("adult", 35, "child", 25));

// Test name, id, duration

Assert.*assertEquals*(tour.getId(), 1);

Assert.*assertEquals*(tour.getName(), "Great Ocean Road");

Assert.*assertEquals*(tour.pickupPlace(), "bus");

}

*@Test*()

public void test\_create\_queenland\_tour() {

ATour tour = new QueenslandTour(10, "Cairns", "Airfare + Accommodation", Map.*of*("adult", 850, "child", 700), 5);

// Test name, id, duration

Assert.*assertEquals*(tour.getId(), 10);

Assert.*assertEquals*(tour.getName(), "Cairns");

Assert.*assertEquals*(tour.pickupPlace(), "airport");

}

*@Test*()

public void test\_create\_international\_tour() {

ATour tour = new InternationalTour(13, "Pattaya", "Airfare + Accommodation + Breakfast",

Map.*of*("adult", 1400, "child", 1150), 7);

// Test name, id, duration

Assert.*assertEquals*(tour.getId(), 13);

Assert.*assertEquals*(tour.getName(), "Pattaya");

Assert.*assertEquals*(tour.pickupPlace(), "airport");

}

}

A screen shot of a computer error

Description automatically generated

### **UML:**

|  |
| --- |
| Customer |
| ------------------------------------------------------------------------------------ |
| - name: String |
| - phoneNumber: String |
| ------------------------------------------------------------------------------------ |
| + Customer(name: String, phoneNumber: String) |
| + getName(): String |
| + setName(name: String): void |
| + getPhoneNumber(): String |
| + setPhoneNumber(phone: String): void |

|  |
| --- |
| Ticket |
| ------------------------------------------------------------------------------------ |
| - id : int |
| - tourName: String |
| - numberAdult: int |
| - numberChild: int |
| - discount: double |
| - total: double |
| - pickUp: String |
| - status: String |
| - customer: Customer (transient) |
| - pAndD: double |
| - managerName: String |
| ------------------------------------------------------------------------------------ |
| + Ticket() |
| + getId() : int |
| + setId(id: int) : void |
| + getTourName(): String |
| + setTourName(tourName: String): void |
| + getNumberAdult(): int |
| + setNumberAdult(numberAdult: int): void |
| + getNumberChild(): int |
| + setNumberChild(numberChild: int): void |
| + getDiscount(): double |
| + setDiscount(discount: double): void |
| + getTotal(): double |
| + setTotal(total: double): void |
| + getPickUp(): String |
| + setPickUp(pickUp: String): void |
| + getStatus(): String |
| + setStatus(status: String): void |
| + getCustomer(): Customer |
| + setCustomer(customer: Customer): void |
| + getpAndD(): double |
| + setpAndD(pAndD: double): void |
| + toString(): String |

|  |
| --- |
| ATour <abstract> |
| ------------------------------------------------------------------------------------ |
| - id : int |
| - name : String |
| - price : Map<String, Integer> |
| - numberAvailability : int |
| ------------------------------------------------------------------------------------ |
| + getId() : int |
| + setId(id: int) : void |
| + getName() : String |
| + setName(name: String) : void |
| + getPrice() : Map<String, Integer> |
| + setPrice(price: Map<String, Integer>) : void |
| + getNumberAvailability() : int |
| + setNumberAvailability(numberAvailability: int) : void |
| + discountPercent(numberAdult: int, numberChild: int) : double |
| + disCount(numberAdult: int, numberChild: int) : double |
| + total(numberAdult: int, numberChild: int) : double |
| + printToString() : void |
| + pickupPlace() : String [abstract] |

|  |
| --- |
| CityTour |
| ------------------------------------------------------------------------------------ |
| - duration: String |
| ------------------------------------------------------------------------------------ |
| + CityTour(id: int, name: String, duration: String, price: Map<String, Integer>) |
| + getDuration(): String |
| + setDuration(duration: String): void |
| + pickupPlace(): String |

|  |
| --- |
| AttractionTour |
| ------------------------------------------------------------------------------------ |
| - inclution: String |
| ------------------------------------------------------------------------------------ |
| + AttractionTour(id: int, name: String, inclution: String, price: Map<String, Integer>) |
| + getInclution(): String |
| + setInclution(inclution: String): void |
| + pickupPlace(): String |

|  |
| --- |
| StateA |
| ------------------------------------------------------------------------------------ |
| - tours: List<ATour> |
| - tickets: List<Ticket> |
| - listAttractionBus: String[] |
| ------------------------------------------------------------------------------------ |
| + createTourList() |
| + displayCityTour() |
| + displayAttractionTour() |
| + displayListAttractionBus() |
| + getTotalTicket(List<Ticket>): double |
| + reverseOneTicket(Scanner, int): Ticket |
| + reverseTicket(Scanner) |
| + displayListAvailability() |
| + displayListTickets(List<Ticket>) |
| + displayAllSales() |
| + displayReversedTicketOfTour(Scanner) |
| + searchTicket(Scanner) |
| + refundOneTicket(Ticket) |
| + refundTicket(Scanner) |
| + menu(Scanner) |
| + handleChoice(char, Scanner): int |
| + main(String[] args) |

**Note:**

* "CityTour" and "AttractionTour" are inherited from "ATour" class.

1. **initialize tour list**

StateA → ATour (CityTour/AttractionTour)

1. **Create Ticket objects and save information**

StateA → Ticket

1. **Enter customer information during booking process**

StateA → Customer

1. **"Ticket" stores the "ATour" information the user has booked.**

Ticket → ATour (CityTour/AttractionTour)

1. **"Ticket" also stores "Customer" information to associate the ticket with the customer.**

Ticket → Customer

1. **"ATour" reduces the number of available tickets when a ticket is booked and adjusts the information accordingly.**

ATour (CityTour/AttractionTour) → Ticket

## **State B:**

### **QueenslandTour:**

package assignment2;

import java.util.Map;

public class QueenslandTour extends ATour {

private int days; //Number of days of tour.

private String inclution; //Store information about services included in the tour

//Get and set methods

public int getDays() {

return days;

}

public void setDays(int days) {

this.days = days;

}

// Constructor

public QueenslandTour(int id, String name, String inclution, Map<String, Integer> price, int days) {

super(id, name, price);

this.inclution = inclution;

this.days = days;

}

*@Override*

public double disCount(int numberAdult, int numberChild) {

return 0;

}

//Get and set methods

*@Override*

public String pickupPlace() {

return "airport";

}

public String getInclution() {

return inclution;

}

public void setInclution(String inclution) {

this.inclution = inclution;

}

}

### **InternationalTour:**

package assignment2;

import java.util.Map;

public class InternationalTour extends ATour {

private String inclution; //Store information about services included in the tour

private int days; //Number of days of tour.

//Get and set methods

public int getDays() {

return days;

}

public void setDays(int days) {

this.days = days;

}

// Constructor

public InternationalTour(int id, String name, String inclution, Map<String, Integer> price, int days) {

super(id, name, price);

this.inclution = inclution;

this.days = days;

}

*@Override*

public double disCount(int numberAdult, int numberChild) {

return 0;

}

//Get and set methods

*@Override*

public String pickupPlace() {

return "airport";

}

public String getInclution() {

return inclution;

}

public void setInclution(String inclution) {

this.inclution = inclution;

}

}

### **State B:**

package assignment2;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

public class StateB extends StateA {

    @Override

    void createTourList() {

        // Add new tours to the "tours" list. Tours include "QueenslandTour" and "InternationalTour"

        super.createTourList();

        tours.add(new QueenslandTour(10, "Cairns", "Airfare + Accommodation", Map.of("adult", 850, "child", 700), 5));

        tours.add(new QueenslandTour(11, "Gold Cost ", "Airfare + Accommodation", Map.of("adult", 600, "child", 450), 5));

        tours.add(new InternationalTour(12, "Phuket", "Airfare + Accommodation + Breakfast",

                Map.of("adult", 1350, "child", 1100), 7));

        tours.add(new InternationalTour(13, "Pattaya", "Airfare + Accommodation + Breakfast",

                Map.of("adult", 1400, "child", 1150), 7));

        tours.add(new InternationalTour(14, "Singapore", "Airfare + Accommodation", Map.of("adult", 1200, "child", 1000),

                7));

    }

    void displayQueenslandTour() {

        for (ATour tour : tours) { //Browse all tours in the "tours" list

            if (tour instanceof QueenslandTour) { //Print tours of type "QueenslandTour"

                tour.printToString();

            }

        }

    }

    void displauInternationalTour() {

        for (ATour tour : tours) { //Browse all tours in the "tours" list

            if (tour instanceof InternationalTour) { //Print tours of type "InternationalTour"

                tour.printToString();

            }

        }

    }

    // Override reserve one ticket method

    @Override

    Ticket reverseOneTicket(Scanner input, int idTicket) {

        System.out.println("\nNWhich type of tour you wish to book");

        System.out.println("\tA: City Tour");

        System.out.println("\tB: Attractions");

        System.out.println("\tC: Queensland");

        System.out.println("\tD: Overseas");

        System.out.print("Please enter your choice: ");

        char choice = input.nextLine().charAt(0);

        System.out.println("Pick your choice");

        switch (choice) {

            case 'A':

                displayCityTour();

                break;

            case 'B':

                displayAttractionTour();

                break;

            case 'C':

                displayQueenslandTour();

                break;

            case 'D':

                displauInternationalTour();

                break;

            default:

                System.out.println("Choice is invalid");

                return null;

        }

        System.out.println("Which tour you wish to buy ticket/s for: ");

        int tourId = input.nextInt();

        input.nextLine();

        ATour tour = null;

        // Find tour with id

        for (ATour atour : tours) {

            if (atour.getId() == tourId) {

                tour = atour;

            }

        }

        if (tour == null) {

            System.out.println("Your tour is invalid");

            return null;

        }

        // Get number adult and child

        System.out.println("Booking tickets for " + tour.getName());

        System.out.println("How many adult tickets - Maximum " + tour.getNumberAvailability());

        int numberAdult = input.nextInt();

        input.nextLine();

        System.out.println("How many child tickets - Maximum " + tour.getNumberAvailability());

        int numberChild = input.nextInt();

        input.nextLine();

        String pickUp = "--";

        double pAndD = 0;

        Customer customer = null;

        // If tour is not overseas

        if (tour.pickupPlace() == "bus") { //"bus" string

            // Get board bus choice

            displayListAttractionBus();

            System.out.println("\nWhere do you wish to board the bus");

            System.out.println("Please enter your choice:");

            int busChoice = input.nextInt();

            input.nextLine();

            // Validate bus choice

            if (busChoice <= 0 || busChoice > 5) {

                System.out.println("Choice is invalid");

                return null;

            }

            pickUp = listAttractionBus[busChoice - 1];

        }

        // If tour is overseas

        else if (tour.pickupPlace() == "airport") {

            System.out.println("Enter boarding airport");

            pickUp = input.nextLine();

            System.out.println("Add Airport Pickup & Drop at the destination at %150. Y/N?");

            char pickupChoice = input.nextLine().charAt(0);

            // Have Pick And Drop fee

            switch (pickupChoice) {

                case 'Y':

                    pAndD = 150;

                    break;

                case 'N':

                    pAndD = 0;

                    break;

                default:

                    System.out.println("Choice is invalid");

                    return null;

            }

            System.out.println("For interstate/Overseas Travel enter customer name and Contact number");

            System.out.println("Name:");

            String name = input.nextLine();

            System.out.println("Contact Nubmer:");

            String phoneNumber = input.nextLine();

            customer = new Customer(name, phoneNumber);

        }

        // Calculate discount and total

        double discount = tour.disCount(numberAdult, numberChild);

        double total = tour.total(numberAdult, numberChild);

        // Create new Ticket

        Ticket ticket = new Ticket(idTicket, tour.getName(), numberAdult, numberChild, discount,

                total, pickUp);

        if(customer != null) {

            ticket.setpAndD(pAndD);

            ticket.setCustomer(customer);

        }

        // Decrease number availability of this tour

        tour.setNumberAvailability(tour.getNumberAvailability() - numberAdult - numberChild);

        return ticket;

    }

    // Override display list tickets method

    @Override

    void displayListTickets(List<Ticket> listTickets) {

        System.out.printf("%-5s %-2s %-40s %-6s %-8s %-10s %-10s %-20s %-10s %-10s %-13s%n", "Ticket", "Id", "Tour", "Adults", "Children",

                "Total", "Discount", "PickUp/AirBoard", "P&D", "Customer", "Contract number");

        System.out.println(

                "------------------------------------------------------------------------------------------------------------------------------------------------------------------------");

        for (Ticket reserTicket : listTickets) {

            if(reserTicket.getCustomer() != null) {

                System.out.printf("%-5s %-2d %-40s %-6d %-8d %-10.2f %-10.2f %-20s %-10.2f %-10s %-13s%n", "------", reserTicket.getId(),

                reserTicket.getTourName(), reserTicket.getNumberAdult(), reserTicket.getNumberChild(),

                reserTicket.getTotal(), reserTicket.getDiscount(), reserTicket.getPickUp(), reserTicket.getpAndD(), reserTicket.getCustomer().getName(), reserTicket.getCustomer().getPhoneNumber());

            } else {

                System.out.printf("%-5s %-2d %-40s %-6d %-8d %-10.2f %-10.2f %-20s%n", "------", reserTicket.getId(),

                    reserTicket.getTourName(), reserTicket.getNumberAdult(), reserTicket.getNumberChild(),

                    reserTicket.getTotal(), reserTicket.getDiscount(), reserTicket.getPickUp());

            }

        }

        System.out.printf("%-10s %-10.2f%n", "------ Total: $", getTotalTicket(listTickets));

    }

    // Show availability of each tour (D choice)

    @Override

    void displayListAvailability() {

        System.out.printf("%-40s %-20s%n", "Tour", "Current Availability");

        for (ATour tour : tours) {

            if(tour instanceof QueenslandTour || tour instanceof InternationalTour) {

                continue;

            }

            System.out.printf("%-40s %-20d%n", tour.getName(), tour.getNumberAvailability());

        }

    }

    @Override

    void displayReversedTicketOfTour(Scanner input) {

        displayCityTour();

        displayAttractionTour();

        displayQueenslandTour();

        displauInternationalTour();

        System.out.print("Select the tour number: ");

        int idTour = input.nextInt();

        input.nextLine();

        ATour tour = null;

        for(ATour atour : tours) {

            if(atour.getId() == idTour) tour = atour;

        }

        if(tour == null) {

            System.out.println("Choice is invalid");

            return;

        }

        List<Ticket> listTickets = new ArrayList<>();

        for(Ticket reservedTicket: tickets) {

            if(reservedTicket.getTourName() == tour.getName()) {

                listTickets.add(reservedTicket);

            }

        }

        System.out.println("Reversed Tickets");

        displayListTickets(listTickets);

    }

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        StateB stateB = new StateB();

        stateB.createTourList();

        // While loop to display menu always

        while (true) {

            menu(input);

            char choice = input.nextLine().charAt(0);

            int resultAfterChoice = stateB.handleChoice(choice, input);

            // If choice X, the while loop will be broken

            if (resultAfterChoice == 0)

                break;

        }

    }

}

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

## **State C:**

### **InsuranceException:**

package assignment2;

public class InsuranceException extends Exception {

public InsuranceException(String message) {

super(message);

}

}

### **IInsurance:**

package assignment2;

public interface IInsurance {

double getPriceInsuarance(int numberClient);

void createInsuarance(ATour tour, boolean isCreateInsurance) throws InsuranceException;

}

### **StateC:**

package assignment2;

import java.util.Scanner;

public class StateC extends StateB implements IInsurance {

*@Override*

public void createInsuarance(ATour tour, boolean isCreateInsurance) throws InsuranceException {

// If oversea tour, must have Insurance

// If the tour is international without insurance, throw an "InsuranceException"

if (!isCreateInsurance && tour instanceof InternationalTour) {

throw new InsuranceException("Overseas Travel requires Insurance cover. Reservation not made");

}

// If Victoria (state A) day tour, can not have insurance

// If the tour is one of the non-insurance types and the user chooses insurance, also throws an "InsuranceException" exception

else if (isCreateInsurance && (tour instanceof CityTour || tour instanceof AttractionTour)) {

throw new InsuranceException("Victoria Day Tour no need Insurance cover.");

}

}

*@Override*

public double getPriceInsuarance(int numberClient) {

// Single person

if(numberClient == 1) return 200;

// Couple

else if (numberClient == 2) return 350;

// Family

else if(numberClient > 2 && numberClient <= 5) return 500;

else if(numberClient <= 0) return 0;

else {

// Over 5 person

return 200 \* numberClient;

}

}

// Override reserve one ticket method

*@Override*

Ticket reverseOneTicket(Scanner input, int idTicket) {

System.***out***.println("Which type of tour you wish to book");

System.***out***.println("\tA: City Tour");

System.***out***.println("\tB: Attractions");

System.***out***.println("\tC: Queensland");

System.***out***.println("\tD: Overseas");

System.***out***.print("Please enter your choice: ");

char choice = input.nextLine().charAt(0);

System.***out***.println("Pick your choice");

switch (choice) {

case 'A':

displayCityTour();

break;

case 'B':

displayAttractionTour();

break;

case 'C':

displayQueenslandTour();

break;

case 'D':

displauInternationalTour();

break;

default:

System.***out***.println("Choice is invalid");

return null;

}

System.***out***.println("Which tour you wish to buy ticket/s for: ");

int tourId = input.nextInt();

input.nextLine();

ATour tour = null;

// Find tour with id

for (ATour atour : *tours*) {

if (atour.getId() == tourId) {

tour = atour;

}

}

if (tour == null) {

System.***out***.println("Your tour is invalid");

return null;

}

// Get number adult and child

System.***out***.println("Booking tickets for " + tour.getName());

System.***out***.println("How many adult tickets - Maximum " + tour.getNumberAvailability());

int numberAdult = input.nextInt();

input.nextLine();

System.***out***.println("How many child tickets - Maximum " + tour.getNumberAvailability());

int numberChild = input.nextInt();

input.nextLine();

String pickUp = "--";

double pAndD = 0;

Customer customer = null;

// If tour is not overseas

if (tour.pickupPlace() == "bus") { //"bus" string

// Get board bus choice

displayListAttractionBus();

System.***out***.println("\nWhere do you wish to board the bus");

System.***out***.println("Please enter your choice:");

int busChoice = input.nextInt();

input.nextLine();

// Validate bus choice

if (busChoice <= 0 || busChoice > 5) {

System.***out***.println("Choice is invalid");

return null;

}

pickUp = *listAttractionBus*[busChoice - 1];

}

// If tour is overseas

else if (tour.pickupPlace() == "airport") {

System.***out***.println("Enter boarding airport");

pickUp = input.nextLine();

System.***out***.println("Add Airport Pickup & Drop at the destination at %150. Y/N?");

char pickupChoice = input.nextLine().charAt(0);

// Have Pick And Drop fee

switch (pickupChoice) {

case 'Y':

pAndD = 150;

break;

case 'N':

pAndD = 0;

break;

default:

System.***out***.println("Choice is invalid");

return null;

}

System.***out***.println("For interstate/Overseas Travel enter customer name and Contact number");

System.***out***.println("Name:");

String name = input.nextLine();

System.***out***.println("Contact Nubmer:");

String phoneNumber = input.nextLine();

customer = new Customer(name, phoneNumber);

}

// Create Insurance

double insurancePrice = getPriceInsuarance(numberAdult + numberChild);

System.***out***.println("The tour is being planned for " + (numberAdult + numberChild) + " guest/s: This requires insurne cover for : $" + insurancePrice);

System.***out***.println("Confirm insurance cover for " + insurancePrice + ". Y/N ?");

char insuranceChoice = input.nextLine().charAt(0);

boolean changeToBoolean = false;

switch (insuranceChoice) {

case 'Y':

changeToBoolean = true;

break;

case 'N':

changeToBoolean = false;

break;

default:

System.***out***.println("Choice is invalid");

return null;

}

// Handling exception

try {

createInsuarance(tour, changeToBoolean);

} catch (Exception e) {

System.***out***.println(e.getMessage());

}

// Calculate discount and total

double discount = tour.disCount(numberAdult, numberChild);

double total = tour.total(numberAdult, numberChild);

// Create new Ticket

Ticket ticket = new Ticket(idTicket, tour.getName(), numberAdult, numberChild, discount,

total, pickUp);

if(customer != null) {

ticket.setpAndD(pAndD);

ticket.setCustomer(customer);

}

// Decrease number availability of this tour

tour.setNumberAvailability(tour.getNumberAvailability() - numberAdult - numberChild);

return ticket;

}

public static void main(String[] args) {

Scanner input = new Scanner(System.***in***);

StateC stateC = new StateC();

stateC.createTourList();

// While loop to display menu always

while (true) {

*menu*(input);

char choice = input.nextLine().charAt(0);

int resultAfterChoice = stateC.handleChoice(choice, input);

// If choice X, the while loop will be broken

if (resultAfterChoice == 0)

break;

}

}

}

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

## **State D:**

### **Account:**

package assignment2;

import java.io.Serializable;

public class Account implements Serializable {

private static final long ***serialVersionUID*** = 1L;

private String name;

private String username;

private String password;

//Get and set methods

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

// Constructor

public Account(String name, String username, String password) {

this.name = name;

this.username = username;

this.password = password;

}

}

### **State D:**

package assignment2;

import java.io.EOFException;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class StateD extends StateC {

    public static List<Account> accounts = new ArrayList<>(); // List of archives in "Account"

    public static Account currentAccount; //Stores the currently logged in account.

    // Override reserve one ticket method

   @Override

   Ticket reverseOneTicket(Scanner input, int idTicket) {

       System.out.println("Which type of tour you wish to book");

       System.out.println("\tA: City Tour");

       System.out.println("\tB: Attractions");

       System.out.println("\tC: Queensland");

       System.out.println("\tD: Overseas");

       System.out.print("Please enter your choice: ");

       char choice = input.nextLine().charAt(0);

       System.out.println("Pick your choice");

       switch (choice) {

           case 'A':

               displayCityTour();

               break;

           case 'B':

               displayAttractionTour();

               break;

           case 'C':

               displayQueenslandTour();

               break;

           case 'D':

               displauInternationalTour();

               break;

           default:

               System.out.println("Choice is invalid");

               return null;

       }

       System.out.println("Which tour you wish to buy ticket/s for: ");

       int tourId = input.nextInt();

       input.nextLine();

       ATour tour = null;

       // Find tour with id

       for (ATour atour : tours) {

           if (atour.getId() == tourId) {

               tour = atour;

           }

       }

       if (tour == null) {

           System.out.println("Your tour is invalid");

           return null;

       }

       // Get number adult and child

       System.out.println("Booking tickets for " + tour.getName());

       System.out.println("How many adult tickets - Maximum " + tour.getNumberAvailability());

       int numberAdult = input.nextInt();

       input.nextLine();

       System.out.println("How many child tickets - Maximum " + tour.getNumberAvailability());

       int numberChild = input.nextInt();

       input.nextLine();

       String pickUp = "--";

       double pAndD = 0;

       Customer customer = null;

       // If tour is not overseas

       if (tour.pickupPlace() == "bus") {

           // Get board bus choice

           displayListAttractionBus();

           System.out.println("\nWhere do you wish to board the bus");

           System.out.println("Please enter your choice:");

           int busChoice = input.nextInt();

           input.nextLine();

           // Validate bus choice

           if (busChoice <= 0 || busChoice > 5) {

               System.out.println("Choice is invalid");

               return null;

           }

           pickUp = listAttractionBus[busChoice - 1];

       }

       // If tour is overseas

       else if (tour.pickupPlace() == "airport") {

           System.out.println("Enter boarding airport");

           pickUp = input.nextLine();

           System.out.println("Add Airport Pickup & Drop at the destination at %150. Y/N?");

           char pickupChoice = input.nextLine().charAt(0);

           // Have Pick And Drop fee

           switch (pickupChoice) {

               case 'Y':

                   pAndD = 150;

                   break;

               case 'N':

                   pAndD = 0;

                   break;

               default:

                   System.out.println("Choice is invalid");

                   return null;

           }

           System.out.println("For interstate/Overseas Travel enter customer name and Contact number");

           System.out.println("Name:");

           String name = input.nextLine();

           System.out.println("Contact Nubmer:");

           String phoneNumber = input.nextLine();

           customer = new Customer(name, phoneNumber);

       }

       // Create Insurance

       double insurancePrice = getPriceInsuarance(numberAdult + numberChild);

       System.out.println("The tour is being planned for " + (numberAdult + numberChild) + " guest/s: This requires insurne cover for : $" + insurancePrice);

       System.out.println("Confirm insurance cover for " + insurancePrice + ". Y/N ?");

       char insuranceChoice = input.nextLine().charAt(0);

       boolean changeToBoolean = false;

       switch (insuranceChoice) {

           case 'Y':

               changeToBoolean = true;

               break;

           case 'N':

               changeToBoolean = false;

               break;

           default:

               System.out.println("Choice is invalid");

               return null;

       }

       // Handling exception

       try {

           createInsuarance(tour, changeToBoolean);

       } catch (Exception e) {

           System.out.println(e.getMessage());

       }

       // Calculate discount and total

       double discount = tour.disCount(numberAdult, numberChild);

       double total = tour.total(numberAdult, numberChild);

       // Create new Ticket

       Ticket ticket = new Ticket(idTicket, tour.getName(), numberAdult, numberChild, discount,

               total, pickUp);

       if(customer != null) {

           ticket.setpAndD(pAndD);

           ticket.setCustomer(customer);

       }

       ticket.setManagerName(currentAccount.getName());

       // Decrease number availability of this tour

       tour.setNumberAvailability(tour.getNumberAvailability() - numberAdult - numberChild);

       return ticket;

   }

    // Override display list tickets method

    @Override

    void displayListTickets(List<Ticket> listTickets) {

        System.out.printf("%-5s %-2s %-40s %-6s %-8s %-10s %-10s %-20s %-10s %-10s %-13s %-10s%n", "Ticket", "Id", "Tour", "Adults", "Children",

                "Total", "Discount", "PickUp/AirBoard", "P&D", "Customer", "Contract number", "Sales Admin");

        System.out.println(

                "-----------------------------------------------------------------------------------------------------------------------------------------------------------------");

        for (Ticket reserTicket : listTickets) {

            if(reserTicket.getCustomer() != null) {

                System.out.printf("%-5s %-2d %-40s %-6d %-8d %-10.2f %-10.2f %-20s %-10.2f %-10s %-13s %12s%n", "------", reserTicket.getId(),

                reserTicket.getTourName(), reserTicket.getNumberAdult(), reserTicket.getNumberChild(),

                reserTicket.getTotal(), reserTicket.getDiscount(), reserTicket.getPickUp(), reserTicket.getpAndD(), reserTicket.getCustomer().getName(), reserTicket.getCustomer().getPhoneNumber(), reserTicket.getManagerName());

            } else {

                System.out.printf("%-5s %-2d %-40s %-6d %-8d %-10.2f %-10.2f %-20s %-10s %-10s %-13s %12s%n", "------", reserTicket.getId(),

                    reserTicket.getTourName(), reserTicket.getNumberAdult(), reserTicket.getNumberChild(),

                    reserTicket.getTotal(), reserTicket.getDiscount(), reserTicket.getPickUp(), "--", "--", "--", reserTicket.getManagerName());

            }

        }

        System.out.printf("%-10s %-10.2f%n", "------ Total: $", getTotalTicket(listTickets));

    }

    //Write a list of tickets to a file.

    void writeTicketsToFile(String filename) {

        try (ObjectOutputStream ois = new ObjectOutputStream(new FileOutputStream(filename))) {

            ois.writeObject(tickets);

            System.out.println("Ticket have been written to the file.");

        }  catch (Exception e) {

            e.printStackTrace();

        }

    }

    //read list of tickets from a file

    void readTicketsFromFile(String filename) {

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {

            tickets = (List<Ticket>) ois.readObject();

            System.out.println(accounts.get(0).getName());

        } catch (EOFException e) {

            // If file is empty, create account first

            return;

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    // Create Account method

    void createAccounts(Scanner input) {

        System.out.println("No Sales Executives Account Exists. Create new Sales Executives Accounts");

        System.out.print("Enter Admin Username to login: ");

        String adminUsername = input.nextLine();

        System.out.print("Enter Admin Password to login: ");

        String adminPassword = input.nextLine();

        accounts.add(new Account(adminUsername, adminUsername, adminPassword));

        do {

            System.out.print("Enter Sales Executives Name: ");

            String name = input.nextLine();

            System.out.print("Enter UserName: ");

            String username = input.nextLine();

            System.out.print("Enter Password: ");

            String password = input.nextLine();

            accounts.add(new Account(name, username, password));

            System.out.print("Do you wish to add one more login? Yes/No?: ");

            String choice = input.nextLine();

            if(choice.equals("No")) {

                break;

            } else if (!choice.equals("Yes") && !choice.equals("No")) {

                System.out.println("Choice is invalid");

                return;

            }

        } while (true);

        writeAccountToFile("C:\\Users\\Administrator\\OneDrive - Swinburne University\\COS10033-Advanced Programming\\104178943\_Assignment\_2\\assignment2\\data\\account.txt");

    }

    // Save account to file

    void writeAccountToFile(String filename) {

        try (ObjectOutputStream ois = new ObjectOutputStream(new FileOutputStream(filename))) {

            ois.writeObject(accounts);

            System.out.println("Accounts have been written to the file.");

        }  catch (Exception e) {

            e.printStackTrace();

        }

    }

    // Read account from file

    void readAccountFromFile(Scanner input, String filename) {

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {

            accounts = (List<Account>) ois.readObject();

            System.out.println(accounts.get(0).getName());

        } catch (EOFException e) {

            // If file is empty, create account first

            createAccounts(input);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    // Login method

    static boolean login(Scanner input) {

        System.out.println("Please Login");

        System.out.print("Enter username: ");

        String username = input.nextLine();

        System.out.print("Enter password: ");

        String password = input.nextLine();

        for(Account acc : accounts) {

            if(acc.getUsername().equals(username) && acc.getPassword().equals(password)) {

                // Set current account

                currentAccount = acc;

                return true;

            }

        }

        return false;

    }

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        StateD stateD = new StateD();

        stateD.createTourList();

        stateD.readAccountFromFile(input, "C:\\Users\\Administrator\\OneDrive - Swinburne University\\COS10033-Advanced Programming\\104178943\_Assignment\_2\\assignment2\\data\\account.txt");

        // Login first

        do {

            boolean isLoginValid = login(input);

            if(!isLoginValid) {

                System.out.println("Login fail");

            } else {

                break;

            }

        } while(true);

        String ticketFileName = "C:\\Users\\Administrator\\OneDrive - Swinburne University\\COS10033-Advanced Programming\\104178943\_Assignment\_2\\assignment2\\data\\data.txt";

        stateD.readTicketsFromFile(ticketFileName);

        while (true) {

            menu(input);

            char choice = input.nextLine().charAt(0);

            int resultAfterChoice = stateD.handleChoice(choice, input);

            // If choice X, the while loop will be broken

            if (resultAfterChoice == 0) {

                // Save ticket before exit

                stateD.writeTicketsToFile(ticketFileName);

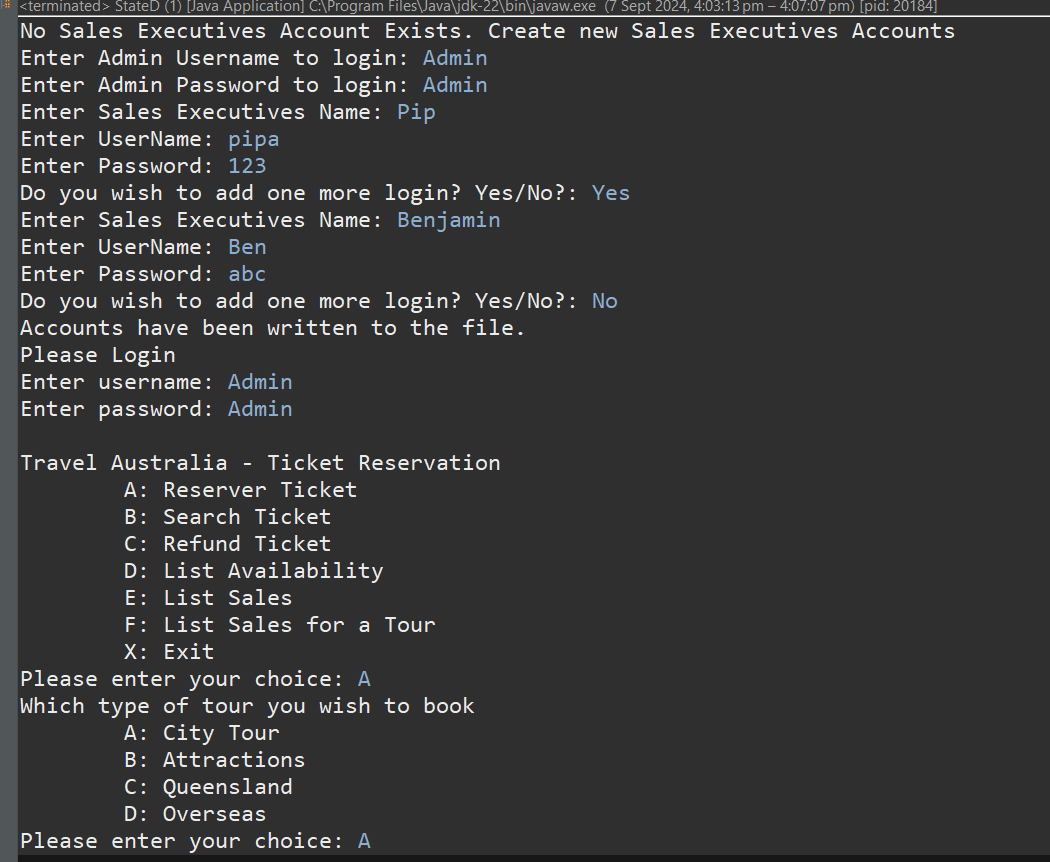
                break;

            }

        }

    }

}

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A black background with white text

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated