**Q: Write a program to read a string and validate whether the given string is a valid color code based on the following rules:**

**- Must start with "#" symbol**

**- Must contain six characters after #**

**- It may contain alphabets from A-F or digits from 0-9**

**Include a class UserMainCode with a static method validateColorCode which accepts a string. The return type (integer) should return 1 if the color is as per the rules else return -1.**

**Create a Class Main which would be used to accept a String and call the static method present in UserMainCode.**

**Input and Output Format:**

**Input consists of a string.**

**Output consists of a string (Valid or Invalid).**

**Refer sample output for formatting specifications.**

**Sample Input 1:**

**#FF9922**

**Sample Output 1:**

**Valid**

**Sample Input 2:**

**#FF9(22**

**Sample Output 2:**

**Invalid**

**Question 1:**  
     
a)Create a Class **Main** with the following static methods:

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Boolean validateEmailId(String email) | Validate the email id based on the rules given below. Returns **true** ifemail id is valid else return **false** |
| static Boolean validateSpam(String email) | Checks for the spam domain name in the given email id. Returns **true** if the mail is spam else returns **false** |

b) While validating email id follow the below rules. The format of the email id is given below  
   
            username@domain.TLD  
  
where, TLD - Top Level Domain  
  
1. The email id should start only with alphabets(either uppercase or lowercase).  
2. The email username can contain alphabets(either uppercase or lowercase), numbers and the special characters (  .   and   \_    ).  
3. The email username should not contain any special characters other than**" . "** and **" \_ "**.  
4. After the username special character @ should present.  
5. The email domain should contain only alphabets(either uppercase or lowercase).  
6. After email domain, a value dot ( . ) should present.  
7. The email Top Level Domain should contain only alphabets(both uppercase and lowercase) and it should have only 2 to 6 characters.  
  
Example: **alpha\_Beta.01@google.com** is a valid email id.  
Since the username contain only alphabets, numbers and a special character ( \_ and . ), then the @ symbol is present. The domain name should contain only alphabets and the symbol dot( . ) and the Top Level Domain have only 3 characters.  
  
c) While checking for spam follow the below rules,  
  
If the domain name of the email id is one of the following then it is spam email id.  
             {**advancedpdfconverter, passwordmaster, smartfixer, downloadavideo** }  
  
**Note:** Print "**Email is valid**" if email is valid else print "**Email is invalid**". Print "**Email is spam**" if email is spam else print "**Email is not spam**".  
            All the above print statements are present in the main method.  
  
Menu:  
1.Validate Email  
2.Check for Spam  
  
**Sample Input and Output 1:**  
  
1.Validate Email  
2.Check for Spam  
Enter your choice:  
**1**  
Enter the email to be validated:  
**alpha\_Beta.02@mail.com**  
Email is valid  
  
**Sample Input and Output 2:**  
  
1.Validate Email  
2.Check for Spam  
Enter your choice:  
**1**  
Enter the email to be validated:  
**0alpf&sk@mail.in**  
Email is invalid  
  
**Sample Input and Output 3:**  
  
1.Validate Email  
2.Check for Spam  
Enter your choice:  
**2**  
Enter the email to be validated:  
**jane@abc.in**  
Email is not spam  
  
**Sample Input and Output 4:**  
  
1.Validate Email  
2.Check for Spam  
Enter your choice:  
**2**  
Enter the email to be validated:  
**jack@passwordmaster.com**  
Email is spam

**Question -2:**

Create a Class named as **Main**, which contains the following methods,

|  |  |
| --- | --- |
| **No** | **Method Name** |
| 1 | public static void parseName(BufferedReader br) |
| 2 | public static void isValidEmail(BufferedReader br) |
| 3 | public static void playContactNumber(BufferedReader br) |
| 4 | public  static void userLifeTime(BufferedReader br) |

**parseName ( ):**  
The customer name contains first name and last name separated by any “special character”. Given a name, display the first and last name.  
Examples of Special character like: **' , '** , **'@'** , **'#' , . . . . . (NOTE: There can be any special character).**  
  
**isValidEmail ( ):**  
Check if the email address entered is valid. A Valid email address would end with 3 domains (com, org, net) and would contain a “@”.  
  
**playContactNumber ( ):**  
Every contact number would be in the format (ISD Code-STD Code-Number). The size of the fields would be (3 digits – 4 digits – 10 digits). Given a contact number, find the sum of last 10 digits until it reaches to a single digit. Print the digit.  
  
**userLifeTime ( ):**  
Given a “createdOn” Date, print number of minutes before which the user was created. Assume the current date to be ‘28-07-2017 09:00’.  
Date format : "**dd/MM/yyyy HH:mm**";  
  
**Example for LifeTime calculation:**  
If the createdOn date is "28-07-2017 08:20", then the life time is 40 minutes.

**Sample Input and Output:**

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**1**

Enter name:

**John%Abraham\*Lincoln**

John Abraham Lincoln

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**1**

Enter name:

**Marc;Farnando**

Marc Farnando

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**2**

Enter E-mail id:

[**john.com**](http://john.com/)

Email is invalid

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**2**

Enter E-mail id:

**john@.com**

Email is invalid

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**2**

Enter E-mail id:

[**john@a.com**](mailto:john@a.com)

Email id is valid

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**3**

Enter contact number:

**844-7485-44784578459**

Contact number invalid

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**3**

Enter contact number:

**58-847-8547123654**

Contact number invalid

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**3**

Enter contact number:

**847-7845-9557898865**

Sum of contact number: 7

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**4**

Enter Created on date(dd-MM-yyyy HH:mm):

**25-07-2017 10:30**

Life time is: 4230 minutes

Menu

1. Parse Name

2. Valid Email

3. Play Contact Number

4. User Lifetime

5. Exit

**5**

**Question 3:**

a)Create a class Main with the following static methods:

Method Name Description

static Boolean validateMailId(String mailId)

Validate the mailId based on the rules given below. Returns true if mailId is valid else return false

static Boolean validateMobileNo(String mobileNo)

Validate the mobileNo based on the rules given below. Returns true if mobileNo is valid else return false

b) While validating mail id follow the below rules. The format of the mail id is given below

username@domain.TLD

Where, TLD - Top Level Domain

1. The mail Id should start only with alphabets(lowercase).

2. The username part of mail Id can contain alphabets(lowercase), numbers and the special characters ( . and \_ ).

3. The username part of the mail Id should not contain any special characters other than " . " and " \_ ".

4. After the username special character @ should present.

5. The mail Id domain should contain only alphabets(lowercase).

6. After the mail Id domain, a dot ( . ) should be present.

7. The mail id Top Level Domain should contain only alphabets(lowercase) and it should have only 2 to 6 characters.

Example: george\_poll.01@google.com is a valid mail id.

c)While validating mobile number follow the below rules,

1.The mobile number should have prefix as "+91"(international calling code).

2.The mobile number should have 10 digit numbers [0-9].

3.First digit of the mobile number should be greater than zero.

Example:+919876543210 is a valid mobile number.

Note:

Print "Mail Id is valid" if mail id is valid else print "Mail Id is invalid".

Print "Mobile number is valid" if mobile number is valid else print "Mobile number is invalid".

All the above print statements are present in the main method.

[All text in bold corresponds to input]

Sample Input and Output 1:

Validation

1)By mail id

2)By mobile number

Enter Your choice

1

Enter the mail id to be validated:

brandon@gmail.com

Mail Id is valid

Sample Input and Output 2:

Validation

1)By mail id

2)By mobile number

Enter Your choice

1

Enter the mail id to be validated:

james$20@gmail.com

Mail Id is invalid

Sample Input and Output 3:

Validation

1)By mail id

2)By mobile number

Enter Your choice

2

Enter the Mobile number to be validated:

+919876543210

Mobile number is valid

Sample Input and Output 4:

Validation

1)By mail id

2)By mobile number

Enter Your choice

2

Enter the Mobile number to be validated:

9652341870

Mobile number is invalid

**Question 4:**

Create a custom checked Exception “InvalidEmailException” with custom error message. "InvalidEmailException: Invalid Email for the user".  
Read member details from Input (Console). One line would consist details relating one member in comma separated format. Create member objects for each line and add it to an arraylist.  
a.       A valid email has an @ and ends with “.com / .org”. During the parse, if an email id is invalid, Raise the custom exception and dont add  the entity into the list.  
b.      Sort the arraylist of members based on firstname ( hint: Use Comparable).  
  
**Sample Input and Output:  
[All text in bold corresponds to input and the rest corresponds to output.]**  
  
Enter the number of Members:  
**5**  
Enter the member details:  
**1,joe,root,**[**joe.root@a.com**](mailto:joe.root@a.com)**,1234567890,AH1,12-12-2001,12-12-2010  
2,ben,stokes,ben.stokes!**[**a.com**](http://a.com/)**,2345678901,AH2,12-12-2002,12-12-2011**  
InvalidEmailException: Invalid Email for ben  
**3,virat,kohli,**[**virat.kohli@a.com**](mailto:virat.kohli@a.com)**,3456789012,AH3,12-12-2003,12-12-2012  
4,ravi,varma,**[**ravi.varma@a.in**](mailto:ravi.varma@a.in)**,4357689010,AH4,11-10-2003,12-10-2013**  
InvalidEmailException: Invalid Email for ravi  
**5,mahendra,bagubali,**[**mahi.bali@abc.com**](mailto:mahi.bali@abc.com)**,1234798012,AH5,10-02-2003,12-02-2015**  
  
The member details are:  
1,joe,root,[joe.root@a.com](mailto:joe.root@a.com),1234567890  
5,mahendra,bagubali,[mahi.bali@abc.com](mailto:mahi.bali@abc.com),1234798012  
3,virat,kohli,[virat.kohli@a.com](mailto:virat.kohli@a.com),3456789012

Create a class Hotel with the following private fields

Field Name Data Type

registrationNo String

registrationDate Date

Include appropriate getters and setters for the private fields. Include a default constructor along with a 2-argument constructor. The registration number should be in the format 3 chars-2 nos-2 chars-3 nos like MUM-00-MH-123.

If the input entered by the user is not in this format, print ‘invalid registration number’ else print ‘valid registration number’. The registration number is case sensitive.

In the second option, we assume that the user can enter the data without focusing on case and special characters, convert into above mentioned format and display.

In the third option, check whether the registration date has expired or not. If the date is not more than 10 years old it is valid.

Create a Main class for input and output operations which displays a menu [refer to Sample Input].

• Create a static function validateRegistrationNumber() that takes the String registration number and returns boolean value in the Main class.

• Create a static function convertRegistrationNumber() that takes the String registration number and prints the processed String value in the Main class.

• Create a static function validateRegistrationDate() that takes the String registration date and returns boolean value in the Main class.

Sample Input

Menu

1. Validate registration number

2. Convert hotel registration number

3. Has registration expired

1

KOL-15-WB-101

Sample Output

valid registration number

Sample Input

Menu

1. Validate registration number

2. Convert hotel registration number

3. Has registration expired

2

KOL#15#wb#101

Sample Output

KOL-15-WB-101

Sample Input

Menu

1. Validate registration number

2. Convert hotel registration number

3. Has registration expired

3

16-12-1996

Sample Output

Registration was done 22 years before. Renew.

Sample Input

Menu

1. Validate registration number

2. Convert hotel registration number

3. Has registration expired

3

16-12-2016

Sample Output

Registration was done 1 years before. Continue.

Question:

A company has asked to write a full proof code for IP address validation. The IP address needs to pass certain validation for getting accepted. The validation rules are as given

a. The string must have 4 octets

b. Each octet must be between [0-255]

c. No alphabets or special characters are allowed Print "Valid" if the above validation is passed else "Invalid".

Sample Input

666.10.10.20

Sample Output

Invalid

Sample Input

127.0.0.1

Sample Output

Valid

Question:

A company has an online chat facility. To become a member, users have to create an account with password.

The password needs to pass certain validation for getting accepted. The validation rules are as given

a. The string must contain at least 1 lowercase alphabetical character

b. The string must contain at least 1 uppercase alphabetical character

c. The string must contain at least 1 numeric character

d. The string must contain at least one special character

e. The string must be between 8 and 12 characters in length

f. The string must begin with uppercase alphabetical character Print "Valid" if the above validation is passed else "Invalid".

Sample Input

Macheff@$%

Sample Output

Invalid

Sample Input

Macheff@$%12

Sample Output

Valid