**Duplicate mobile number exception**

Write a java program to find the duplicate mobile number using the exception handling mechanism.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
Create a Class called **ContactDetail** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| mobile | String |
| alternateMobile | String |
| landLine | String |
| email | String |
| address | String |

Include getters and setters.  
Include default and parameterized constructors.  
Format for a parameterized constructor is **ContactDetail(String mobile, String alternateMobile,String landLine, String email, String address)**  
  
Override the **toString()** method to display the Contact details as specified.  
  
Create a class called **ContactDetailBO** with following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| static void validate(String mobile,String alternateMobile) | This method throws DuplicateMobileNumber exception if the mobile and alternateMobile are the same. |

Create a driver class called **Main**. In the Main method, obtain inputs from the user. Validate the mobile and alternateMobile and display the ContactDetail if no exception occurs else handle the exception.  
  
Pass the exception message as "**Mobile number and alternate mobile number are same**". If mobile and alternateMobile are the same.  
  
**Input and Output format:**  
Refer to sample Input and Output for formatting specifications.  
  
**Note: All text in bold corresponds to the input and rest corresponds to the output.**  
  
**Sample Input and Output 1:**  
  
Enter the contact details  
**9874563210,9874563210,0447896541,johndoe@abc.in,22nd street kk nagar chennai**  
DuplicateMobileNumberException: Mobile number and alternate mobile number are same  
  
**Sample Input and Output 2:**  
  
Enter the contact details  
**9874563210,9876543211,0447896541,johndoe@abc.in,22nd lane RR nagar kovai**  
Mobile:9874563210  
Alternate mobile:9876543211  
LandLine:0447896541  
Email:johndoe@abc.in  
Address:22nd lane RR nagar kovai

public class DuplicateMobileNumberException extends Exception{

public DuplicateMobileNumberException(String s)

{

super(s);

}

public String toString(){

return "DuplicateMobileNumberException: Mobile number and alternate mobile number are same ";

}

}

public class ContactDetailBO {

static void validate(String mobile, String alternateMobile) throws DuplicateMobileNumberException {

if (mobile.equals(alternateMobile)) {

throw new DuplicateMobileNumberException("DuplicateMobileNumberException :");

}

}

}-----------

public class ContactDetail {

//Your code here

private String mobile;

private String alternateMobile;

private String landLine;

private String email;

private String address;

public ContactDetail(String mobile, String alternateMobile, String landLine, String email, String address) {

this.mobile = mobile;

this.alternateMobile = alternateMobile;

this.landLine = landLine;

this.email = email;

this.address = address;

}

public ContactDetail()

{

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

public String getAlternateMobile() {

return alternateMobile;

}

public void setAlternateMobile(String alternateMobile) {

this.alternateMobile = alternateMobile;

}

public String getLandLine() {

return landLine;

}

public void setLandLine(String landLine) {

this.landLine = landLine;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String toString()

{

return "Mobile:"+this.mobile+"\nAlternate mobile:"+this.alternateMobile+"\nLandLine:"+this.landLine+"\nEmail:"+this.email+"\nAddress:"+this.address;

}

}

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the contact details");

String st = br.readLine();

String[] str = st.split(",");

ContactDetail userInfo = new ContactDetail(str[0], str[1], str[2], str[3], str[4]);

ContactDetailBO BO = new ContactDetailBO();

try {

BO.validate(str[0], str[1]);

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

} catch (DuplicateMobileNumberException e) {

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

}

}

}

**SeatNotAvailableException**

An organization is organizing a charity fate for the well being of poor kids. Since the manager was running short on time, he asked you to help him with the ticket bookings. You being from a programming background decide to design a program that asks the user about the seat number they want. Seat booking details are stored in an array. If the seat number requested is available booking should be done else print the message " **SeatNotAvailableException**". If the seat number requested is not in the range throws an exception **ArrayIndexOutOfBoundsException.**  
  
Create a class **SeatNotAvailableException** that extends Exception.  
  
Create an array of size n\*n (n rows each with n seats) which is got from the user. Get the tickets to be booked from the user and handle any exception that occurs in **Main**Class. (Take seat numbers from 0 to (n\*n)-1)  
  
**Note**: Vacant seats are denoted by (**0)**and booked seats by (**1)**. Show message as "**Already Booked**" as a Custom exception.  
  
**Input and Output format:**  
Refer sample Input and Output for formatting specifications.t of the output.  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input and Output 1:**  
  
Enter the number of rows and columns of the show:  
**3**  
Enter the number of seats to be booked:  
**2**  
Enter the seat number 1  
**8**  
Enter the seat number 2  
**0**  
The seats booked are:  
1 0 0  
0 0 0  
0 0 1  
  
**Sample Input and Output 2:**  
  
Enter the number of rows and columns of the show:  
**3**  
Enter the number of seats to be booked:  
**2**  
Enter the seat number 1  
**9**  
java.lang.ArrayIndexOutOfBoundsException: 9  
The seats booked are:  
0 0 0  
0 0 0  
0 0 0  
  
**Sample Input and Output 3:**  
  
Enter the number of rows and columns of the show:  
**4**  
Enter the number of seats to be booked:  
**3**  
Enter the seat number 1  
**15**  
Enter the seat number 2  
**14**  
Enter the seat number 3  
**15**  
SeatNotAvailableException: Already Booked  
The seats booked are:  
0 0 0 0  
0 0 0 0  
0 0 0 0  
0 0 1 1

public class SeatNotAvailableException extends Exception{

public SeatNotAvailableException(String s) {

super(s);

}

}   
 ----------------------------------------------------------------------------------------------------------

import java.util.Scanner;

import java.io.\*;

public class Main {

public static void main(String args[]) throws SeatNotAvailableException{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of rows and columns of the show:");

int n = sc.nextInt();

int size = (n\*n);

int s;

int a[] = new int[size];

int mat[][] = new int[n][n];

System.out.println("Enter the number of seats to be booked:");

int seats = sc.nextInt();

try {

for(int i=0;i<seats;i++) {

System.out.println("Enter the seat number "+(i+1));

s = sc.nextInt();

if(a[s]==0) {

a[s] =1;

int t=0;

for(int j=0;j<n;j++) {

for(int k=0;k<n;k++) {

mat[j][k] = a[t];

t++;

}

}

}else {

throw new SeatNotAvailableException("Already Booked");

}

}

}catch (Exception e) {

System.out.println(e);

}

finally {

System.out.println("The seats booked are:");

for(int i=0;i<n;i++) {

for(int j=0;j<n;j++) {

System.out.print(mat[i][j]+" ");

}

System.out.println();

}

}

}

}

**Parse Exception**

For our application, we would have obtained date inputs. If the user enters a different format other than specified, an **Invalid Date Exception** occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.  
  
Create a driver class called **Main**. In the main method, Obtain start time and end time for stage event show, if an exception occurs, handle the exception and notify the user about the right format.  
  
**Input format:**  
The input consists of the start date and end date.   
The format for the date is **dd-MM-yyyy-HH:mm:ss**  
  
**Output format:**  
Refer sample Input and Output for formatting specifications

**Note: All text in bold corresponds to the input and rest corresponds to the output.**  
  
**Sample Input and Output 1:**  
  
Enter the stage event start date and end date  
**27-01-2017-12**  
Input dates should be in the format 'dd-MM-yyyy-HH:mm:ss'  
  
**Sample Input and Output 2:**  
  
Enter the stage event start date and end date  
**27-01-2017-12:0:0  
28-01-2017-12:0:0**  
Start date:27-01-2017-12:00:00  
End date:28-01-2017-12:00:00

import java.util.\*;

import java.text.ParseException;

import java.text.SimpleDateFormat;

public class Main {

public static void main(String[] args) {

// your code here

String d,d1;

Scanner s=new Scanner(System.in);

try{

System.out.println("Enter the stage event start date and end date");

d=s.next();

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy-HH:mm:ss");

String d2=sdf.format(sdf.parse(d));

d1 =s.next();

String d3=sdf.format(sdf.parse(d1));

System.out.println("Start date:"+d2);

System.out.println("End date:"+d3);

}catch (ParseException e) {

System.out.println("Input dates should be in the format 'dd-MM-yyyy-HH:mm:ss'");

}

}

**Arithmetic Exception**

 An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at runtime, it disrupts the normal flow of the program. For example, there are 10 statements in your program and there occurs an exception at statement 5, the rest of the code will not be executed i.e. statement 6 to 10 will not run. If we perform exception handling, the rest of the statement will be executed. That is why we use exception handling.

For practice in exception handling, obtain the cost for 'n' days of an item and n as input and calculate the cost per day for the item. In case, zero is given as input for n, an arithmetic exception is thrown, handle the exception and prompt the user accordingly.

Create a driver class called **Main**. In the Main method, obtain input from the user and store the values in int type. Handle exception if one occurs.  
  
**Input format:**  
The first line of input is an integer which corresponds to the cost of the item for n days.  
The second line of input is an integer which corresponds to the value n.  
  
**Output format:**  
If the value of n is zero throws an exception.  
Otherwise, print the integer output which corresponds to the cost per day of the item.

**NOTE: All text in bold corresponds to the input and rest corresponds to the output.**  
  
**Sample Input  and Output 1:**  
  
Enter the cost of the item for n days  
**100**  
Enter the value of n  
**0**  
java.lang.ArithmeticException: / by zero  
  
**Sample Input and Output 2:**  
  
Enter the cost of the item for n days  
**100**  
Enter the value of n  
**20**  
Cost per day of the item is 5

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the cost of the item for n days");

int days = sc.nextInt();

System.out.println("Enter the value of n");

int num = sc.nextInt();

try{

int cost = days/num;

System.out.println("Cost per day of the item is "+cost);

}catch(ArithmeticException e){

System.out.println(e);

}

}

}

**ArrayIndexOutOfBoundsException**

The next prominent exception which you will see is ArrayIndexOutOfBoundsException. It occurs when the program tries to access the array beyond its size. As we know arrays have fixed size. So when you try to use array beyond its size it throws this exception. Let's try to handle this exception.  
  
Handling this exception will also prove to be good for our application. For example, if there are only 100 seats in the event and the user tries to book the 105th seat, it will throw this exception. So you must handle it to do a specific job.  
  
Create an array of size 100 and assume it as seat array. Get the tickets to be booked from the user and handle any exception that occurs in **Main** Class. At last display all the tickets booked.  
  
**Input and Output format:**  
The first line of input consists of an integer which corresponds to the number of seats to be booked.  
The next n lines of input consist of the integer which corresponds to the seat number.  
Refer to sample Input and Output for formatting specifications.  
  
**Note: All Texts in bold corresponds to the input and rest are output.  
  
Sample Input and Output 1:**  
  
Enter the number of seats to be booked:  
**5**  
Enter the seat number 1  
**23**  
Enter the seat number 2  
**42**  
Enter the seat number 3  
**65**  
Enter the seat number 4  
**81**  
Enter the seat number 5  
**100**  
The seats booked are:  
23  
42  
65  
81  
100  
  
**Sample Input and Output 2:**  
  
Enter the number of seats to be booked:  
**4**  
Enter the seat number 1  
**12**  
Enter the seat number 2  
**101**  
java.lang.ArrayIndexOutOfBoundsException: 100

import java.util.Scanner;

public class Main {

public static void main(String args[]) {

int[] seatArray = new int[100];

int n,count = 1;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of seats to be booked:");

n = sc.nextInt();

try{

for(int i=0;i<n;i++) {

System.out.println("Enter the seat number "+(i+1));

count = sc.nextInt();

seatArray[count-1] = count;

}

System.out.println("The seats booked are:");

for(int i = 0; i< 100; i++){

if(seatArray[i] != 0){

System.out.println(seatArray[i]);

}

}

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println(e);

}

}

}