**WEEK 2 TASK**

**AIM :-** Develop an algorithm to convert long URLs into shorter links. Create a user-friendly interface for users to input long URLs and receive shortened links as output.

**PROGRAM :-**

Import javax.swing.\*;

Import java.awt.event.ActionEvent;

Import java.awt.event.ActionListener;

Public class LinkShortenerGUI {

Private static class LinkShortener {

Public String shortenUrl(String longUrl) {

// Simulating shortening functionality (replace this with actual logic)

Return <http://short.url/> + longUrl.hashCode(); // Generating a fake shortened URL

}

}

Private LinkShortener linkShortener;

Public LinkShortenerGUI() {

linkShortener = new LinkShortener();

}

Private void createAndShowGUI() {

JFrame frame = new JFrame(“Link Shortener”);

Frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Frame.setSize(400, 150);

JPanel panel = new JPanel();

Frame.add(panel);

placeComponents(panel);

frame.setVisible(true);

}

Private void placeComponents(JPanel panel) {

Panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));

JLabel longUrlLabel = new JLabel(“Long URL:”);

JTextField longUrlTextField = new JTextField(20);

JButton shortenButton = new JButton(“Shorten”);

JLabel shortUrlLabel = new JLabel(“Shortened URL:”);

Panel.add(longUrlLabel);

Panel.add(longUrlTextField);

Panel.add(shortenButton);

Panel.add(shortUrlLabel);

shortenButton.addActionListener(new ActionListener() {

@Override

Public void actionPerformed(ActionEvent e) {

String longUrl = longUrlTextField.getText();

If (!longUrl.isEmpty()) {

Try {

String shortUrl = linkShortener.shortenUrl(longUrl);

shortUrlLabel.setText(“Shortened URL: “ + shortUrl);

} catch (Exception ex) {

shortUrlLabel.setText(“Error: “ + ex.getMessage());

}

} else {

shortUrlLabel.setText(“Please enter a long URL.”);

}

}

});

}

Public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

Public void run() {

New LinkShortenerGUI().createAndShowGUI();

}

});

}

}

**Code Explanation :-**

* Class Structure: Defines a class `LinkShortenerGUI` encapsulating functionalities for a GUI-based link shortening tool.
* Inner Class `LinkShortener`: Simulates a link shortening functionality by hashing the provided URL to generate a fake shortened URL.
* Instance Variables: Contains an instance of `LinkShortener` for performing the URL shortening within the GUI.
* `LinkShortenerGUI` Constructor: Initializes the `LinkShortener` instance for use within the GUI.
* `createAndShowGUI()` Method: Sets up the GUI components by creating a `JFrame` window, defining its attributes (title, size, and close operation), and adding a `JPanel`.
* `placeComponents()` Method: Organizes the components (labels, text field, button) within the panel using a `BoxLayout` in the Y\_AXIS direction.
* Button Action Listener: Defines an action listener for the “Shorten” button, capturing the input from the text field, attempting to shorten the URL using the `LinkShortener`, and updating the displayed shortened URL label accordingly.
* Main Method: Utilizes `SwingUtilities.invokeLater()` to ensure the GUI creation and display are performed within the Event Dispatch Thread (EDT) for Swing.
* Event-Driven Operation: The GUI is event-driven, where the action of clicking the “Shorten” button triggers the URL shortening process and updates the displayed result.
* Simulated Functionality: While this code simulates URL shortening by using a hash code to generate a fake shortened URL, in a real implementation, the `LinkShortener` class would typically interface with an actual URL shortening service or algorithm to produce valid shortened URLs.