Práctica 4: Carga Dinámica usando RMI.

Práctica 4. (Será realizada en la Sala de Computo)

El objetivo de esta práctica es conocer el proceso para ejecutar una aplicación distribuida usando la configuración Cliente Bootstrapped y Servidor Bootstrapped. En esta configuración todo el código del cliente, y todo el código del servidor es cargado, vía un pequeño programa, en la máquina cliente o la máquina servidora. Este potente mecanismo permite que un proceso pueda usar el código de clases que no estaban presentes en su JVM cuando inició su ejecución, descargándose automáticamente en tiempo de ejecución desde otra JVM.

En una aplicación basada en RMI se puede incluir la carga dinámica de clases desde servidores vía protocolos FTP o HTTPS. Esta es una característica poderosa ya que las clases pueden estar almacenadas en un servidor, y todos los nodos de un sistema RMI pueden descargar estas clases para operar.

La aplicación a la cual se le aplicará la Carga Dinámica será sobre un ejemplo de "Calculadora". La lógica de esta aplicación es simple con el fin manejar la configuración Bootstrapped tanto en el lado cliente como en el servidor, ver figura 1.

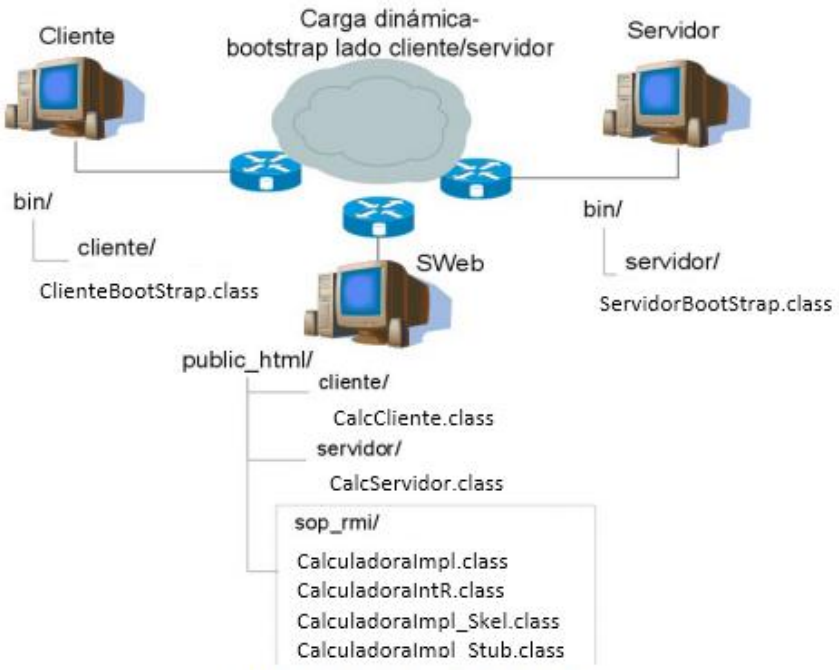

Figura 1: Diagrama de red enriquecido

Tener en cuenta: La solución de este ejercicio debe ser comprimida en formato rar y enviada a la cuenta de correo electrónico del docente. El nombre del archivo comprimido debe seguir el siguiente formato lsd_rmi_p4_apellidoN1_apellidoN2.rar. Donde apellidoN1 corresponde al primer apellido de uno de los integrantes y apellidoN2 corresponde al primer apellido del segundo integrante del grupo. En la carpeta se deben especificar el rol de cada integrante.

Para está práctica se deben descargar el archivo ii-2019-fuentes-calculadora.rar del sitio destinado al curso. Estos archivos se deben organizar en una estructura de directorios de trabajo donde se ubicaran los archivos fuente (directorio 'src') y los archivos binarios (bytecode) (directorio bin) ver figura 2.

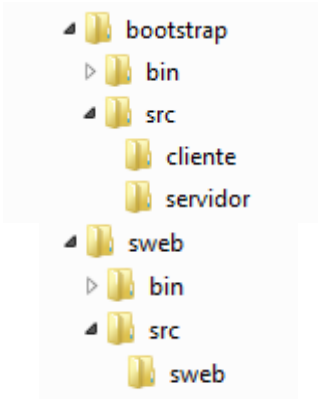

Figura 2. Estructura del directorio de trabajo.

Para esta aplicación se deben crear tres directorios. Cada uno de estos directorios debe seguir la estructura que se muestra en la Figura 2. Los directorios a ubicar son los siguientes:

a. **sweb:** Directorio donde se almacenan las clases de un Servidor Web básico (ClassFileServer, ClassServer).

b. boostrap: Contendrá los programas bootstrap del cliente (ClienteBootStrap.java) y boostrap del servidor (ServidorBootStrap.java). **Se deben modificar con el fin que se cargue el cliente y el servidor que se están usando.**

c. lsd-rmi-p4: Contiene las clases cliente y servidor de la aplicación Calculadora.

d. **public_html:** Directorio utilizado por el Servidor Web para compartir los archivos públicos, en este caso correspondería a los archivos .class de la aplicación. Se debe ubicar en la ruta c:\public_html si es Windows, o home\ public_html si es linux

Para la ejecución de la aplicación Calculadora se deben compilar todos los archivos fuente (.java) de cada uno de los directorios descritos anteriormente, utilizando el siguiente comando de compilación:

```
javac -d ..\bin nom_dir\*.java
```

Con el propósito de realizar la comparación de la ejecución de un programa usando el modo de configuración Cerrado y el modo de configuración Bootstrapped, a continuación se describe el procedimiento a seguir en cada uno de los modos de configuración.

1. Ejecución usando la configuración cerrada.
2. Ejecución usando la configuración Bootstrapped en el lado cliente y el lado servidor.

1. Ejecución usando la configuración cerrada.

En este tipo de configuración no hay Carga Dinámica y todas las clases son ubicadas en su respectiva máquina (máquina cliente y máquina servidor). Procedimiento de ejecución: Ubicarse en el subdirectorio 'bin/' del directorio 'lsd-rmi-p4'.

a) Lanzar el N_S:

```
rmiregistry 8080
```

b) Lanzar el programa servidor:

```
java servidor. CalcServidor
```

c) Lanzar el programa cliente:

```
java cliente. CalcClienteCerrado
```

2. Ejecución usando la configuración bootstrapped (Del lado cliente y servidor).

a) Ejecutar el Servidor Web:

Ubicarse en el sudirectorio 'bin' del directorio 'sweb'. En este caso se utilizará un pequeño servidor HTTP proporcionado por oracle. La ejecución de este programa requiere dos parámetros de entrada: el número de puerto por donde estará escuchando este servidor y el segundo parámetro es el directorio donde se compartirán los archivos. Ubicarse en el directorio 'bin/' de 'sweb/':

```
java sweb.ClassFileServer 2080 c:\public_html
```

Nota: Los archivos ubicados en el subdirectorio 'bin' del directorio 'lsd-rmi-p4' deben ser copiados en el directorio c:\public_html

b) Lanzar el N_S:

Ubicarse en el subdirectorio 'bin' del directorio 'bootstrap'. Lanzar el N_S.

```
rmiregistry 8080 -J-Djava.rmi.server.codebase=http://localhost:2080/
```

c) Ejecutar los programas bootstrap del lado Servidor y del lado Cliente:

Ubicarse en el subdirectorio 'bin' del directorio 'bootstrap'. Copiar, en este directorio, el archivo de políticas calc.policy que se proporciona junto con los archivos fuentes.

Los programas bootstrap del lado cliente y del lado servidor hacen uso de dos clases, las cuales juegan un rol importante en el proceso de Carga Dinámica. Estas clases son **java.rmi.SecurityManager y java.rmi.server.RMIClassLoader**. El gestor de seguridad de RMI habilita las restricciones de seguridad indicadas en un archivo de políticas. El sistema RMI solo descargará clases desde una ubicación remota, si un gestor de seguridad RMI ha sido fijado. La clase RMIClassLoader tiene el siguiente método:

public static Class loadClass(String codebase, String name)

Este método carga las clases desde el lugar especificado en el codebase, por lo tanto para la ejecución de los programas bootstrap es necesario indicarle al sistema RMI la ubicación del archivo de políticas mediante la propiedad java.security.policy y el codebase mediante la propiedad java.rmi.server.codebase:

La propiedad java.rmi.server.codebase es usada para especificar una url que puede ser un archivo, o un servidor ftp o http, el cual establece la localización de las clases que serán descargadas a la JVM.

Comando de ejecución del programa bootstrap de lado servidor:

```
java -Djava.security.policy=calc.policy -Djava.rmi.server.codebase=http://localhost:2080/
servidor.ServidorBootStrap
```

Comando de ejecución del programa bootstrap de lado cliente:

```
java -Djava.security.policy=calc.policy -Djava.rmi.server.codebase=http://localhost:2080/
cliente.ClienteBootStrap
```