Run Instructions:

1. Import the project ImageResizer into Eclipse from the archive file provided.
2. On line 32 of MasterAllocator.java, change the third argument to the location of the bin folder of the project.
3. Right click the project, and go to *Run Configurations*.
4. Create a new Java Application Configuration for this project, select the MasterAllocator class as the Main class.
5. Go to the arguments tab, and put the arguments in the following order:

<Input Directory of Images> <Output Directory for resized Images> M N

1. Run using this configuration.

Algorithm:

The main method of the MasterAllocator class computes the number of slave processes to be spawned as #images/m, where m is the threshold provided as a command line argument (we have found 1600 to be a suitable value for our size of dataset), and spawns them. Then it creates a server socket that listens for connections from the slave processes. On a successful connection, it sends a string containing names of 200 (heuristically determined) images concatenated together and sends it.

Having a reasonably large threshold m ensures that too many processes do not get spawned since process context switching is very time consuming, especially on a single system.

The intuition behind sending 200 images is twofold:

1. Some processes get scheduled more often and receive more CPU time. These processes should do more work, rather than dividing the #images equally among all processes at the beginning.
2. The amount of data sent and consequently I/O blocking in the start of the program, when no other processes can run is reduced.

The slave process connects to the server socket, receives the string and tokenizes it. It starts a threadpool, the size of which is min(#cores, #img\_rcvd/N), where N is the threshold for the number of images per thread. It initializes the threadpool and passes successive string tokens to each thread which opens and resizes the image using the Graphics2D class and writes the resized image to the output directory. Once all string tokens are consumed, it tries to reconnect to the master and receive more images. When all images have been resized, the master sends a 0 which is recognized by the slave processes as the end of the program and they exit, after which the master decrements its count of the running processes. Once that reaches 0, the master process exits.