Lab 3 : CLASSES & OBJECTS

1. OBJECTIVE

The objective of Lab3 is to practise on processing array of objects.

2. INTRODUCTION

Very often a program needs to process information in an array of objects in several ways. We will practise on writing array of objects in this lab.

Before you start your task, below is a brief introduction to Java syntax and constructs for arrays :

- **Arrays in Java (and C/C++) are indexed from *0* to SIZE-1.**

- **Creating** arrays with 12 elements of integer type that stores the number of days in each month (initialized to 0):

```
int [ ] days ;
days = new int[12] ;     // 2 lines code
```

OR

```
int [ ] days = new int[12];  // one line code
```

- **Assign a value to an array element :**

```
e.g.    days[0] = 31 ;
        days[1] = 28 ;
```

OR

```
int[ ] days = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 } ;
```

- **Accessing array elements:**

```
Ex 1.  sales[0] = 143.50 ;

Ex 2.  if ( sales[23] == 50.0 ) ...

Ex 3.  sales[8] = sales[5] – sales[2] ;

Ex 4.  while ( sales[364] != 0.0 ) {...}

Ex 5.  for ( int i=0 ; i < 365 ; i++ )
           sales[ i ] = 0 ;
```

- **Java stores the size of the array automatically in an instance variable named *length***

e.g.

```
public class PrintingDays
{
  public static void main( String[] args )
  {
    int i;
    int[] days = {31,28,31,30,31,30,31,31,30,31,30,31};
    // print the number of days in each month
    for ( i=0 ; i < days.length ; i++ ) // traverse array
      System.out.println( "Month " + (i+1) +
        " has " +  days[i] + " days." );
  }
}
```

```
Program Output
Month 1 has 31 days.
Month 2 has 28 days.
...
Month 12 has 31 days.
```

- **Creating an array of *objects*:**

e.g. Rectangle[ ] rectArray = new Rectangle[5]; // allocate 5 spaces to hold the elements
rectArray[0] = new Rectangle(15,10); // assign Rectangle object to the space
rectArray[1] = new Rectangle(10,20); //
……

3. *Your Tasks for this LAB*

The Colossus Airlines fleet consists of one plane with a seating capacity of 12. It makes one flight daily. In this experiment, you are required to write a seating reservation application program. The problem specification is given below:

A. Write a class *PlaneSeat* that has the following features. Each *PlaneSeat* object should hold a seat identification number (seatId), a marker (assigned) that indicates whether the seat is assigned and the customer number (customerId) of the seat holder. The class diagram is given below:

PlaneSeat
- seatId: int - assigned: boolean - customerId: int
+ PlaneSeat(seat_id: int) + getSeatID(): int + getCustomerID(): int + isOccupied(): boolean + assign(cust_id: int): void + unAssign(): void

where

PlaneSeat() - is the constructor for the class.
getSeatID() – a get method that returns the seat number.
getCustomerID() – a get method that returns the customer number.
isOccupied() – a method that returns a boolean on whether the seat is occupied.
assign() – a method that assigns a seat to a customer.
unAssign() – a method that unassigns a seat.

Implement the class PlaneSeat.

B. Write a class *Plane* that comprises 12 seats. The class should create an array of 12 objects from the class *PlaneSeat*.

The class diagram is given below:

Plane
- seat: PlaneSeat[ ] - numEmptySeat: int
+ Plane() - sortSeats() : PlaneSeat[ ] + showNumEmptySeats(): void + showEmptySeats(): void + showAssignedSeats(bySeatId : boolean): void + assignSeat(seatId : int, cust_id : int): void + unAssignSeat(seatId : int): void

where

seat – instance variable containing information on the seats in the plane. It is declared as an array of 12 seat objects.
numEmptySeat – instance variable containing information on the number of empty seats.
Plane() – a constructor for the class Plane.
sortSeats() – a method to sort the seats according to ascending order of customerID. A copy of the original seat array is used for sorting instead of the original.
showNumEmptySeats() – a method to display the number of empty seats.
showEmptySeats() – a method to display the list of empty seats.
showAssignedSeat() – a method to display the assigned seats with seat ID and customer ID. If ***bySeatId*** is true, the order will be by seatID, else order is by customerID.
assignSeat() – a method that assigns a customer ID to an empty seat .
unAssignSeat() – a method that unassigns a seat.

Implement the class Plane.

C. Write an application class *PlaneApp* that implements the seating reservation program.

The class *PlaneApp* should be able to support the following:
(1) Show the number of empty seats
(2) Show the list of empty seats
(3) Show the list of customers together with their seat numbers in the order of the seat numbers
(4) Show the list of customers together with their seat numbers in the order of the customer ID
(5) Assign a customer to a seat
(6) Remove a seat assignment

The menu should also contain option (7) (i.e. quit) for terminating the program. After the user selects a particular option, the corrsponding operation will be executed. If the selected option is not (7), then the program shows the menu for user selection again. This application does not need to save data into a file between runs.

Important:
Remember to do all the programming inside the sub-directory *lab6* and name the source codes as **PlaneSeat.java**, **Plane.java** and **PlaneApp.java** and name the compiled codes as **PlaneSeat.class**, **PlaneSeat.class** and **PlaneApp.class**.

Test Data:
Test your application program with the following data:
1. Assign a customer to a seat with SeatID=10, CustomerID = 10001.
2. Assign a customer to a seat with SeatID=12, CustomerID = 10002.
3. Assign a customer to a seat with SeatID=8, CustomerID = 10003.
4. Show the the list of customers together with their seat numbers in the order of the seat numbers.
5. Show the number of empty seats.
6. Show the list of empty seats.
7. Assign (attempt) a customer to a seat with any existing CustomerID, and SeatID. (Should give a warning message!)
8. Remove the seat assignment with SeatID=10.

9. Assign (attempt) a customer to a seat with SeatID = 12.
10. Remove the seat assignment with SeatID=12.
11. Show the list of customers together with their seat numbers in the order of the seat numbers.
12. Show the number of empty seats.
13. Show the list of empty seats.
14. Quit

Expected outputs:

```
(1) Show number of empty seats
(2) Show the list of empty seats
(3) Show the list of seat assignments by seat ID
(4) Show the list of seat assignments by customer ID
(5) Assign a customer to a seat
(6) Remove a seat assignment
(7) Exit

  Enter the number of your choice: 5
Assigning Seat ..
  Please enter SeatID: 10
  Please enter Customer ID: 10001
Seat Assigned!

  Enter the number of your choice: 5
Assigning Seat ..
  Please enter SeatID: 12
  Please enter Customer ID: 10002
Seat Assigned!

  Enter the number of your choice: 5
Assigning Seat ..
  Please enter SeatID: 8
  Please enter Customer ID: 10003
Seat Assigned!

   Enter the number of your choice: 3
The seat assignments are as follow:
SeatID 8 assigned to CustomerID 10003.
SeatID 10 assigned to CustomerID 10001.
SeatID 12 assigned to CustomerID 10002.

   Enter the number of your choice: 4
The seat assignments are as follow:
SeatID 10 assigned to CustomerID 10001.
SeatID 12 assigned to CustomerID 10002.
SeatID 8 assigned to CustomerID 10003.
  Enter the number of your choice: 1
There are 9 empty seats

  Enter the number of your choice: 2
The following seats are empty:
SeatID 1
SeatID 2
SeatID 3
SeatID 4
SeatID 5
SeatID 6
SeatID 7
SeatID 9
SeatID 11

  Enter the number of your choice: 5
Assigning Seat ..
  Please enter SeatID: 8
  Please enter Customer ID: 10004
Seat already assigned to a customer.

  Enter the number of your choice: 6
  Enter SeatID to unassign customer from: 10
Seat Unassigned!
```

```
  Enter the number of your choice: 5
Assigning Seat ..
  Please enter SeatID: 12
  Please enter Customer ID: 10005
Seat already assigned to a customer.

  Enter the number of your choice: 6
  Enter SeatID to unassign customer from: 12
Seat Unassigned!

  Enter the number of your choice: 3
The seat assignments are as follow:
SeatID 8 assigned to CustomerID 10003.

   Enter the number of your choice: 1
There are 11 empty seats

  Enter the number of your choice: 2
The following seats are empty:
SeatID 1
SeatID 2
SeatID 3
SeatID 4
SeatID 5
SeatID 6
SeatID 7
SeatID 9
SeatID 10
SeatID 11
SeatID 12

  Enter the number of your choice: 7
```