**Final Project Report: Image Resizing and Compression Application in Java**

**Prepared by:**ByteXL

**Educator:** Sivanraj A

### Executive Summary

This report details the development of an Image Compression project built in Java. The project aims to reduce image file sizes while maintaining visual quality through resizing and compression techniques. Key features include image loading, resizing, and adjusting JPEG quality, which highlight practical applications of image processing in Java. The project also emphasizes ease of use with runtime file selection, making it a user-friendly tool for image compression.

### 1. Tools and Technologies Utilized

* **Programming Language**: Java
* **Image Processing Libraries**: BufferedImage, ImageIO, Graphics2D
* **Testing and Debugging**: VSCode
* **Documentation and Design**: Nimbus Platform

### 2. Project Overview

The project provides an image compression tool that loads, resizes, and compresses image files to reduce storage requirements. By utilizing Java’s standard libraries, the project supports JPEG compression and offers a simple file selection dialog for ease of use. Best practices in modular design, error handling, and user feedback are implemented to ensure efficiency and reliability.

### 3. System Requirements

**Software Requirements**

* JDK 8 or higher
* Nimbus Platform for documentation
* Compatible IDEs: VSCode

**Hardware Requirements**

* Minimum RAM: 2GB
* Processor: Dual-core or better
* Operating System: 32bit or 64-bit (Linux, Windows, MacOS)

### 4. Functional Requirements

The project fulfills the following functionalities for image compression:

* **Image Loading**: Load image files for processing.
* **Image Resizing**: Adjust image dimensions to reduce pixel data.
* **JPEG Compression**: Modify image quality to control file size.
* **File Output**: Save the compressed image as a new file.

### 5. User Interface Requirements

The program includes a command-line interface (CLI) that provides:

* **File Selection Dialog**: Users can select an image file at runtime.
* **Feedback Messages**: Confirmation of successful or unsuccessful operations.
* **Input Validation**: Ensures valid image file formats are selected.

### 6. Inputs and Outputs

**Inputs**

* Image file selected by the user
* Desired compression quality and output dimensions

**Outputs**

* Compressed image file
* Feedback messages on success or failure

### 7. System Subcomponents

The project includes the following main components:

* **Image Loading Component**: Handles loading of user-selected image files.
* **Image Resizing Component**: Resizes images to specified dimensions.
* **JPEG Compression Component**: Compresses images to desired quality.
* **File Output Component**: Saves the processed image to a new file.

### 8. Potential for Other Applications

The image compression tool can be adapted for various use cases, including:

* **Photo Management Systems**: Reducing storage requirements for large image libraries.
* **Web Development**: Compressing images to optimize website loading times.
* **Mobile Applications**: Decreasing image file sizes to reduce data usage.

### 9. Test Case Design

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Function** | **Expected Outcome** |
| TC001 | Load Image | Image loaded successfully |
| TC002 | Resize Image | Image resized to specified dimensions |
| TC003 | Compress Image Quality | Image compressed to expected quality level |
| TC004 | Save Compressed Image | Compressed image saved successfully |
| TC005 | Invalid File Selection | Proper error message displayed for unsupported files |
| TC006 | File Not Found | Proper error message displayed for missing files |

### 10. Future Enhancements

Suggested improvements for future versions include:

* **Additional File Format Support**: Add support for formats beyond JPEG, such as PNG and BMP.
* **Batch Processing**: Allow multiple files to be compressed at once.
* **Graphical User Interface (GUI)**: Introduce a GUI for improved usability.

### 11. References

* Java Image Processing Documentation
* BufferedImage and ImageIO Usage Guides
* Java Best Practices for Image Manipulation
* Nimbus Platform Usage for Documentation

### 12. Project Reflection

**Technical Challenges Encountered**

* **Compression Quality Control**: Balancing file size reduction and image quality was essential but challenging.
* **Error Handling**: Managing exceptions when loading unsupported or corrupted files required thorough testing.
* **Resizing Accuracy**: Ensuring aspect ratio preservation during resizing was crucial for quality results.

**Software Engineering Insights**

The project reinforced best practices in modular programming and error handling within a Java application. Image processing concepts were explored thoroughly, adding valuable insights into real-world application development.

**Personal Development**

This project enhanced my understanding of Java's image processing capabilities and gave practical experience in balancing efficiency with quality.

**Additional Knowledge Gained**

Future improvements could involve learning advanced image processing libraries for faster and more diverse compression techniques.

**Personal Development**

This project improved my knowledge in cryptographic protocols and DSA, emphasizing secure data handling and C programming practices.

**Additional Knowledge Gained**

Future work could benefit from integrating advanced C features and learning assembly-level programming for optimized cryptographic operations.