A paper with text and images

Description automatically generated

A white paper with black text

Description automatically generated

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

public class LossyCompression {

    public static void main(String[] args) {

        // Load image

        BufferedImage image = load("input.jpg");

        // Perform lossy compression

        byte[] compressedData = lossyCompression(image);

        // Save compressed data to file

        saveToFile(compressedData, "output.compressed");

    }

    public static BufferedImage load(String filePath) {

        try {

            return ImageIO.read(new File(filePath));

        } catch (IOException e) {

            e.printStackTrace();

            return null;

        }

    }

    public static void saveToFile(byte[] data, String filePath) {

        try {

            // Write data to file

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    public static byte[] lossyCompression(BufferedImage image) {

        // Convert image to YCbCr color space

        byte[][][] imageYCbCr = convertToYCbCr(image);

        // Apply Discrete Cosine Transform (DCT) to each block

        byte[][][] blocks = divideImageIntoBlocks(imageYCbCr);

        for (int i = 0; i < blocks.length; i++) {

            for (int j = 0; j < blocks[i].length; j++) {

                blocks[i][j] = applyDCT(blocks[i][j]);

            }

        }

        // Quantize DCT coefficients

        byte[][][] quantizedCoefficients = quantize(blocks);

        // Encode quantized coefficients

        byte[] encodedData = encode(quantizedCoefficients);

        return encodedData;

    }

    public static byte[][][] convertToYCbCr(BufferedImage image) {

        // Implementation not provided, convert RGB to YCbCr

        return null;

    }

    public static byte[][][] divideImageIntoBlocks(byte[][][] imageYCbCr) {

        // Implementation not provided, divide image into blocks

        return null;

    }

    public static byte[] applyDCT(byte[][] block) {

        // Implementation not provided, apply DCT to block

        return null;

    }

    public static byte[][][] quantize(byte[][][] dctCoefficients) {

        // Implementation not provided, quantize DCT coefficients

        return null;

    }

    public static byte[] encode(byte[][][] quantizedCoefficients) {

        // Implementation not provided, encode quantized coefficients

        return null;

    }

}