Invalidname

Coding 100 points

DESCRIPTION

Write a Java program to evaluate **firstName** and **lastName** of the **User** class for the given input. For any given invalid input for **firstName/lastName** the program should thrown an **InvalidNameException** and handle them gracefully by displaying the cause of the abnormality.

The **constructor** and the **setFirstName** and **setLastName** methods of the **User** class must throw an **InvalidNameException** if the length of the ***firstName*** and ***lastName*** is less than six and has numbers/special characters.

```
ote : Create an user defined exception "InvalidNameException"
      class to handle any invalid firstName/lastName value as
      input.
      Create an User class with all the fields defined in the
      problem statement (userId and phoneNumber are int data
      types and rest of the fields are String type) and define
      para constructor, setters and getters, where the para
      constructor and setFirstName and setLastName methods
      should monitor for the invalid input. The program should
      monitor for these special characters -
                        !@#$%&*()'+,-./:;<=>?[]^_`{|}
      Create a Source class for the program to initiate the
      execution.
```

```
Sample Input Format :
```

1001

ravish@gmail.com

password

Please choose a language and write your code.

SUBMIT

ACCEPTED Score: **100 points** (details)

CODE INPUT OUTPUT Java 8 RUN CODE

```
1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6
7  class InvalidNameException extends RuntimeException{
8      private static final long serialVersionUID=111;
9      public InvalidNameException(){
10
11     }
12     public InvaldNameException(String msg){
13         super(msg);
14     }
15 }
16
17 class User{
18     private int userId;
19     private String emailId;
20     private String password;
21     private String firstName;
22     private String lastName;
23     private String city;
24     private String gender;
25     private int phoneNumber;
26     public User(){
27
28     }
29     public User(int userId,String emailId,String password,String firstName,String lastName,
30     String city,String gender,int phoneNumber){
31         this.userId=userId;
32         this.emailId=emailId;
33         this.password=password;
34         this.city=city;
35         this.gender=gender;
36         this.phoneNumber=phoneNumber;
37         Pattern p=Pattern.compile("[^A-Za-z]");
```

password

Ravish

Kumaran

Bangalore

Male

123456789

Sample Output Format :

User [userId=1001, emailId=ravish@gmail.com, password=password, firstName=Ravish, lastName=Kumaran, city=Bangalore, gender=Male, phoneNumber=123456789]

Sample Input Format :

1001

raj@gmail.com

password

Raj

Kumaran

Bangalore

Male

123456789

Sample Output Format :

The first name is too short.

```
        Matcher mnp=m.matcher(firstName);
        boolean spch=m.find();
        Pattern p1=Pattern.compile("[0-9]");
        Matcher m1=p1.matcher(firstName);
        boolean tsnum=m1.find();
        if(firstName.length()<4){
            throw new InvalidNameException("The first name is too short.");
        }

        else if(tsnum){
            throw new InvalidNameException("The first name contains digits.");
        }
        else if(spch){
            throw new InvalidNameException("The first name contains special characters.");
        }
        else
        {
            this.firstName=firstName;
        }
        Pattern pp=Pattern.compile("[^A-Za-z]");
        Matcher mp=pp.matcher(lasName);
        boolean spchp=mp.find();
        Pattern pp1=Pattern.compile("[0-9]");
        Matcher mp1=pp1.matcher(lastName);
        boolean tsnump=mp1.find();
        if(lastName.length()<4){
            throw new InvalidNameException("The last name is too short.");
        }

        else if(tsnump){
            throw new InvalidNameException("The last name contains digits.");
        }
         else if(spchp){
            throw new InvalidNameException("The last name contains special characters.");
        }
        else
        {
            this.lastName=lastName;
        }
    }

    @Override
    public String toString()
    {
        return "User [userId="+userId+", emailId="+emailId+", password="+password+
```

The first name is too short.

Sample Input Format :

1001

ravish@gmail.com

password

Ravish

Kumar

Bangalore

Male

123456789

Sample Output Format :

The last name is too short.

Sample Input Format :

1001

ravish@gmail.com

password

Ravish7

Kumaran

Bangalore

```
    }

    @Override
    public String toString()
    {
        return "User [userId="+userId+", emailId="+emailId+", password="+password+
        ", firstName="+firstName+", lastName="+lastName+", city="+city+", gender="+gender+
        ", phoneNumber="+phoneNumber+"]";
    }
}


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

        Scanner sc=new Scanner(System.in);
        int userId,phoneNumber;
        String emailId,password,firstName,lastName,city,gender;
        userId=Integer.parseInt(sc.nextLine());
        emailId=sc.nextLine();
        password=sc.nextLine();
        firstName=sc.nextLine();
        lastName=sc.nextLine();
        city=sc.nextLine();
        gender=sc.nextLine();
        phoneNumber=Integer.parseInt(sc.nextLine());
        try{
            User ui=new User(userId,emailId,password,firstName,lastName,city,gender,phoneNumber);
            System.out.println(ui);
        }
        catch(InvalidNameException e){
            System.out.println(e.getMessage());
        }
    }
}
```

6 revisions found for this solution. SHOW REVISIONS

Crud operations

DESCRIPTION

Write a Java program to evaluate the **User** in performing the CRUD Operations.

Create an user define exception class **InvalidRideException** to handle the exceptions whenever their is an abnormality caused for the given input.

Create an interface as **UserService** with abstract methods such as **requestRide**, **searchRide** and **displayRide** and provide **RideService** implementation class to define those abstract methods and handle **InvalidRideException** whenever user enters an invalid input for **requestRide** and **searchRide** methods. *(And also should handle for negative input.)*

The program should handle those exceptional conditions by displaying the cause of the abnormality.

```
: the requestRide method should check for the empty values
  only.
  The searchRide method should check whether fromLocation
  and toLocation are same or not and whether seatsLeft is
  more than totalSeats given.
```

```
Sample Input Format :
```

ACCEPTED Score: **100 points** (details)

CODE INPUT OUTPUT Java 8 RUN CODE

```
1  import java.io.*;
2  import java.util.*;
3  import java.text.*;
4  import java.math.*;
5  import java.util.regex.*;
6
7  class InvalidRideEntry extends Exception
8  {
9      public InvalidRideEntry(String msg)
10     {
11         super(msg);
12     }
13 }
14
15 interface UserService
16 {
17     abstract void requestRide(Ride r) throws InvalidRideEntry;
18     abstract void searchRide(Ride r) throws InvalidRideEntry;
19     abstract Ride displayRide(Ride r);
20 }
21
22 class RideService implements UserService
23 {
24     public RideService(){}
25
26     public void requestRide(Ride r) throws InvalidRideEntry
27     {
28         if(r.rideId.equals("") || r.fromLocation.equals("") || r.toLocation.equals("")
29             || r.seatsLeft.equals("") || r.totalSeats.equals("") || r.startTime.equals("
30             || r.endTime.equals("") || r.isStarted.equals("") || r.finished.equals("")
31         )
32         throw new InvalidRideEntry("Ride cannot be processed - Empty value.");
33     }
34
```

Sample Input Format :

R1001

Bangalore

Delhi

5

25

24-04-2022

30-04-2022

Yes

Yes

Sample Output Format :

Ride [rideId=R1001, fromLocation=Bangalore, toLocation=Delhi, seatsLeft=5, totalSeats=25, startTime=24-04-2022, endTime=30-04-2022, isStarted=Yes, finished=Yes]";

Sample Input Format :

R1001

Bangalore

5

25

```
    }

    public void searchRide(Ride r) throws InvalidRideEntry
    {
        if(r.fromLocation.equals(r.toLocation))
        throw new InvalidRideEntry("Invalid location/seats to process the ride.");

        if(Integer.parseInt(r.seatsLeft) > Integer.parseInt(r.totalSeats))
        throw new InvalidRideEntry("Invalid location/seats to process the ride.");

        if(Integer.parseInt(r.seatsLeft) < 0)
        throw new InvalidRideEntry("Invalid location/seats to process the ride.");

        if(Integer.parseInt(r.totalSeats) < 0)
        throw new InvalidRideEntry("Invalid location/seats to process the ride.");
    }

    public Ride  displayRide(Ride r)
    {
        System.out.println(r);

        return r;
    }
}

class Ride
{
    String rideId;
    String fromLocation;
    String toLocation;
    String seatsLeft;
    String totalSeats;
    String startTime;
    String endTime;
    String isStarted;
    String finished;

    public Ride(){}
    public Ride(String a,String b,String c,String d,String e,String f,
            String g,String h,String i) throws InvalidRideEntry
    {
```

Sample Input Format :

R1001

Bangalore

5

25

24-04-2022

30-04-2022

Yes

Yes

Sample Output Format :

Ride cannot be processed - Empty value.

Sample Input Format :

R1001

Bangalore

Bangalore

5

25

24-04-2022

```
62      String toLocation;
63      String seatsLeft;
64      String totalSeats;
65      String startTime;
66      String endTime;
67      String isStarted;
68      String finished;
69
70      public Ride(){}
71      public Ride(String a,String b,String c,String d,String e,String f,
72              String g,String h,String i) throws InvalidRideEntry
73      {
74          rideId = a; fromLocation = b; toLocation = c; seatsLeft = d;
75          totalSeats = e; startTime = f; endTime = g; isStarted = h; finished=i;
76      }
77
78      public String toString()
79      {
80          return "Ride [rideId="+rideId+", fromLocation="+fromLocation+", toLocation="
81              +toLocation+", seatsLeft="+seatsLeft+", totalSeats="+totalSeats+
82              ", startTime="+startTime+", endTime="+endTime+", isStarted="+isStarted+
83              ", finished="+finished+"]";
84      }
85  }
86
87  public class Source {
88      public static void main(String args[] ) throws Exception {}
89  }
90
91
92
```

10 revisions found for this solution. HIDE REVISIONS

Accepted • a minute ago • *latest submission*
100 points scored. 5 testcases passed out of 5 (details)

Partially accepted • 5 minutes ago

Location Capacity

CODE INPUT OUTPUT Java 8 RUN CODE

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class InvalidLocationException extends RuntimeException
{
    private static final long serialVersionUID=-1510453123644668900L;
    public InvalidLocationException(String message)
    {
        super(message);
    }
}
class InvalidCapacityException extends RuntimeException
{
    private static final long serialVersionUID=-1510453123644668900L;
    public InvalidCapacityException(String message)
    {
        super(message);
    }
}
class Ride
{
    private String rideId;
    private String fromLocation;
    private String toLocation;
    private String seatsLeft;
    private String totalSeats;
```

```
    private String seatsLeft;
    private String totalSeats;
    private String startDate;
    private String endDate;
    private String isStarted;
    private String finished;
   public Ride()
    {

    }
   public Ride(String rideId,String fromLocation,String toLocation,
   String seatsLeft,
   String totalSeats,String startDate,String endDate,
   String isStarted,String finished)throws InvalidLocationException,
   InvalidCapacityException
   {
       this.rideId=rideId;
       this.fromLocation=fromLocation;
       this.toLocation=toLocation;
       this.seatsLeft=seatsLeft;
       this.totalSeats=totalSeats;
       this.startDate=startDate;
       this.endDate=endDate;
       this.isStarted=isStarted;
       this.finished=finished;
   }

   public String getrideId()
   {
       return rideId;
   }
   public String getfromLocation()
```

```
58      }
59      public String getfromLocation()
60      {
61          return fromLocation;
62      }
63      public String gettoLocation()
64      {
65          return toLocation;
66      }
67      public String getseatsLeft()
68      {
69          return seatsLeft;
70      }
71      public String gettotalSeats()
72      {
73          return totalSeats;
74      }
75      public String getstartDate()
76      {
77          return startDate;
78      }
79      public String getendDate()
80      {
81          return endDate;
82      }
83      public String getisStarted()
84      {
85          return isStarted;
86      }
87      public String getfinished()
88      {
89          return finished;
```

```
88  {
89      return finished;
90  }
91  public void setrideId(String rideId)
92  {
93      this.rideId=rideId;
94  }
95  public void setfromLocation(String fromLocation )
96  {
97      this.fromLocation=fromLocation;
98  }
99  public void settoLocation(String toLocation)
100 {
101     if(toLocation==fromLocation)
102     {
103     throw new InvalidLocationException("Your destination is same as the origin.")
104     }
105     else{
106     this.toLocation=toLocation;
107     }
108 }
109 public void setseatsLeft(String seatsLeft)
110 {
111     this.seatsLeft=seatsLeft;
112 }
113 public void settotalSeats(String totalSeats)
114 {
115
116     .if(seatsLeft.length()<0)
117     {
118         throw new InvalidCapacityException("Invalid Input.Given seat capacity is
119     }
```

```
    }
}
public void setseatsLeft(String seatsLeft)
{
    this.seatsLeft=seatsLeft;
}
public void settotalSeats(String totalSeats)
{

    if(seatsLeft.length()<0)
    {
throw new InvalidCapacityException("Invalid Input.Given seat capacity is negative.")
    }
    else if(seatsLeft.length() > totalSeats.length())
    {
    throw new InvalidCapacityException("Your booking exceeds available capacity")
    }
    else
    {
    this.totalSeats=totalSeats;
    }
}
public void setstartDate(String startDate)
{
    this.startDate=startDate;
}
public void setendDate(String endDate)
{
    .this.endDate=endDate;
}
public void setisStarted(String isStarted)
{
```

```
135         this.endDate=endDate;
136     }
137     public void setisStarted(String isStarted)
138     {
139         this.isStarted=isStarted;
140     }
141     public void setfinished(String finished)
142     {
143         this.finished=finished;
144     }
145     @Override
146     public String toString()
147     {
148         return "Ride [rideId="+rideId+", fromLocation="+fromLocation+
149         ", toLocation="+toLocation+", seatsLeft="+seatsLeft+
150         ", totalSeats="+totalSeats+", startDate="+startDate+
151         ", endDate="+endDate+", isStarted="+isStarted+
152         ", finished="+finished+"]";
153     }
154
155 }
156
157 // Class name should be "Source",
158 // otherwise solution won't be accepted
159 public class Source {
160     public static void main(String args[] ) throws Exception {
161         /* Enter your code here. Read input from STDIN. Print output to STDOU
162         Scanner sc=new Scanner(System.in);
163         String rideId=sc.nextLine();
164         String fromLocation=sc.nextLine();
165         String toLocation=sc.nextLine();
166         String seatsLeft=sc.nextLine();
```

```
// Class name should be "Source",
// otherwise solution won't be accepted
public class Source {
    public static void main(String args[] ) throws Exception {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        Scanner sc=new Scanner(System.in);
        String rideId=sc.nextLine();
        String fromLocation=sc.nextLine();
        String toLocation=sc.nextLine();
        String seatsLeft=sc.nextLine();
        String totalSeats=sc.nextLine();
        String startDate=sc.nextLine();
        String endDate=sc.nextLine();
        String isStarted=sc.nextLine();
        String finished=sc.nextLine();
        try
        {
         Ride ride=new Ride(rideId,fromLocation,toLocation,seatsLeft,totalSeats,
         startDate,endDate,isStarted,finished);
        System.out.println(ride);

        }
        catch(InvalidLocationException e)
        {
            System.out.println(e.getMessage());
        }
        catch(InvalidCapacityException e)
        {
            System.out.println(e.getMessage());
        }

    }
```