HOMEWORK 3 AREP (Taller Clientes y Servicios)

By : Juan Camilo Ortiz Medina

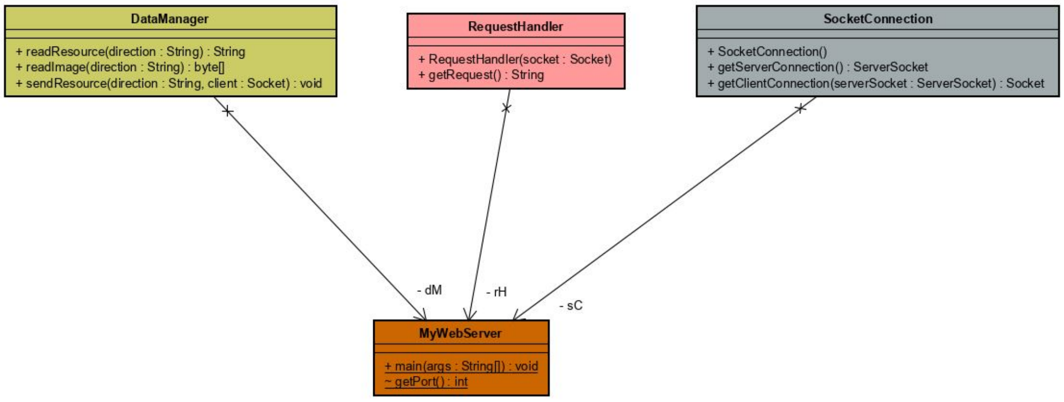
23 August 2020

1 Introduction

This workshop presents different challenges that will help you explore the concepts of naming schemes and clients and services..

2 Design

2.1 CHALLENGE 1:


DataManager
+ readResource(direction : String) : String
+ readImage(direction : String) : byte[]
+ sendResource(direction : String, client : Socket) : void
RequestHandler
+ RequestHandler(socket : Socket)
+ getRequest() : String
SocketConnection
+ SocketConnection()
+ getServerConnection() : ServerSocket
+ getClientConnection(serverSocket : ServerSocket) : Socket
- dM
- rH
- sC
MyWebServer
+ main(args : String[]) : void
~ getPort() : int

-DataManager-

readResource

- @param direction Name of the file, without path.
- @return file, html file.

readImage

- @param direction name of the image that you need to read.
- @return finalData, image in byte array.

sendResource

- @param direction name of the resource that server needs to send.
- @param client Client Socket to know where to send the resource.

-RequestHandler-

RequestHandler

- @param socket this is the socket that the RequestHandler needs to review request.
- @return finalResource the name of the file that client needs.

getRequest

- @return name of the request made by the client throught GET petitions.

-SocketConnection-

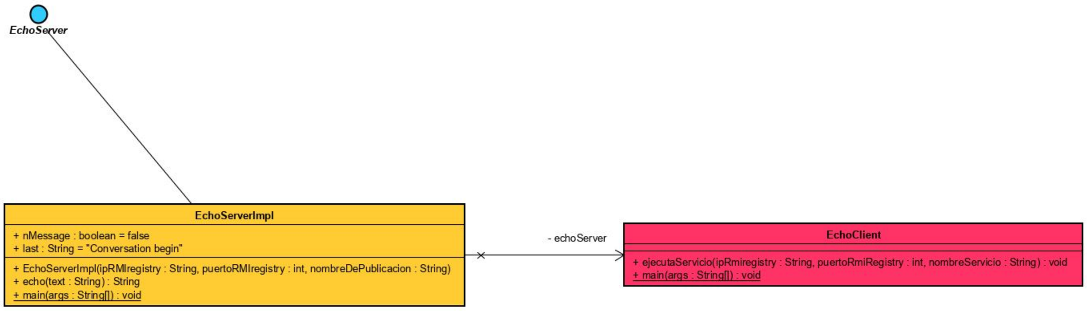
getServerConnection

- @return socket server.

getClientConnection

- @param serverSocket he client will connect to this server.
- @return clientSocket clientSocket Socket del cliente.

2.2 CHALLENGE 2:


EchoServer
EchoServerImpl
+ nMessage : boolean = false
+ last : String = "Conversation begin"
+ EchoServerImpl(ipRMIregistry : String, puertoRMIregistry : int, nombreDePublicacion : String)
+ echo(text : String) : String
+ main(args : String[]) : void
- echoServer
EchoClient
+ ejecutaServicio(ipRmiregistry : String, puertoRmiRegistry : int, nombreServicio : String) : void
+ main(args : String[]) : void

-EchoClient-

ejecutaServicio

- run the service.

echo

- echo server implementation.

EchoServer

- (Interface)

3 Start

Copy the project through git clone in any direction to start working:

```
git clone https://github.com/Juaco9502/Taller-Clientes-y-Servicios—AREP.git
```

4 Pre-Requisites

- Java jdk 7 or more.
- Maven (Apache Maven)
- Git

5 Install

5.1 Run in terminal:

This command works to compile the program:

```
mvn package
```

5.2 Documentation(Optional):

If you want to view the documentation of the application, execute the following command and check the following path: root / target:

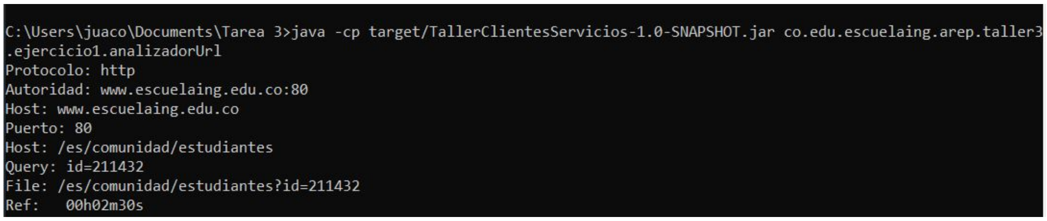
```
mvn javadoc:javadoc
```

5.3 Run:

5.3.1 EXERCISE 1

Write a program in which you create a URL object and print each of the components of a URL to the screen. That is, you must use the following methods: getProtocol, getAuthority, getHost, getPort, getPath, getQuery, getFile, getRef. Make sure neither prints an empty string, this implies that the URL you use for your object must contain enough data.

```
java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar
co.edu.escuelaing.arep.taller3.ejercicio1.analizadorUrl
```

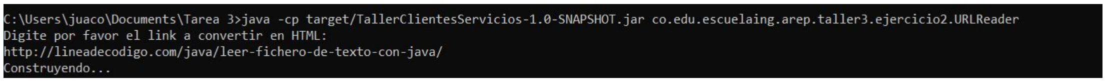


```
C:\Users\juaco\Documents\Tarea 3>java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3
.ejercicio1.analizadorUrl
Protocolo: http
Autoridad: www.escuelaing.edu.co:80
Host: www.escuelaing.edu.co
Puerto: 80
Host: /es/comunidad/estudiantes
Query: id=211432
File: /es/comunidad/estudiantes?id=211432
Ref:   00h02m30s
```

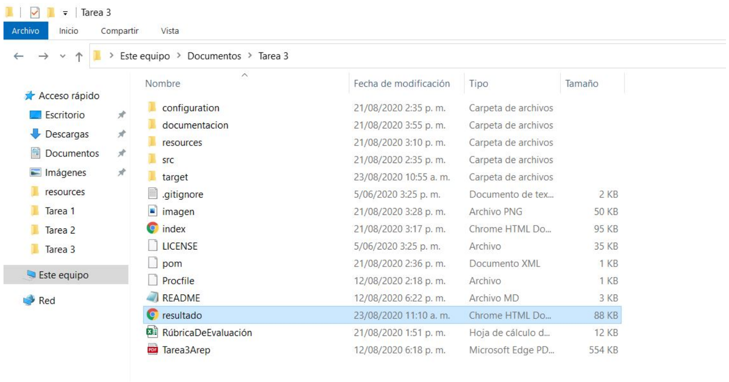
5.3.2 EXERCISE 2

Write a browser application that, given a URL, reads data from that address and stores it in a file with the name result.html. Try to view this file in the browser. Your implementation must be a program that receives the URL parameter via the command line.

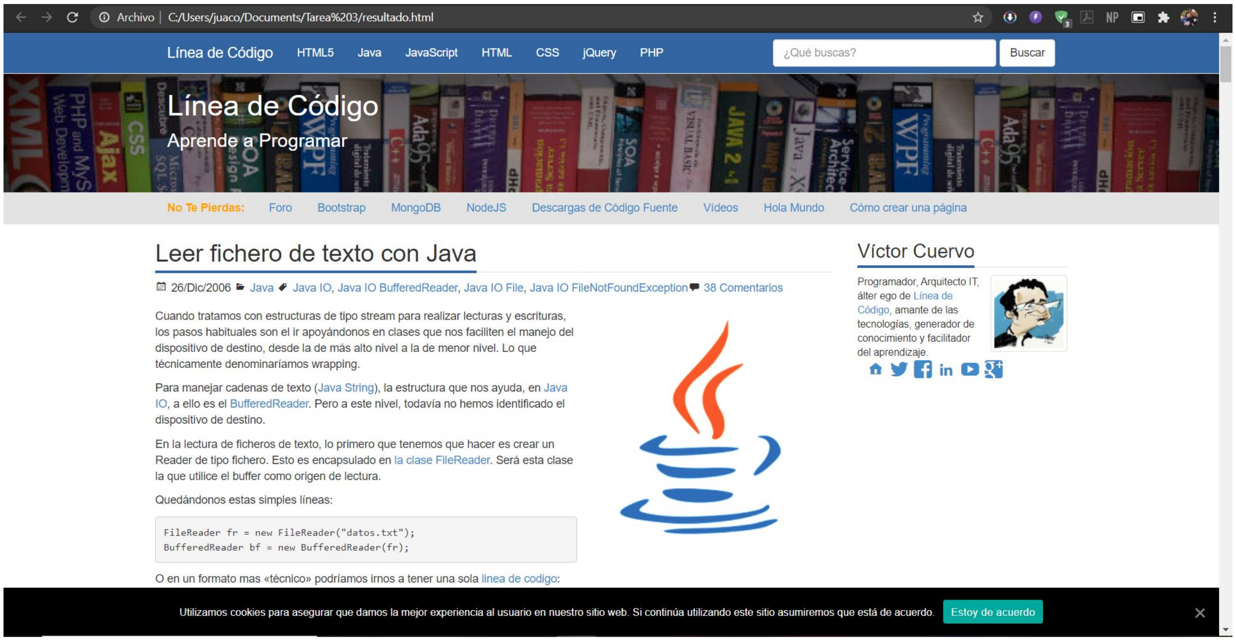
```
java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar
co.edu.escuelaing.arep.taller3.ejercicio2.URLReader
```



```
C:\Users\juaco\Documents\Tarea 3>java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.ejercicio2.URLReader
Digite por favor el link a convertir en HTML:
http://lineadecodigo.com/java/leer-fichero-de-texto-con-java/
Construyendo...
```



Nombre	Fecha de modificación	Tipo	Tamaño
configuration	21/08/2020 2:35 p. m.	Carpeta de archivos	
documentacion	21/08/2020 3:55 p. m.	Carpeta de archivos	
resources	21/08/2020 3:10 p. m.	Carpeta de archivos	
src	21/08/2020 2:35 p. m.	Carpeta de archivos	
target	23/08/2020 10:55 a. m.	Carpeta de archivos	
.gitignore	5/06/2020 3:25 p. m.	Documento de tex...	2 KB
imagen	21/08/2020 3:28 p. m.	Archivo PNG	50 KB
index	21/08/2020 3:17 p. m.	Chrome HTML Do...	95 KB
LICENSE	5/06/2020 3:25 p. m.	Archivo	35 KB
pom	21/08/2020 2:36 p. m.	Documento XML	1 KB
Procfile	12/08/2020 2:18 p. m.	Archivo	1 KB
README	12/08/2020 6:22 p. m.	Archivo MD	3 KB
resultado	23/08/2020 11:10 a. m.	Chrome HTML Do...	88 KB
RúbricaDeEvaluación	21/08/2020 1:51 p. m.	Hoja de cálculo d...	12 KB
Tarea3Arep	12/08/2020 6:18 p. m.	Microsoft Edge PD...	554 KB



Línea de Código

Aprende a Programar

Leer fichero de texto con Java

26/Dic/2006 Java Java IO, Java IO BufferedReader, Java IO File, Java IO FileNotFoundException 38 Comentarios

Cuando tratamos con estructuras de tipo stream para realizar lecturas y escrituras, los pasos habituales son el ir apoyándonos en clases que nos faciliten el manejo del dispositivo de destino, desde la de más alto nivel a la de menor nivel. Lo que técnicamente denominaríamos wrapping.

Para manejar cadenas de texto (Java String), la estructura que nos ayuda, en Java IO, a ello es el BufferedReader. Pero a este nivel, todavía no hemos identificado el dispositivo de destino.

En la lectura de ficheros de texto, lo primero que tenemos que hacer es crear un Reader de tipo fichero. Esto es encapsulado en la clase FileReader. Será esta clase la que utilice el buffer como origen de lectura.

Quedándonos estas simples líneas:

```
FileReader fr = new FileReader("datos.txt");
BufferedReader bf = new BufferedReader(fr);
```

O en un formato mas «técnico» podríamos irnos a tener una sola linea de codigo:

Víctor Cuervo

Programador, Arquitecto IT, álter ego de Línea de Código, amante de las tecnologías, generador de conocimiento y facilitador del aprendizaje.

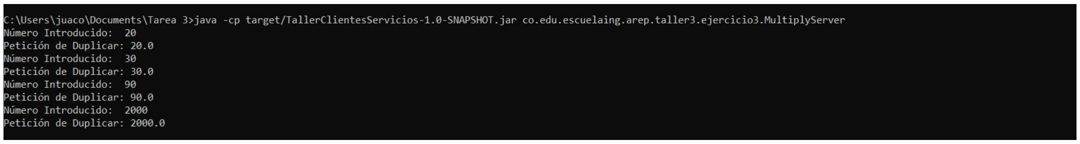
Utilizamos cookies para asegurar que damos la mejor experiencia al usuario en nuestro sitio web. Si continúa utilizando este sitio asumiremos que está de acuerdo.

5.3.3 EXERCISE 3

Using sockets write a server that receives a number and answers the square of this number. Write a customer to test it and send him a sequence of 20 numbers.

Power on the server:

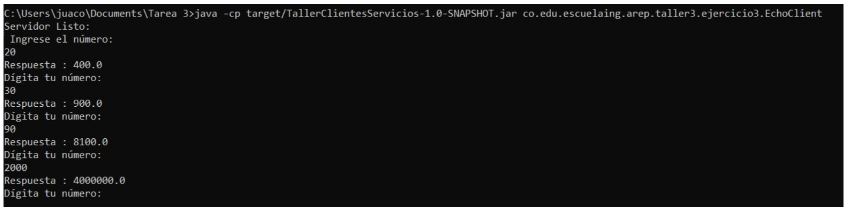
```
java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.ejercicio3.MultiplyServer
```



```
C:\Users\juaco\Documents\Tarea 3>java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.ejercicio3.MultiplyServer
Número Introducido:  20
Petición de Duplicar: 20.0
Número Introducido:  30
Petición de Duplicar: 30.0
Número Introducido:  90
Petición de Duplicar: 90.0
Número Introducido:  2000
Petición de Duplicar: 2000.0
```

Request the number:

```
java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.ejercicio3.EchoClient
```

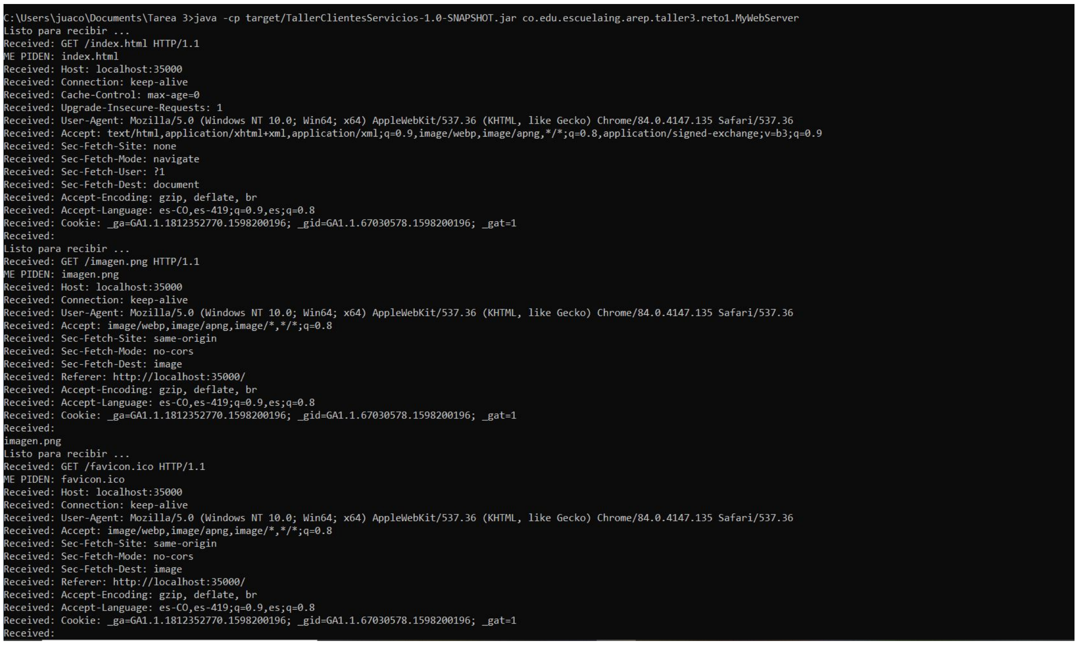


```
C:\Users\juaco\Documents\Tarea 3>java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.ejercicio3.EchoClient
Servidor Listo:
 Ingrese el número:
20
Respuesta : 400.0
Digita tu número:
30
Respuesta : 900.0
Digita tu número:
90
Respuesta : 8100.0
Digita tu número:
2000
Respuesta : 4000000.0
Digita tu número:
```

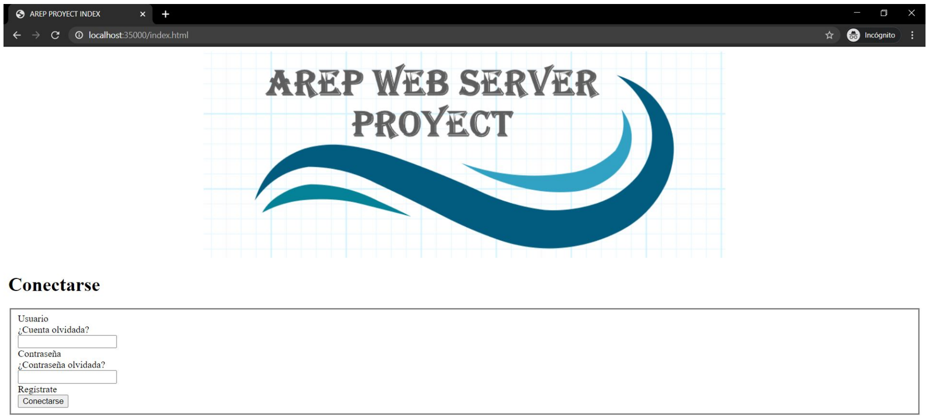
5.3.4 CHALLENGE 1

Write a web server that supports multiple requests in a row (non-concurrent). The server should return all requested files, including html pages and images. Build a website with javascript to test your server. Deploy your solution on Heroku. DO NOT use web frameworks like Spark or Spring use only Java and the libraries for network management.

```
java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.reto1.MyWebServer
```



```
C:\Users\juaco\Documents\Tarea 3>java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.reto1.MyWebServer
Listo para recibir ...
Received: GET /index.html HTTP/1.1
ME PIDEN: index.html
Received: Host: localhost:35000
Received: Connection: keep-alive
Received: Cache-Control: max-age=0
Received: Upgrade-Insecure-Requests: 1
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Received: Sec-Fetch-Site: none
Received: Sec-Fetch-Mode: navigate
Received: Sec-Fetch-User: ?1
Received: Sec-Fetch-Dest: document
Received: Accept-Encoding: gzip, deflate, br
Received: Accept-Language: es-CO,es-419;q=0.9,es;q=0.8
Received: Cookie: _ga=GA1.1.1812352770.1598200196; _gid=GA1.1.67030578.1598200196; _gat=1
Received:
Listo para recibir ...
Received: GET /imagen.png HTTP/1.1
ME PIDEN: imagen.png
Received: Host: localhost:35000
Received: Connection: keep-alive
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36
Received: Accept: image/webp,image/apng,image/*,*/*;q=0.8
Received: Sec-Fetch-Site: same-origin
Received: Sec-Fetch-Mode: no-cors
Received: Sec-Fetch-Dest: image
Received: Referer: http://localhost:35000/
Received: Accept-Encoding: gzip, deflate, br
Received: Accept-Language: es-CO,es-419;q=0.9,es;q=0.8
Received: Cookie: _ga=GA1.1.1812352770.1598200196; _gid=GA1.1.67030578.1598200196; _gat=1
Received:
imagen.png
Listo para recibir ...
Received: GET /favicon.ico HTTP/1.1
ME PIDEN: favicon.ico
Received: Host: localhost:35000
Received: Connection: keep-alive
Received: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36
Received: Accept: image/webp,image/apng,image/*,*/*;q=0.8
Received: Sec-Fetch-Site: same-origin
Received: Sec-Fetch-Mode: no-cors
Received: Sec-Fetch-Dest: image
Received: Referer: http://localhost:35000/
Received: Accept-Encoding: gzip, deflate, br
Received: Accept-Language: es-CO,es-419;q=0.9,es;q=0.8
Received: Cookie: _ga=GA1.1.1812352770.1598200196; _gid=GA1.1.67030578.1598200196; _gat=1
Received:
```



Conectarse

Usuario
¿Cuenta olvidada?

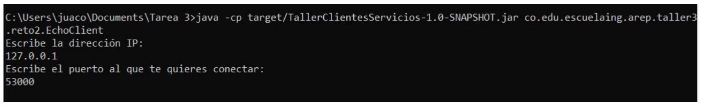
Contraseña
¿Contraseña olvidada?

Regístrate
Conectarse

5.3.5 CHALLENGE 2

Using your server and java (DO NOT use web frameworks like Spark or Spring). Write a Spark-like framework that allows you to publish ”get” web services with lambda functions and allows you to access static resources such as pages, javascripts, images, and CSSs. Create an application that connects to a database from the server to test your solution. Deploy your solution on Heroku.

```
java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3.reto2.EchoClient
```



```
C:\Users\juaco\Documents\Tarea 3>java -cp target/TallerClientesServicios-1.0-SNAPSHOT.jar co.edu.escuelaing.arep.taller3
.reto2.EchoClient
Escribe la dirección IP:
127.0.0.1
Escribe el puerto al que te quieres conectar:
53000
```

5.4 HEROKU:

Use the following link in the browser:

```
https://tarea3-arep.herokuapp.com/
```

6 Pre-Requisites

- Maven - Dependency Management
- JAVA JDK 8 - Building
- JUnit 3.8.1 - Test

7 License

This project is licensed under the GNU General Public License.