Implementation of a Link Shortener application in Java. This example will include a class to manage the shortening and expanding of URLs, a basic hash function for generating short URLs, and error handling. We'll also provide a simple command-line interface (CLI) for user interaction.

Step 1: Create the URL Shortener Class

import java.util.HashMap;

import java.util.Map;

import java.util.Random;

public class URLShortener {

private static final String CHARACTERS = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

private static final int BASE = CHARACTERS.length();

private static final int SHORT\_URL\_LENGTH = 6;

private Map<String, String> longToShortMap;

private Map<String, String> shortToLongMap;

private Random random;

public URLShortener() {

longToShortMap = new HashMap<>();

shortToLongMap = new HashMap<>();

random = new Random();

}

// Generates a short URL for a given long URL

public String shortenURL(String longURL) {

if (longToShortMap.containsKey(longURL)) {

return longToShortMap.get(longURL);

}

String shortURL;

do {

shortURL = generateShortURL();

} while (shortToLongMap.containsKey(shortURL));

longToShortMap.put(longURL, shortURL);

shortToLongMap.put(shortURL, longURL);

return shortURL;

}

// Expands a given short URL to its original long URL

public String expandURL(String shortURL) {

return shortToLongMap.getOrDefault(shortURL, "Invalid short URL");

}

// Generates a random short URL

private String generateShortURL() {

StringBuilder shortURL = new StringBuilder(SHORT\_URL\_LENGTH);

for (int i = 0; i < SHORT\_URL\_LENGTH; i++) {

shortURL.append(CHARACTERS.charAt(random.nextInt(BASE)));

}

return shortURL.toString();

}

}

Step 2: Create the Command-Line Interface (CLI)

import java.util.Scanner;

public class URLShortenerCLI {

public static void main(String[] args) {

URLShortener urlShortener = new URLShortener();

Scanner scanner = new Scanner(System.in);

while (true) {

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(); // Consume newline

switch (choice) {

case 1:

System.out.print("Enter long URL: ");

String longURL = scanner.nextLine();

String shortURL = urlShortener.shortenURL(longURL);

System.out.println("Short URL: " + shortURL);

break;

case 2:

System.out.print("Enter short URL: ");

shortURL = scanner.nextLine();

longURL = urlShortener.expandURL(shortURL);

System.out.println("Long URL: " + longURL);

break;

case 3:

System.out.println("Exiting...");

scanner.close();

return;

default:

System.out.println("Invalid option. Please try again.");

}

}

}

}

**How It Works**

1. **URLShortener Class**:
   * **shortenURL**: Takes a long URL and generates a unique short URL.
   * **expandURL**: Takes a short URL and returns the corresponding long URL.
   * **generateShortURL**: Generates a random short URL using a combination of characters.
2. **URLShortenerCLI Class**:
   * Provides a simple command-line interface for users to shorten and expand URLs.
   * Uses a Scanner to take user input and perform actions based on the user's choice.

**Running the Application**

Compile the URLShortener and URLShortenerCLI classes and run the URLShortenerCLI class. You will see a menu allowing you to choose between shortening a URL, expanding a URL, and exiting the application.

\*bash\*

javac URLShortener.java URLShortenerCLI.java

java URLShortenerCLI

This simple implementation gives you a basic understanding of data structures (using HashMap), algorithms (generating unique short URLs), and basic Java programming. You can expand on this by adding features such as persistence (saving the mappings to a file or database) and a web-based interface.