Java Problems

Vishwas K Singh, SME-FSD

Contents

1 IP Address Validator & Classifier

1.1 Problem Statement:

1.1.1 Given

A list of IPv4 addresses in string format.

1.1.2 Task

1. Validate: Ensure each IPv4 address in the list adheres to the standard format: four decimal numbers, each ranging from 0 to 255, separated by periods.
2. Extract: Extract the unique IPv4 addresses from the list, eliminating duplicates.
3. Classify: Categorize each unique IPv4 address into its respective class (A, B, C, D, or E) based on the first octet:

IPv4 Class	Octet Range
Class A	1-127
Class B	128-191
Class C	192-223
Class D	224-239
Class E	240-255

4. Group and Print: Group the unique IPv4 addresses by their class and print them in a clear, organized format.

1.1.3 Example Input:

```
192.168.1.1
10.0.0.1
172.16.254.1
255.255.255.255
-0.12.23.1
1b.35.1.10
192.168.1.1
```

1.1.4 Expected Output

> Class A:
> 10.0.0.1
> Class B:
> 172.16.254.1

> Class C:
> 192.168.1.1
> Class E:
> 255.255.255.255

2 Name Swapper

2.1 Scenario

A social media platform is developing a new feature that allows users to rearrange their names. This feature is essential for users who have changed their names, prefer a different name order, or simply want to experiment with their profile.

2.2 Problem Statement

- Given a user's full name in the format `first_name last_name`
- write a Java program to:

> **Input**: Prompt the user to enter their full name.
> **Parse**: Split the full name into two parts: first name and last name.
> **Swap**: Swap the first name and last name.
> **Output**: Print the swapped full name in the format `last_name first_name`

2.3 Expected Output

> Input: John Doe Output: Doe John

3 Text Analysis Tool

3.1 Scenario

A content analyst at a digital marketing agency needs to analyze large volumes of text data, such as blog posts, social media comments, or customer reviews. They need a tool to quickly extract key information from the text, such as character frequencies, word count, and line count.

3.2 Problem Statement

Write a Java program to analyze a given text string and provide the following information:

- **Character Frequency**:
  - Count the occurrence of each character in the text.
  - Display the character and its corresponding frequency.
- **Word Count**:
  - Determine the total number of words in the text.
- **Line Count**:
  - Calculate the number of lines in the text.
- **Total Character Count**:
  - Count the total number of characters in the text.

3.3 Example

3.3.1 Input Text

```
This is a sample text.
It has multiple lines.
We can analyze it.
```

3.3.2 Output

```
Character Frequencies:
Character | Frequency
|---------------------|
  t        |    7
  i        |    6
  s        |    6
  a        |    5

Word Count: 13
Line Count: 3
Total Character Count: 64
```

4 Email Parser

4.1 Scenario:

A cybersecurity analyst is tasked with analyzing a large dataset of emails to identify potential phishing attempts or malicious activities. To automate the process, they need a tool to extract the username and domain name from each email address.

4.2 Problem Statement:

Write a Java program to:

- **Validate Email**:
  Check if the given email address conforms to the following rules:

1. At least one character in the username.
2. Only `.` and `_` allowed in the username.
3. Must contain an `@` symbol.
4. Must have a domain name with at least one `.`

- **Extract Components**:
  If the email is valid, extract the username and domain name.
- **Display Output**:
  Print the extracted username and domain name.

4.3 Example:

4.3.1 Input:

```
john.doe@example.com
```

4.3.2 Output:

```
Username: john.doe
Domain Name: example.com
```