Link Shortener Development

Submitted by: V Chandra Sekhar

Date: 15/02/2025

1. Introduction

A Link Shortener converts long URLs into unique short links while allowing retrieval of the original URL. This project involves developing a Java-based Link Shortener system with persistence for saving mappings.

2. Objectives

- Develop a Java-based URL Shortener system.
- Implement a mechanism to shorten and expand URLs.
- Ensure uniqueness of short URLs and prevent collisions.
- Handle errors like duplicate long URLs and invalid short URLs.
- Provide data persistence for maintaining URL mappings.
- Develop a simple CLI interface for user interaction.

3. Implementation Details

The project consists of two main classes:

1. URLShortener - Manages URL shortening, expansion, and persistence.
2. LinkShortenerApp - Provides a user interface for interaction.

4. Java Source Code

URLShortener.java (With Persistence):

```
import java.io.*;
import java.util.*;

public class URLShortener {
    private static final String BASE_URL = "https://short.ly/";
    private static final String FILE_NAME = "url_mappings.txt";
    private Map<String, String> urlMap = new HashMap<>();
    private Map<String, String> reverseMap = new HashMap<>();
    private static final String CHARACTERS = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    private Random random = new Random();
```

```
    public URLShortener() {
        loadMappings();
    }

    public String shortenURL(String longURL) {
        if (reverseMap.containsKey(longURL)) {
            return BASE_URL + reverseMap.get(longURL);
        }
        String shortCode;
        do {
            shortCode = generateShortCode();
        } while (urlMap.containsKey(shortCode));

        urlMap.put(shortCode, longURL);
        reverseMap.put(longURL, shortCode);
        saveMappings();
        return BASE_URL + shortCode;
    }

    public String expandURL(String shortURL) {
        String code = shortURL.replace(BASE_URL, "");
        return urlMap.getOrDefault(code, "Invalid Short URL");
    }

    private String generateShortCode() {
        StringBuilder shortCode = new StringBuilder(6);
        for (int i = 0; i < 6; i++) {
            shortCode.append(CHARACTERS.charAt(random.nextInt(CHARACTERS.length())));
        }
        return shortCode.toString();
    }

    private void saveMappings() {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            out.writeObject(urlMap);
        } catch (IOException e) {
            System.out.println("Error saving URL mappings.");
        }
    }

    private void loadMappings() {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            urlMap = (HashMap<String, String>) in.readObject();
            urlMap.forEach((key, value) -> reverseMap.put(value, key));
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("No previous data found. Starting fresh.");
        }
    }
}
```

LinkShortenerApp.java:

```
import java.util.Scanner;
```

```
public class LinkShortenerApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        URLShortener shortener = new URLShortener();

        while (true) {
            System.out.println("\nLink Shortener Menu:");
            System.out.println("1. Shorten URL");
            System.out.println("2. Expand URL");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter the long URL: ");
                    String longURL = scanner.nextLine();
                    System.out.println("Shortened URL: " + shortener.shortenURL(longURL));
                    break;
                case 2:
                    System.out.print("Enter the short URL: ");
                    String shortURL = scanner.nextLine();
                    System.out.println("Original URL: " + shortener.expandURL(shortURL));
                    break;
                case 3:
                    System.out.println("Exiting... Goodbye!");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
}
```

5. Code Explanation

URLShortener.java:

- Maintains two HashMaps (urlMap and reverseMap) for bidirectional mapping.

- Uses a random 6-character code generator for short URLs.

- Implements file operations for persistence using ObjectOutputStream and ObjectInputStream.

- Provides methods shortenURL() and expandURL().

LinkShortenerApp.java:

- Displays a simple command-line interface with options.

- Uses a while-loop to keep the program running until the user exits.

- Collects input and invokes methods from URLShortener.

Error Handling:

- Prevents collisions by checking existing short codes.
- Handles invalid short URL expansion.
- Displays proper messages for invalid menu choices.

6. Output Verification Examples

Example 1: Shortening a URL

User Input: "https://www.example.com/page1"

Expected Output: "Shortened URL: https://short.ly/abc123"

Example 2: Expanding a URL

User Input: "https://short.ly/abc123"

Expected Output: "Original URL: https://www.example.com/page1"

Example 3: Invalid Short URL

User Input: "https://short.ly/xyz999"

Expected Output: "Invalid Short URL"

Example 4: Persistence Verification

Restart program -> Previously saved URLs remain accessible.

7. Conclusion

This improved implementation of the Link Shortener system ensures data persistence and enhanced error handling. The project successfully demonstrates Java's capabilities in handling data structures, file operations, and user interaction.