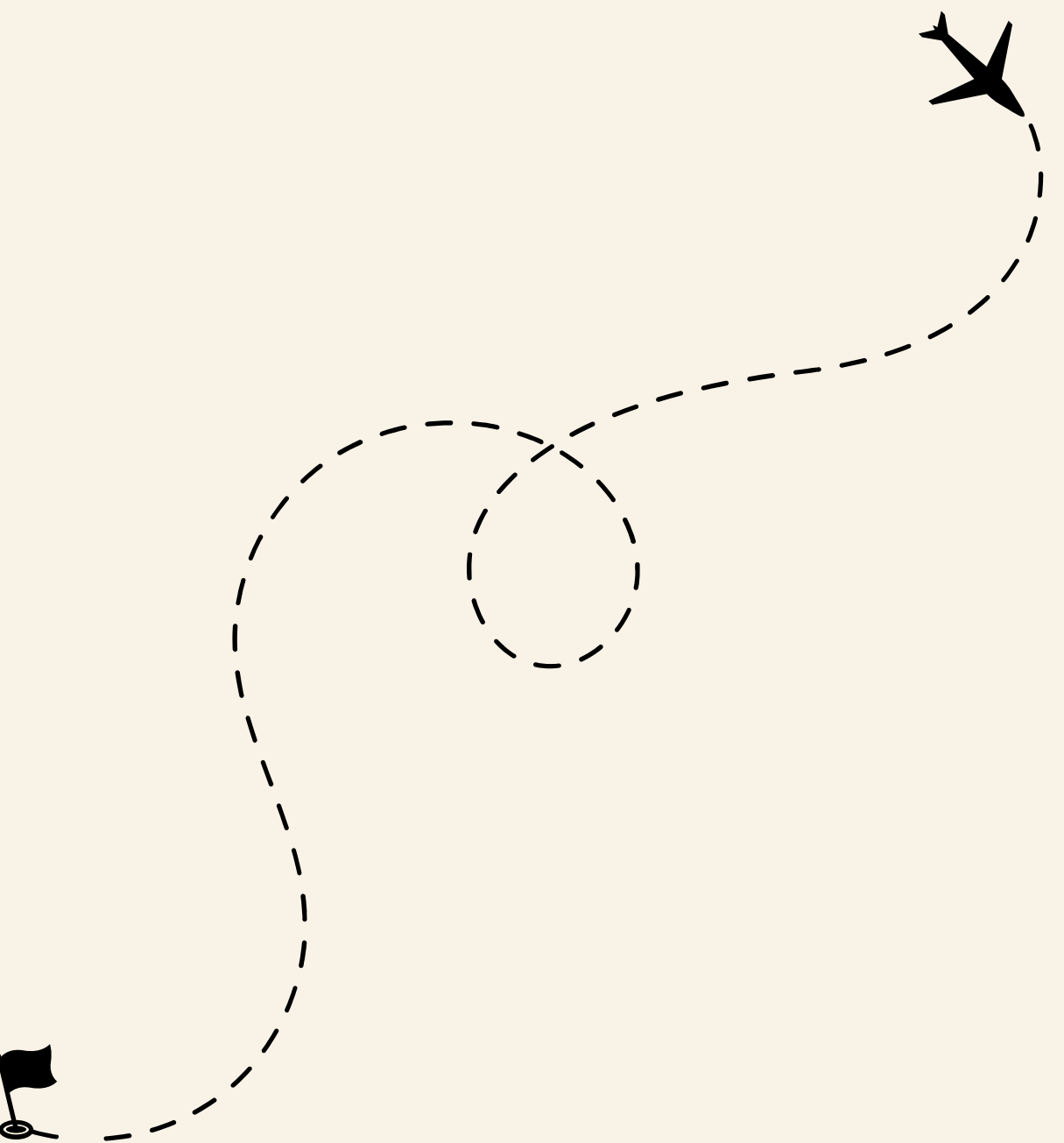


# FLIGHT BOOKING SYSTEM



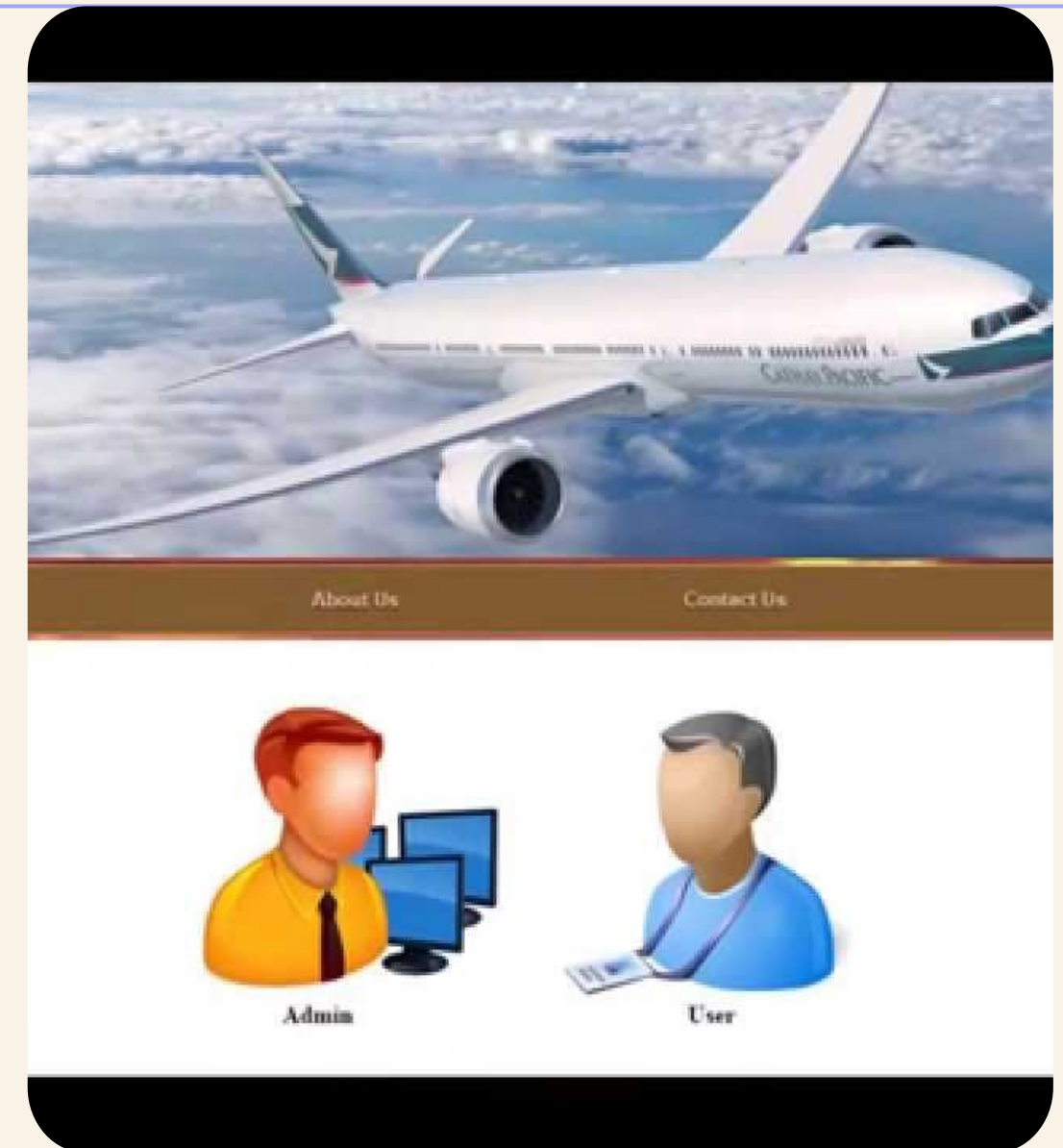
PREPARED BY:



AAKANKSHA  
ALISHA NASIR

# INTRODUCTION:

The Flight Booking System was developed using only C++ programming language. This system is a user-friendly kind of system that can be easily modified. The system provide you an efficient way for booking a flight ticket. It will give you a convenient way to to record all the passenger information. By using this system it will lessen the problem for booking a wrong flight detail to the passenger.



# FEATURES

- *Can Add Passenger Details*
- *Can Choose Flight Destination*
- *Auto Generate PNR Status*
- *Can Allow to Cancel Book Flight*

```
Welcome To Airline Flight Booking System
```

```
1.Book Flight<1>  
2.Cancel Fight<2>  
3.Check Ticket<3>  
4.Exit<4>
```

```
Please enter your choice: _
```

# FLIGHT MENU BOOKING

```
1.Domestic Fights(1)
2.International Flights(2)

Please enter your option
1
Enter Date of Flight(DDMMYY).Please enter a valid date.
062922
1.New York(1)    2.New Jersey(2)    3.Sydney(3)    4.Los Angeles(4)

    Enter Source
1
    Enter destination
2

                Flights Found

Airline:      Departure:      Arrival:      Price:      Category:
1.Eagle(1)    08:00          11:05         $100        Refundable
2.Falcon(2)   14:00          17:05         $100        Refundable
3.Jet Speed(3) 19:00          22:05         $100        Refundable

Enter Date of Flight(DDMMYY).Please enter a valid date.
New York(1)
Enter Source" << endl;
```

# PASSENGER FLIGHT DETAIL

```
"C:\Users\Mark\Desktop\Airline Flight Booking System in C++\Airline Flight Bo...
Flights Found
Airline:      Departure:      Arrival:      Price:      Category:
1.Eagle<1>    08:00        11:05        $100        Refundable
2.Falcon<2>   14:00        17:05        $100        Refundable
3.Jet Speed<3> 19:00        22:05        $100        Refundable

Enter your choice
1

Flight selected:
Eagle
Departure Time : 08:00
Arrival Time: 11:05

Enter passenger details
First Name:John
Last Name:Smith

Gender:
Male-press:1::
Female-press:2::1
Age:21
```

# CHECK TICKET MENU

```
1.Domestic Flights<1>
2.International Flights<2>

Please enter your option
1
Please enter your PNR no.:
1
PNR:1
Flight:Eagle
Name:John Smith
DOJ:62922
Departure Time:8:00
Arrival Time:11:05
Your ticket

Do you wish to continue:(y/Y)
_
```

```
nEnter Date of Flight(DDMMYY)." << "Please enter a valid date." << endl;
```

# CANCELED BOOK MENU

```
1.Domestic Flights(1)
2.International Flights(2)

Please enter your option
1
Please enter your PNR no.:
1
PNR:1
Flight:Eagle
Name:John Smith
DOJ:62922
Departure Time:8:00
Arrival Time:11:05
Your Above ticket is being canceled:
Amount refunded: $100

Do you wish to continue:(y/Y)
```

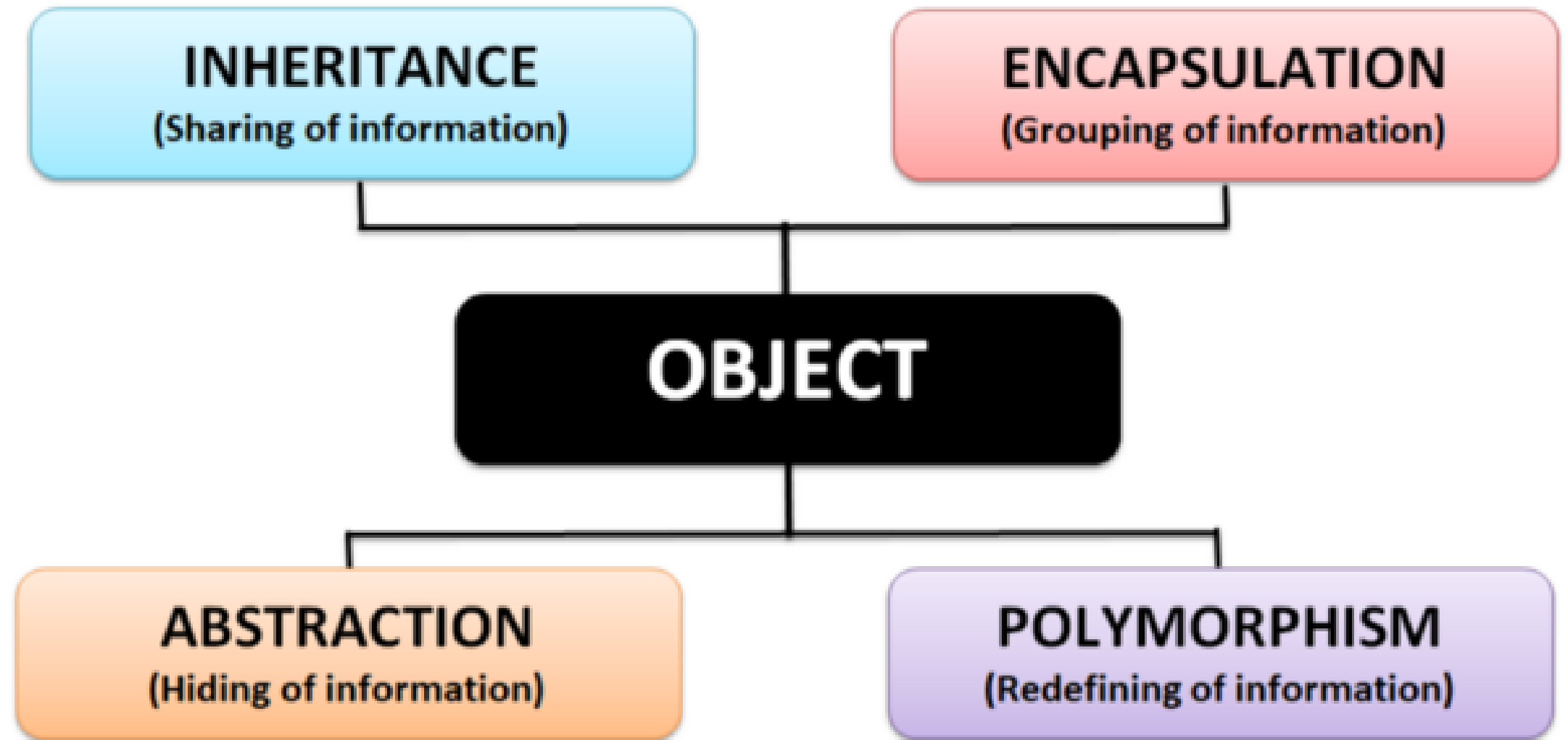
```
r Date of Flight(DDMMYY)." << "Please enter a valid date." << endl;
```

# ENCAPSULATION

# ABSTRACTION

# INHERITENCE

# POLYMORPHISM





# ENCAPSULATION

- **Classes:** `d_booking`, `i_booking`, `passenger`, and `payment` encapsulate related data and behaviors.
- **Data and Methods:** Data members like `pnr`, `f_d`, `toja`, `tojd`, `doj`, and methods like `d_pnr()`, `j_detail()`, `select_flight()`, `p_detail()`, etc., are encapsulated within their respective classes.

*Encapsulation ensures data security and restricts direct access to data.*



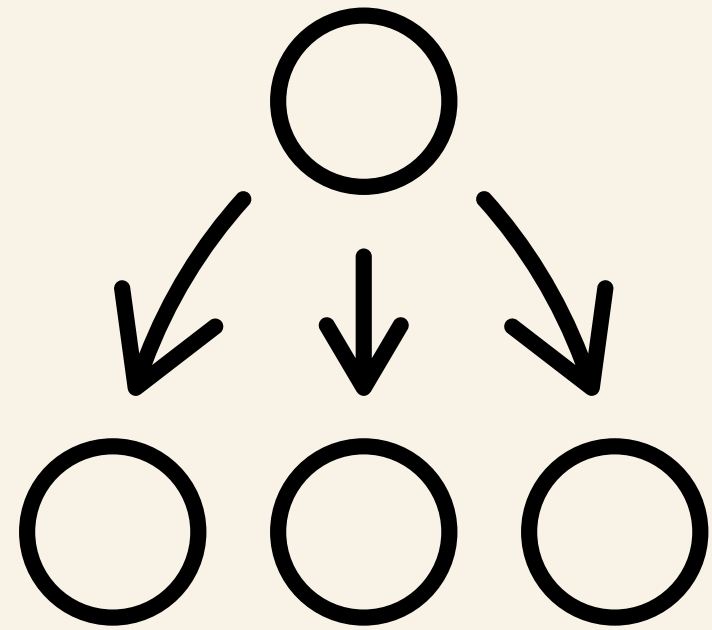
# ABSTRACTION

- **Class Interfaces:** Each class provides a clear interface (**public section**) for interacting with its methods while hiding the internal implementation details (**private or protected sections**).
- **Methods:** Methods like **pay\_detail()**, **j\_detail()**, **select\_flight()**, etc., provide abstracted interfaces to interact with their functionalities without revealing their inner workings.

*Example:  
**p\_detail()** abstracts the process of collecting passenger details,  
**pay\_detail()** abstracts the payment process, etc.*



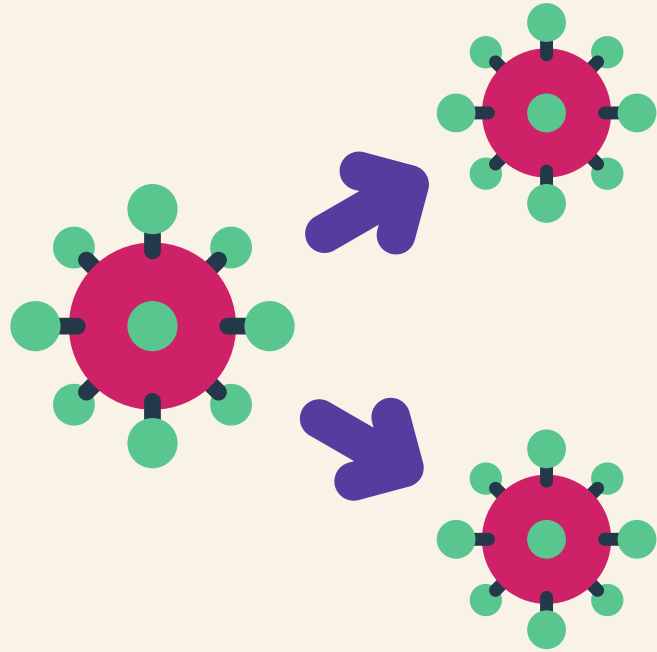
# INHERITANCE



**Example:**  
***class passenger : public d\_booking, public i\_booking** demonstrates inheritance, enabling **passenger** to access **j\_detail()** and other methods from **d\_booking** and **i\_booking**.*

- Class Inheritance: The **passenger class inherits** from **both d\_booking and i\_booking**, acquiring their functionalities.
- Reuse of Code: The passenger class reuses the methods and attributes from **d\_booking and i\_booking**, promoting code reuse.

# POLYMORPHISM



***j\_detail()** and **j\_detaili()** provide different flight details based on source and destination, **select\_flight()** and **select\_flighti()** exhibit different flight selections based on user input.*


**Method Overriding:** Functions like **disp()** and **dispi()** are overridden in the **passenger class** to provide specific implementations for different types of flights (**d\_booking** and **i\_booking**). This is an example of runtime polymorphism achieved through function overriding.

# MAIN FUNCTION

## Initialization:

- It initializes instances of classes: **d\_booking, i\_booking, passenger, and payment.**
- Initializes variables **ch, ch1, n, and input.**

## Menu Loop (do-while):

- **Displays a menu of options:**
  - a. **Book a Flight** 
  - b. **Cancel a Flight**
  - c. **Check Ticket**
  - d. **Exit**

## Booking a Flight (case 1):

- Prompts the user to select between domestic and international flights.
- Calls functions related to **booking flights, collecting passenger details, payment details, and storing the booking information into the respective files (createfile or createfilei).**

# MAIN FUNCTION

## Cancel a Flight (case 2):

- Similar to the booking process, asks the user to choose between domestic and international flights.
- Asks for the PNR number of the ticket to be canceled and then calls either **cancelticket** or **cancelticketi** functions accordingly to cancel the ticket.

## Check Ticket (case 3):

- Similar to the previous cases, prompts the user to choose between domestic and international flights.
- Asks for the PNR number of the ticket to be checked and then calls either **checkticket** or **checkticketi** functions to **display ticket details**.

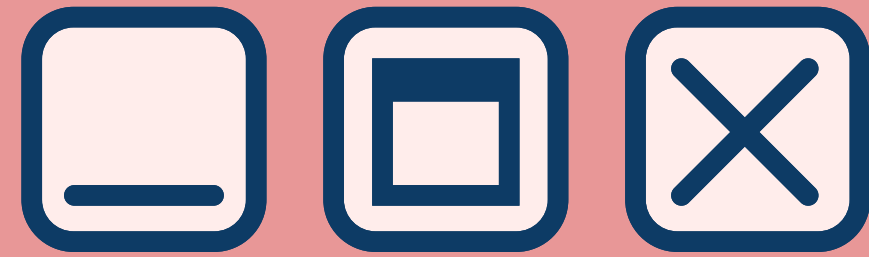
# MAIN FUNCTION

## Exit (case 4):

- Exits the program when the user selects this option.

## Loop Continuation:

- After each operation, the program asks the user if they want to continue (do-while loop condition).
- **If the user inputs 'Y' or 'y', the loop continues. Otherwise, the program ends.**



★ Thanks y'all ★

★

★

