**Invalid Date Exception**

Write a program to check whether the input date is in the valid format or not.  
  
Create  **Program**class with **Main** method, get the starting and ending date-time of the Stage Event show from the user.  
If the input date-time is given in the specified format, then display the starting and ending date-time.  
Use try catch block to catch the exception and display the error message.

**Input and Output Format:**

The format for the date is  **dd/MM/yyyy hh:mm:ss tt**  
Refer sample input and output for formatting specifications.

**[All text in bold corresponds to the input and the rest corresponds to output.]**  
  
**Sample Input and Output 1:**  
Enter the start date(dd/MM/yyyy hh:mm:ss tt):  
**18/01/2018 09:00:00 AM**  
Enter the end date(dd/MM/yyyy hh:mm:ss tt):  
**18/01/2018 12:30:00 PM**  
Starting Date: 18/01/2018 09:00:00 AM  
Ending Date: 18/01/2018 12:30:00 PM  
  
**Sample Input and Output 2:**  
Enter the start date(dd/MM/yyyy hh:mm:ss tt):  
**01/25/2017 09:00:00 AM**  
Enter the end date(dd/MM/yyyy hh:mm:ss tt):  
**01/25/2017 11:00:00 AM**  
Invalid Date Format...  
  
**Sample Input and Output 3:**  
Enter the start date(dd/MM/yyyy hh:mm:ss tt):  
**25/01/2018 14:00:00**  
Enter the end date(dd/MM/yyyy hh:mm:ss tt):  
**25/01/2018 16:00:00**  
Invalid Date Format...

**Seat Booking**

Write a program to book the seat and display the result.

**[Note : Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Follow the naming conventions as mentioned. Create separate classes in separate files.]**  
  
Consider class **SeatNotAvailableException**which extends **Exception** class.  
Include a constructor with a string argument "Message", pass the Message to the base class constructor.  
  
Consider **Program**class with **Main**method, get the details of the availability of seats from the user and the seat number for booking.  
Use try catch block to catch the exception and display the error message.  
If the seat is available, display success message as “**Booked successfully**”. If the seat is already booked then display “**Seat booked already**”.  
  
**Note:**In the seat availability string, 0 indicates the seat is available whereas 1 indicates the booked seat.

**Input and Output Format:**  
Refer sample input and output for formatting specifications.

**[All text in bold corresponds to the input and the rest corresponds to output.]**  
  
**Sample Input and Output 1 :**

Enter the booking details  
**001001**  
Enter the seat number to book  
**2**  
Booked successfully  
  
**Sample Input and Output 2 :**

Enter the booking details  
**101001**  
Enter the seat number to book  
**3**  
Seat booked already  
  
**Sample Input and Output 3 :**

Enter the booking details  
**1001100**  
Enter the seat number to book  
**8**  
Array index is out of range.

**Mobile Number Validation**

Write a program to validate the user contact detail and display the user details or exception based on the mobile number.  
  
**[Note : Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Follow the naming conventions as mentioned. Create separate classes in separate files.]**  
  
Consider the class **ContactDetail** with following private attributes.

|  |  |
| --- | --- |
| **Data Type** | **Attribute** |
| long | \_mobile |
| long | \_alternateMobile |
| long | \_landLine |
| string | \_email |
| string | \_address |

The methods for **getters, setters** and **constructors** are given in the template code.  
  
Override**ToString()**method to display the contact details. Include the statement “**Contact Details:**” in this method.  
  
Consider the class **ContactDetailBO** and define the following method,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void Validate(ContactDetail cd) | In this method, check whether the mobile number and alternate mobile number is same or not. If it is same, then throw a DuplicateNumberException with message as “**Exception: Same Mobile number and Alternate Mobile number.**”. Otherwise display the contact details. |

Consider class **DuplicateNumberException** which extends the **Exception** class.  
Include a constructor with a string argument "Message", pass the Message to the base class constructor.  
  
Consider **Program** class with **Main** method, read the contact details from the user.  
Validate the mobile number and alternate mobile number and display the corresponding output.  
Use try and catch blocks to handle the Exception.  
  
  
**Input and Output Format:**  
Refer sample Input and output for formatting specifications.    
**All text in bold corresponds to input and rest corresponds to output.**  
  
**Sample Input and Output 1:**  
Enter the mobile number:  
**9546574125**  
Enter the alternate mobile number:  
**8412365785**  
Enter the landline number:  
**914522334578**  
Enter the email id:  
**joel@gmail.com**  
Enter the address:  
**Madurai**  
Contact Details:  
Mobile: 9546574125  
Alternate Mobile: 8412365785  
LandLine: 914522334578  
Email Id: joel@gmail.com  
Address: Madurai  
  
**Sample Input and Output 2:**  
Enter the mobile number:  
**8341257412**  
Enter the alternate mobile number:  
**8341257412**  
Enter the landline number:  
**914272547845**  
Enter the email id:  
**sofia@gmail.com**  
Enter the address:  
**Salem**  
Exception: Same Mobile number and Alternate Mobile number.