# Lab Task 3: Data Validation

### 3.1. Identify a scenario in your project where data validation is necessary.

In a software project, user input for age and email address needs to be validated to ensure accuracy and prevent errors.

### 3.2. Define a set of validation rules for the identified scenario.

#### 1. Age Validation:

* + Age must be a non-negative integer.
  + Age must be within the range of 0 to 120 years.

#### 2. Email Validation:

* + Email address must follow the standard email format:
    - It should contain a local part followed by '@' symbol and a domain part.
    - The local part can contain alphanumeric characters, dots, underscores, and hyphens.
    - The domain part should contain alphanumeric characters and hyphens, separated by dots.
    - The domain must have at least two characters.

### 3.3 Implement two versions of data validation: one without applying the rules and another with the rules applied.

#### 3.3.1 Code without applying validation rules:

import java.util.Scanner;

public class DataValidation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your age: ");

int age = scanner.nextInt();

System.out.print("Enter your email address: ");

String email = scanner.next();

System.out.println("Age: " + age);

System.out.println("Email: " + email);

scanner.close();

}

}

#### 3.3.2. Code with validation rules applied:

package errorchecklist;

import java.util.Scanner;

public class User {

private int age;

private String email;

public User() {

// Default constructor

}

public void setAge(int age) {

this.age = age;

}

public int getAge() {

return age;

}

public void setEmail(String email) {

this.email = email;

}

public String getEmail() {

return email;

}

public void validateAge(Scanner scanner) {

while (true) {

System.out.print("Enter your age (0-120): ");

int inputAge = scanner.nextInt();

if (inputAge >= 0 && inputAge <= 120) {

this.age = inputAge;

break;

} else {

System.out.println("Invalid age. Please enter a valid age.");

}

}

}

public void validateEmail(Scanner scanner) {

while (true) {

System.out.print("Enter your email address: ");

String inputEmail = scanner.next();

if (isValidEmail(inputEmail)) {

this.email = inputEmail;

break;

} else {

System.out.println("Invalid email address. Please enter a valid email.");

}

}

}

private boolean isValidEmail(String email) {

return email.matches("^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$");

}

}

### 3.4. Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Description** | **Input Data** | **Expected Outcome** | **Actual Outcome** | **Status** |
| TC-01 | Valid age input within the range | Age: 25, Email: test@example.com | Age: 25, Email: test@example.com |  |  |
| TC-02 | Age input less than 0 | Age: -5, Email: test@example.com | "Invalid age. Please enter a valid age." message |  |  |
| TC-03 | Age input greater than 120 | Age: 150, Email: test@example.com | "Invalid age. Please enter a valid age." message |  |  |
| TC-04 | Valid email input | Age: 30, Email: test@example.com | Age: 30, Email: test@example.com |  |  |
| TC-05 | Email input without "@" symbol | Age: 30, Email: testexample.com | "Invalid email address. Please enter a valid email." |  |  |
| TC-06 | Email input without domain | Age: 30, Email: test@ | "Invalid email address. Please enter a valid email." |  |  |
| TC-07 | Email input with invalid characters | Age: 30, Email: test!@example.com | "Invalid email address. Please enter a valid email." |  |  |
| TC-08 | Email input with invalid domain format | Age: 30, Email: test@example | "Invalid email address. Please enter a valid email." |  |  |
| TC-09 | Email input with single character domain | Age: 30, Email: test@e.com | "Invalid email address. Please enter a valid email." |  |  |
| TC-10 | Email input with valid but long domain | Age: 30, Email: test@verylongdomain.com | Age: 30, Email: test@verylongdomain.com |  |  |