**Lab 4 (Exceptions)**

|  |
| --- |
| **Reservation** |
|  |
| + reserveSeat(seatNumber : int) : void  + reserveSeats(seatN umbers : int[]) : void  + cancelReservation(seatNumber : int) : void  + cancelReservations(seatNumbers : int[]) : void |

|  |
| --- |
| **MovieReservation** |
| - seats : int[] |
| + MovieReservation(seats : int[])  + reserveSeat(seatNumber : int) : void  + reserveSeats(seatN umbers : int[]) : void  + cancelReservation(seatNumber : int) : void  + cancelReservations(seatNumbers : int[]) : void  + toString() : String |

|  |
| --- |
| **TravelTicketReservation** |
| - seats : int[] |
| + TravelTicketReservation(seats : int[])  + reserveSeat(seatN umber : int) : void  + reserveSeats(seatN umbers : int[]) : void  + cancelReservation(seatNumber : int) : void  + cancelReservations(seatNumbers : int[]) : void  + toString() : String |

In this lab, you will be creating a Movie Reservation System and a Travel Ticket Reservation that will implement an interface, and a SeatException class.

The Reservation interface has five methods that need to be implemented which are reserveSeat, reserveSeats, cancelReservation, and cancelReservations.

The SeatException class will extend the Java Exception class. Its constructor will take a string as a parameter and call the superclass' constructor passing the string to it as a parameter.

The MovieReservation class and the TravelTicketReservation class will implement the interface's methods. They also have a private variable which is an array of integers representing the seats. The constructor should initialize this variable.

The reserveSeat method will take an integer named seatNumber as a parameter and sets seats[seatNumber] to one indicating that the seat is reserved. If the seat is already reserved, a SeatException should be thrown.

The reserveSeats method will take an integer array named seatNumbers as a parameter and sets all the seat numbers in the array to one indicating that the seat is reserved. If the seat is already reserved, a SeatException should be thrown.

The cancelReservation method will take an integer named seatNumber as a parameter and sets seat[seatNumber] to zero indicating that the seat is empty. If the seat is already empty, a SeatException should be thrown.

The cancelReservations method will take an integer array named seatNumbers as a parameter and sets all the seat numbers in the array to zero indicating that the seat is empty. If the seat is already empty, a SeatException should be thrown.

You should write a tester class that will have an instance of MovieReservation and TravelTicketReservation and then call the previous methods. You should use try and catch blocks to handle the exceptions that might be thrown in these methods.

For each instance, the tester class should read input from the user by using a scanner. For simplicity, make the number of seats ten. First, the user must enter ten numbers which are either zero or one, so use a for loop to read ten numbers and set the value of seats[i] to the read value.

Then, read an int value from the user and call reserveSeat passing the value as a parameter. Then, read four int values from the user and call reserveSeats passing the values as a parameter. Then do the same thing for cancelReservation and cancelReservations.

There are three types of exceptions that can happen when calling these functions. The first one is a SeatException as discussed earlier. The second one is an IndexOutOfBoundsException which will occur if the user enters a number that is less than zero or greater that seats.length. The third one is an InputMismatchException if the user enters a value which is not an integer.

After each call to the functions, you should print the seats array, or print "Seat number does not exist" if an IndexOutOfBoundsException occurs, or "Seat already reserved or empty" if a SeatException occurs, or "Input should be an Integer" if an InputMismatchException occurs.

Solution: <http://www.javaproblems.com/2013/12/creating-exception-application-in-java.html>