aASSIGNMENT-1

GAURAV MITTAL 2012CSB1013

OBJECTIVE

The purpose of this assignment is to build a system where any machine which is connected to a network can offer certain services to local or remote machine.

OVERVIEW

The objective is achieved by creating a service on a TCP port using programming API which can be used by the clients to communicate with the server and send requests asking for a specific type of service.

In this assignment, I have created an Image Processing API using Java Programming Language which can be used to process an image by applying operations over it, such as, Scaling, Changing Brightness, Gaussian Blur and more.

DESCRIPTION

- The Image Processing API comprises of two packages forming the client side and server side. The package named imgProcAPI is the client side of the API which provides a client interface with necessary functions, for communicating with the Image Processing Server and asking for various types of operations to be performed on the images.
- The package named imgProcServerAPI forms the server side of the API which creates a ServerSocket object at a given port to act as the Image Processing Server for catering to the various requests put forward by the clients.
- The package imgProcAPI consists of a class ImgProcClient which creates a Socket object to listen on behalf of the client, which can then, establish a connection with the server and upload the image to be worked upon.
- The ImgProcClient consists of various functions to deal with each request separately and to send them to the server for some processing. Finally, processed data and feedback messages are received to be displayed and to be further looked upon.
- The package imgProcServerAPI consists of a class ImgProcServer which defines the ServerSocket to accept Sockets listening to the clients from the server side. Then, it accepts connection from a client and assigns a separate thread of execution for every separate client. With multithreading, multiple clients

can be connected to a server at the same time which is considered to be a better design than accepting connections and queuing them on the server side.

- Each thread is of type ImgProcClientThread which extends thread and defines a socket to behave as a connection for the client on the server side.
- ImgProcClientThread accepts image from the client and creates new ImgProcProtocol object to perform operations on the image.
- ImgProcProtocol performs various operations on the image by calling static functions from the class ImgProcOperations. This class contains functions for every Image Processing Operation which can be performed and helps to generalize these operations by encapsulating them in a separate class.

FEATURES

- The Image Processing API is designed such that multiple images can processed during the same socket connection in a sequential manner. Also, multiple Image Processing operations can be performed back to back on an image loaded on the server without the need to upload and retrieve the image again and again.
- The API is capable of handling images of the following formats:
  - jpg
  - png
  - bmp
- The image uploaded on the server is changed to RGB type image so as to perform certain operations which otherwise cannot be performed.
- The various requests that can be sent to the server are:
  - **PERFORM**: This request asks for which Image Processing operation to be performed on the image. The operation is performed on the image and the processed image by the name, procImage, is stored on the server for retrieval or further processing by the client.

    The various Image Processing Operations, at present, included in the API are:
    - *Gaussian Blur:* Blurs the given image using a 3x3 Gaussian distribution kernel.
    - *Invert Color:* Inverts the color of each pixel of the image by applying a LookUp Transformation.
    - *Sharpen:* Sharpens the image by convolving the image with a 3x3 Sharpening kernel.
    - *Scale Up:* Increases the size of the image to 200% maintaining the aspect ratio.

- *Scale Down:* Decreases the size of the image to 50% maintaining the aspect ratio.
- *Brighten:* Increases the brightness of the image by 50% on all three channels (RGB).
- *Darken:* Decreases the brightness of the image by 50% on all three channels (RGB).

- **REPLACE:** This request helps in replacing the existing image on the server with a new image, thus enabling to process more than one image without starting a new client connection.
- **RECEIVE:** This request helps in receiving back the processed image which can then be saved as an image file of the desired format (jpg,png,bmp).
- **CLOSE:** This request finally closes the connection to the server. No further requests can then be sent to the server.

- Both server side and client side of the API have the necessary functions to cater to these requests and, send and receive feedback messages during a request.
- The API has been developed in such a way that it can be further extended to bring in more types of requests as well as to allow more number of Image Processing Operations to be defined.

TESTING ENVIRONMENT

To test the ImgProcAPI, package called imgProcInterface present in the clientInterface.jar behaves as the client side interface and package called server present in the server.jar behaves as the server side of the API.

- imgProcInterface is designed such that it takes in the server name and image file to be processed as arguments and then, processes the requests. These requests may be present in a file or entered from terminal as per the specified format, which are then parsed and processed using ANTLR v4.
- Server package contains the Main.class which creates the ImgProcServer at a particular port (taken as an argument at the time of execution) and waits for the clients to connect asynchronously.

CONTENTS

The A-1-2012CSB1013 folder contains:

1. ImageSocketAPI folder: This folder further contains two folders, bin folder containing the compiled classes and source folder containing the .java files for various packages.
2. clientInterface.jar: It may be executed with proper arguments to behave as the client side interface of the API.
3. Server.jar: It may be executed with proper arguments to start Image Processing server.
4. request_set.txt: .txt file containing a list of sample requests.

5. ASSIGNMENT_MANUAL.pdf: PDF file describing the working of the program and how to give instructions to the program.
6. Glass.jpg: Image file for testing the program.
7. API jars: This folder contains imgProcAPI.jar and imgProcServerAPI.jar which contain the libraries to behave as Image Processing Client and Image Processing Server respectively. They could be imported in a program to serve as client and server respectively.

HOW TO RUN

1. Open the terminal/command line.
2. Reach the folder containing clientInterface.jar and server.jar.
3. Start the server by executing the following command:
   $ java -jar server.jar <port_no>
   Sample-Run
   $ java –jar server.jar 6789
4. Open another terminal in the same/another machine to start the client.
5. Now, start a client by executing the following command:
   a. If requests are in a file, then:
      $ java -jar clientInterface.jar <server_name/IP> <port_number> <image_file> < <input_textfile_name>
      Sample-Run
      $ java –jar clientInterface.jar localhost 6789 glass.jpg < request_set.txt
   b. Else, simply execute:
      $ java -jar clientInterface.jar <server_name/IP> <port_number> <image_file>
      Sample-Run
      $ java –jar clientInterface.jar localhost 6789 glass.jpg
      #Message Display on successful connection….Wait for it!
      PERFORM SHARPEN;
      PERFORM SCALE_UP;
      RECEIVE "proc.jpg";
      REPLACE "glass.jpg";
      PERFORM INVERT_COLOR;
      RECEIVE "proc1.jpg";
      CLOSE;
      (Press Ctrl-D if Linux or Ctrl-Z if Windows followed by Enter to stop input and query processing)

MANUAL FOR REQUESTS SET

General Instructions

- Whitespaces of any kind are skipped.
- Each request ends with a semi colon.
- Filenames are enclosed in double quotes

PERFORM Request

Syntax:

PERFORM OPERATION ;

OPERATIONS: GAUSSIAN_BLUR | INVERT_COLOR | SHARPEN | BRIGHTEN | DARKEN | SCALE_UP | SCALE_DOWN

REPLACE Request

Syntax:

REPLACE "FILENAME" ;

RECEIVE Request

Syntax:

RECEIVE "FILENAME" ;

CLOSE Request

Syntax:

CLOSE ;