**Running Program**

I developed this using Eclipse IDE and JDK 8. The solution folder contains source files in src directory. We do not need Eclipse to be installed to run this program. The build files are in bin directory, we can run the program from terminal using the following command:

java -cp bin EmailSender [CSVFileName]

If Eclipse is installed, the program can be directly imported and run via Eclipse IDE. Readme.txt file is there with more information.

I have used semicolon(;) as a column separator and new line(\n) as a row separator for CSV file. For example: a typical row of a csv file looks something like this:

stefan@bw.de;STEFAN;SCHROEDINGER

**Implementation**

* **File Generation (Utility.java)**

For testing purpose, I have create a sample utility which creates a random string data. This random string is used to define first name, email, last name.

* **CSV File Reading (EmailSender.java)**

Since the input CSV file could be huge(gigabytes), loading the file as a whole into RAM could not be enough. So, I have implemented BufferedReader to read a line at time, this is pretty efficient than Scanner. Scanner does some other processing also, which makes Scanner less efficient.

* **Email Service (EmailService.java)**

For each line of CSV file, after parsing carried out, the email service carries out task of sending email to corresponding recipient. Since we can have huge recipients, so care should be take regarding performance, we can not do this serially. Since, implementation in threads is required which can fully utilize the power multi-core architecture. Huge number of recipients means we needed huge threads, I have used Java’s ThreadPoolExecutor where we can define a pool of jobs (stored in queue) to be executed at time. I have defined the Queue Limit which maintains the size of queue, so reduce the blocking threads. This has performance advantage also because this limits the job queue being huge.

If queue has job more than limit, the service does not accept new job. This can be checked if the service is ready to accept a new task.

* **Email Sending Job (SendEmailJob.java)**

This is actual job to send email. This job is executed by email service mentioned above. The actual email sending algorithm should be implemented here.

* **Logging**

I have implemented Java Logging API to log email sending status. Currently, it logs to console output, if need, we can log to a file also.

**Result**

The processing of smaller CSV file, is quite faster. While huge CSV files takes time to finish (I have defined half second sleep as a mock to send an email). The program runs quite smoothly without any memory or CPU related issues.