Problem Statement

This question aims to test your knowledge of Regex and Java Library. implement as per the given requirement.

Requirement:

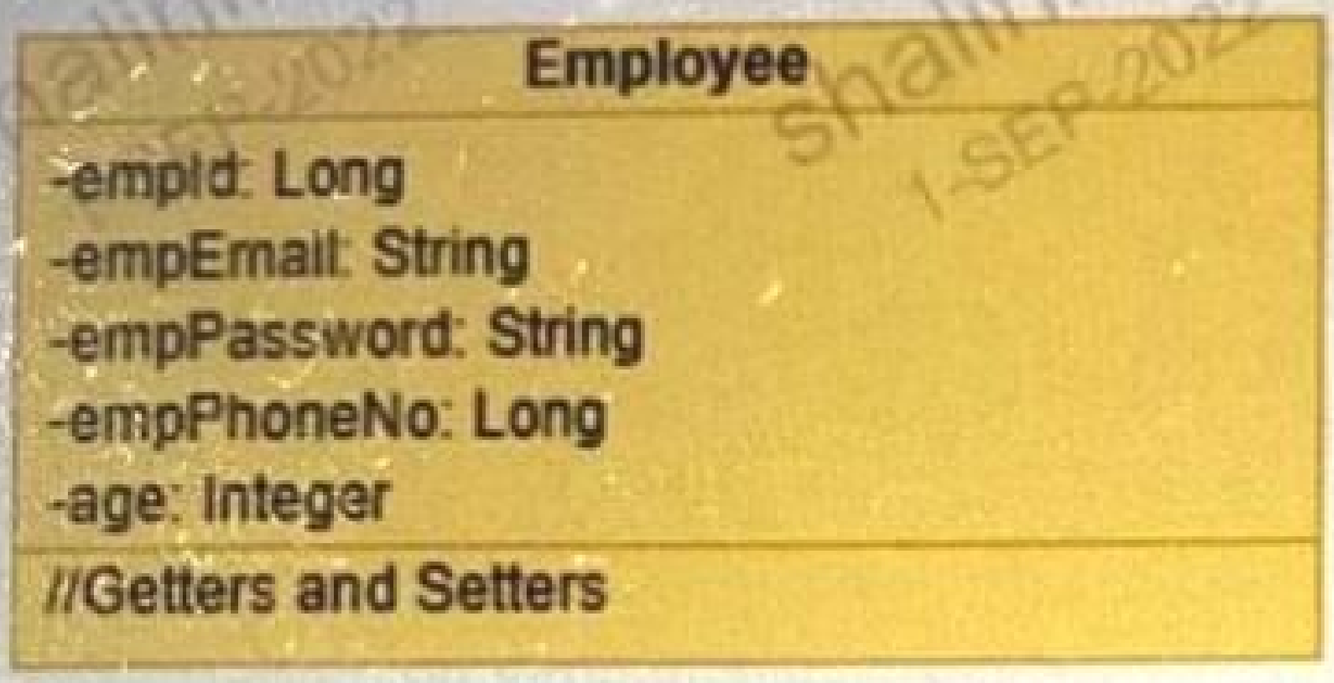
MotoX, a leading MNC in India wants to vaccinate all its employees who are of age 18-44. So, to make the vaccination drive smooth they are creating a new portal in which all the details of its Employees will be taken and validated.

To save time MotoX needs to automate the new portal. As per the first iteration, a series of validations are required for the below components:

1. Employee ID
2. Employee Email
3. Employee Password
4. Employee Phone Number
5. Employee age

Class Diagram:

Class Diagram:



Instructions:

As per the above requirement, the Employee class which is a simple model class, and the Main class accepting all the above parameters with the proper function calls are already provided to you.

Implement the below methods in the Validator class to complete the validation process:

public static Boolean isValidEmployeeId(Long empId):

- This method validates the **empId**.
- **empId** should be a 6-digit or 7-digit positive integer.
- If the above condition is satisfied, return true, otherwise, return false.

Example-

Valid: 123456, 789654, 8765324, 987654, 9876512.

Example-

Valid: 123456, 789654, 8765324, 987654, 9876512.

Invalid: 12334, 78965, 89098765, -7098567, -897654.

public static Boolean isValidEmployeeEmail(String empEmail):

- This method validates **empEmail**.
- The email can be of pattern part1@part2
- part1 can contain both upper case, lower case, and digits from 0-9
- part1 can contain a single dot(.) or underscore(_) or hyphen(-)
- dot(.) or underscore(_) or hyphen(-) shouldn't be the first or last character in the part1
- dot(.) or underscore(_) or hyphen(-) shouldn't appear consecutively anywhere in the part1
- part1 should be of a maximum of 64 characters
- part2 can contain both upper case, lower case, and digits from 0-9
- part2 shouldn't contain any other special character except a dot(.)
- dot(.) shouldn't be the first or last character in the part2
- dot(.) shouldn't appear consecutively anywhere in part2.
- part1 and part2 shouldn't contain any white spaces.
- If the above conditions are satisfied, return true, otherwise, return false.

Example-

Valid: Sachin_Tendulkar@infosys.com, SachinTendulkar@gmail.com, Sachin_Tendulkar123@yahoo.com, Sachi123.tendul@hotmail.com, Sachi.123tendul@hotmail.com

Invalid: Sachin Tendulkar@infosys.com, .SachinTendulkar@infosys.com, Sachin_Tendulkar@.infosys.com, Sachin..Tendulkar@infosys..com

public static Boolean isValidEmployeePassword(String empPassword):

- This method validates **empPassword**.
- It contains at least 8 characters and at most 20 characters.
- It should contain at least one digit.
- It should contain at least one upper case alphabet.
- It should contain at least one lower case alphabet.
- It should contain at least one special character which includes '@', '#'.
- It should not contain any white space.
- If the above conditions are satisfied, return true, otherwise, return false.

Example-

Valid: Sachin@1234, Jack#8989, Jack@Jill1234

Invalid: JackAndJill, Jack1234, 123456, jack#1234

Example-

Valid: Sachin@1234, Jack#8989, Jack@Jill1234

Invalid: JackAndJill, Jack1234, 123456, jack#1234

public static Boolean isValidEmployeePhoneNo(Long empPhoneNo):

- This method validates **empPhoneNo**.
- PhoneNo must be a 10-digit number and all the numbers should not be the same.
- If the above condition is satisfied, return true, otherwise, return false.

Example-

Valid: 7416306445,8179845754

Invalid: 741630644,9999999999

public static Boolean isValidEmployeeAge(Integer age):

- This method validates the **age** of an Employee.
- The 'age' must be greater than equals to 18 years and less than 45 years.
- If the above conditions are satisfied, return true, otherwise, return false.

Example-

Example-

Valid: 18, 27, 35, 44.

Invalid: 17, 1, -1, 45, 46.

Fully Implemented Employee.java, Main.java, and partially implemented Validator.java is already given to you.

Note:

- Please don't alter/change the codes which have already provided.
- Whatever class/interface you are adding OR already provided should not be 'public'.

Languages: Java

Sample Input	Sample Output	Explanation
234567 john_wick@infosys.com John@123 7022713455 18	true true true true true	All the details are valid.
23456	false	empId is invalid as it's a

Modifiers.java | Employee.java | Course.java | Practice.java

```
4
5 public class Practice {
6     public static void main(String[] args) {
7
8         Scanner sc = new Scanner(System.in);
9         Employeee emp = new Employeee();
10
11        Long empId = sc.nextLong();
12        emp.setEmpId(empId);
13
14        String email = sc.next();
15        emp.setEmpEmail(email);
16
17        String password = sc.next();
18        emp.setEmpPassword(password);
19
20        Long phoneNo = sc.nextLong();
21        emp.setEmpPhoneNo(phoneNo);
22
23        Integer age = sc.nextInt();
24        emp.setAge(age);
25
26        Validator.validate(emp);
27        sc.close();
28    }
29 }
30
31 class Employeee {
32     private long empId;
33     private String empEmail;
34     private String empPassword;
35     private long empPhoneNo;
```

Modifiers.java | Employee.java | Course.java | Practice.java

```
32      private long empId;
33      private String empEmail;
34      private String empPassword;
35      private long empPhoneNo;
36      private Integer age;
37
38      public long getEmpId() {
39          return empId;
40      }
41
42      public void setEmpId(long empId) {
43          this.empId = empId;
44      }
45
46      public String getEmpEmail() {
47          return empEmail;
48      }
49
50      public void setEmpEmail(String empEmail) {
51          this.empEmail = empEmail;
52      }
53
54      public String getEmpPassword() {
55          return empPassword;
56      }
57
58      public void setEmpPassword(String empPassword) {
59          this.empPassword = empPassword;
60      }
61
62      public long getEmpPhoneNo() {
63          return empPhoneNo;
```

Modifiers.java | Employee.java | Course.java | Practice.java ×

```
67          this.empPhoneNo = empPhoneNo;
68      }
69
70      public Integer getAge() {
71          return age;
72      }
73
74      public void setAge(Integer age) {
75          this.age = age;
76      }
77
78  }
79
80  class Validator {
81      public static void validate(Employeee employee) {
82          System.out.println(Validator.isValidEmployyeId(employee.getEmpId()));
83          System.out.println(Validator.isValidEmployeeEmail(employee.getEmpEmail()));
84          System.out.println(Validator.isValidEmployeePassword(employee.getEmpPassword()));
85          System.out.println(Validator.isValidEmployeePhoneNo(employee.getEmpPhoneNo()));
86          System.out.println(Validator.isValidEmployeeAge(employee.getAge()));
87      }
88
89      public static Boolean isValidEmployyeId(Long empId) {
90          Boolean flag = false;
91          String regex = "[0-9]{6,7}";
92          if (empId.toString().matches(regex)) {
93              flag = true;
94          }
95          return flag;
96
97      }
98
```

Modifiers.java | Employee.java | Course.java | Practice.java ×

```java
    public static Boolean isValidEmployeeEmail(String empEmail) {
        Boolean flag = false;
        String regex = "([0-9A-Za-z]+[_.-]{0,1}[0-9A-Za-z]+){1,64}@[A-Za-z0-9.]+[_.-]{0,1}[0-9A-Za-z]+";
        if (empEmail.matches(regex)) {
            flag = true;
        }
        return flag;

    }

    public static Boolean isValidEmployeePassword(String empPassword) {
        Boolean flag = false;
        String regex = "^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[@#])[a-zA-Z0-9@#]{8,20}$*";
        if (empPassword.matches(regex)) {
            flag = true;
        }
        return flag;

    }

    public static Boolean isValidEmployeePhoneNo(Long empPhone) {
        Boolean flag = false;
        String phone = empPhone.toString();
        String regex = "[0-9]{10}";
        if (phone.matches(regex) && !((empPhone % 1111111111) == 0)) {
            flag = true;
        }
        return flag;
    }

    public static Boolean isValidEmployeeAge(Integer age) {
```

Modifiers.java | Employee.java | Course.java | *Practice.java ×

```
117     }
118
119     public static Boolean isValidEmployeePhoneNo(Long empPhone) {
120         Boolean flag = false;
121         String phone = empPhone.toString();
122         String regex = "[0-9]{10}";
123         if (phone.matches(regex) && !((empPhone % 1111111111) == 0)) {
124             flag = true;
125         }
126         return flag;
127     }
128     public static Boolean isValidEmployeeAge(Integer age) {
129         Boolean flag = false;
130         if (age >= 18 && age < 45) {
131             flag = true;
132         }
133         return flag;
134
135     }
136 }
137
```

Error Log | Declaration | Console × | JUnit | Javadoc

<terminated> Practice [Java Application] D:\TypeScript SDK 4.5.5\SoftwareCenterApplications\Eclipse IDE for Java Developers

```
23456
john_wick@infosys.com
John@123
7022713455
18
false
true
true
true
```

Modifiers.java | Employee.java | Course.java | Practice.java

```
127        }
128    public static Boolean isValidEmployeeAge(Integer age) {
129        Boolean flag = false;
130        if (age >= 18 && age < 45) {
131            flag = true;
132        }
133        return flag;
134
135    }
136 }
137
138
```

Error Log | Declaration | Console | JUnit | Javadoc

<terminated> Practice [Java Application] D:\TypeScript SDK 4.5.5\SoftwareCenterApplications\Eclip

```
234567
john_wick@infosys.com
John@123
7022713455
18
true
true
true
true
true
```