Hypervisor Remote Control: HRC

Manual de Desarrollo

Carlos Eduardo Gómez Montoya[1,2], PhD(c).
Jaime Alberto Chavarriaga Lozano[1], PhD.
Harold Enrique Castro Barrera[1], PhD.

[1]Universidad de Los Andes
[2]Universidad del Quindío
2018

Tabla de contenidos

Lista de tablas

Capítulo 1. INTRODUCCION

Hypervisor Remote Control es un sistema creado para facilitar la administración remota de máquinas virtuales (MVs) a través de comandos para automatizar las funciones principales del hipervisor.

El sistema tiene una arquitectura cliente-servidor. En cada computador donde se ejecute el hipervisor, se ejecuta también el programa servidor (HRCServer) el cual queda a la espera de comandos enviados por el programa cliente (HRCClient) para su ejecución local. El caso de uso principal de este sistema es la posibilidad de enviar una secuencia de comandos desde el programa cliente para administrar MVs en una sala de cómputo. El cliente especifica el comando y el parámetro, y el servidor traduce el comando a uno que entienda el hipervisor y pueda ejecutarlo.

Mediante el HRC se puede, por ejemplo, encender o apagar una MV, enviando el comando desde un computador para que la MV especificada se encienda o apague en el computador donde está alojada. Dado que el sistema se ejecuta a través de la línea de comandos del sistema operativo, estos comandos se pueden incluir en un script para automatizar fácilmente un grupo de MVs en una sala de cómputo. Adicionalmente, el sistema permite enviar archivos al servidor y, para extender aún más su funcionalidad, permite que el programa cliente envíe un comando para ejecutar un script previamente creado para que se ejecute de manera local en el computador donde corre el programa HRCServer. De esta forma, funciones que no estén incluidas en el HRC pueden ser ejecutadas sin necesidad de modificar el código fuente del sistema.

La aplicación Hypervisor Remote Control, simple desde el punto de vista del desarrollo de software, es bastante útil para la administración efectiva de MVs, ya que la automatización de tareas ahorra tiempo significativo y reduce la probabilidad de errores.

Capítulo 2. configuration

2.1. Configuration

This class reads the properties that are required in the application, which can be changed by user.

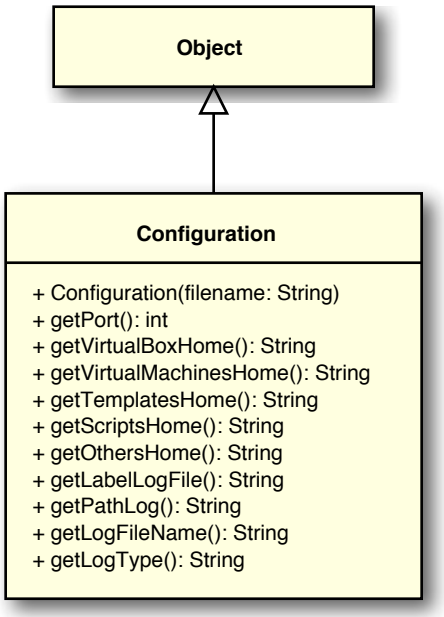
2.1.1. Resumen

```
public class Configuration {

  // Costructor Publico
  public Configuration(String filename);

  // Metodos Publicos
  public String getLabelLogFile();
  public String getLogFileName();
  public String getLogType();
  public String getOthersHome();
  public String getPathLog();
  public int getPort();
  public String getScriptsHome();
  public String getTemplatesHome();
  public String getVirtualBoxHome();
  public String getVirtualMachinesHome();
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



2.1.2. Configuration(String)

```
public Configuration(String filename);
```

This is the constructor. Load the configurations.

Parámetros	
String	The properties filename.

2.1.3. getLabelLogFile()

```
public String getLabelLogFile();
```

This method returns the label of each entry in the log file.

Parámetros	
return	String The label.

2.1.4. getLogFileName()

```
public String getLogFileName();
```

This method returns the log file name.

Parámetros	
return	String The log file name.

2.1.5. getLogType()

```
public String getLogType();
```

This method returns the log type.

Parámetros	
return	String The type of the log file.

2.1.6. getOthersHome()

```
public String getOthersHome();
```

This method returns the directory path where the other files stored are.

Parámetros	
return	String The directory path of the files repository.

2.1.7. getPathLog()

```
public String getPathLog();
```

This method returns the directory path where the log file will be stored.

Parámetros	
return	String The pathLog.

2.1.8. getPort()

```
public int getPort();
```

This method returns the port number.

Parámetros	
return	int The port number.

2.1.9. getScriptsHome()

```
public String getScriptsHome();
```

This method returns the directory path where the scripts stored are.

Parámetros	
return	String The directory path of the scripts repository.

2.1.10. getTemplatesHome()

```
public String getTemplatesHome();
```

This method returns the directory path where the templates stored are.

Parámetros	
return	String The directory path of the templates repository.

2.1.11. getVirtualBoxHome()

```
public String getVirtualBoxHome();
```

This method returns the directory path where VirtualBox installed is.

Parámetros	
return	String The directory path of VirtualBox.

2.1.12. getVirtualMachinesHome()

```
public String getVirtualMachinesHome();
```

This method returns the directory path where the virtual machines stored are.

Parámetros	
return	String The directory path of the virtual machines repository.

Capítulo 3. main

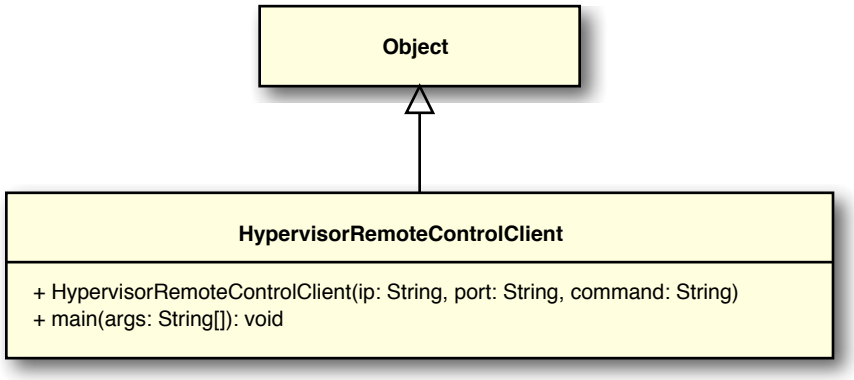
3.1. HypervisorRemoteControlClient

```
public class HypervisorRemoteControlClient {

  // Costructor Publico
  public HypervisorRemoteControlClient(String ip, String port, String command);

  // Metodos Estáticos y Publicos
  public static void main(String[] args);
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



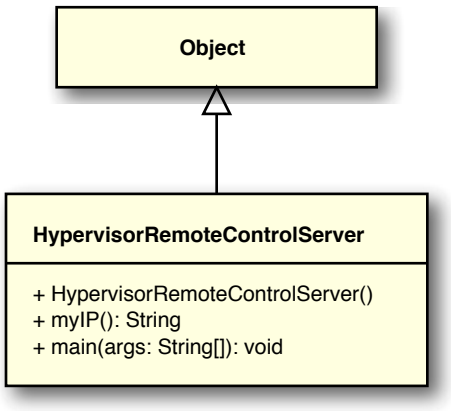
3.2. HypervisorRemoteControlServer

```
public class HypervisorRemoteControlServer {

  // Costructor Publico
  public HypervisorRemoteControlServer();

  // Metodos Estáticos y Publicos
  public static void main(String[] args);
  public static String myIP();
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



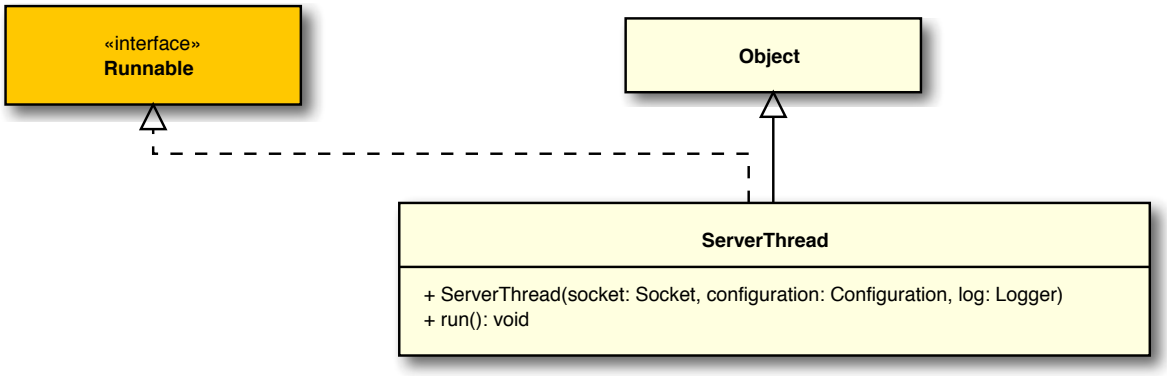
3.3. ServerThread

```
public class ServerThread implements Runnable {

  // Costructor Publico
  public ServerThread(Socket socket, Configuration configuration, Logger log);

  // Metodos Publicos
  public void run();
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



Capítulo 4. util

4.1. Constants

This class gathers all the constants that are required in the name server application.

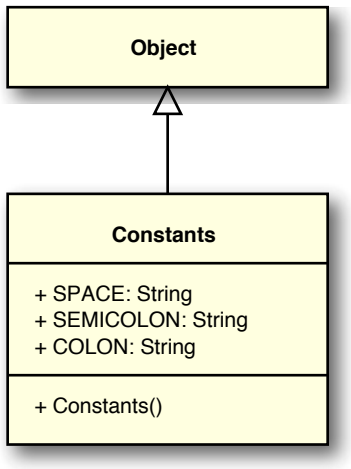
4.1.1. Resumen

```
public class Constants {

  // Campos Estáticos y Públicos
  public static final String COLON = ":";
  public static final String SEMICOLON = ";";
  public static final String SPACE = " ";

  // Costructor Publico
  public Constants();
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



4.2. Digest

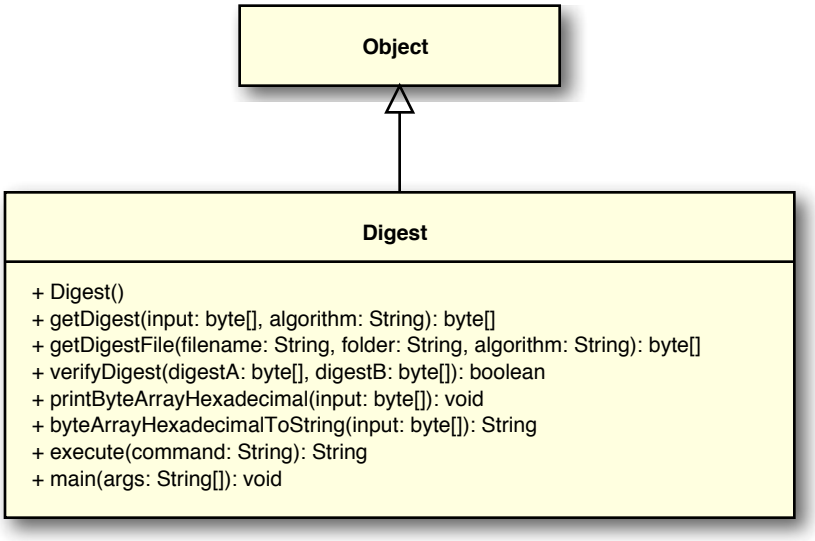
```
public class Digest {

  // Costructor Publico
  public Digest();

  // Metodos Estáticos y Publicos
  public static String byteArrayHexadecimalToString(byte[] input);
  public static String execute(String command);
  public static byte[] getDigest(byte[] input, String algorithm);
```

```
  public static byte[] getDigestFile(String filename,
                                     String folder,
                                     String algorithm);
  public static void main(String[] args);
  public static void printByteArrayHexadecimal(byte[] input);
  public static boolean verifyDigest(byte[] digestA, byte[] digestB);
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



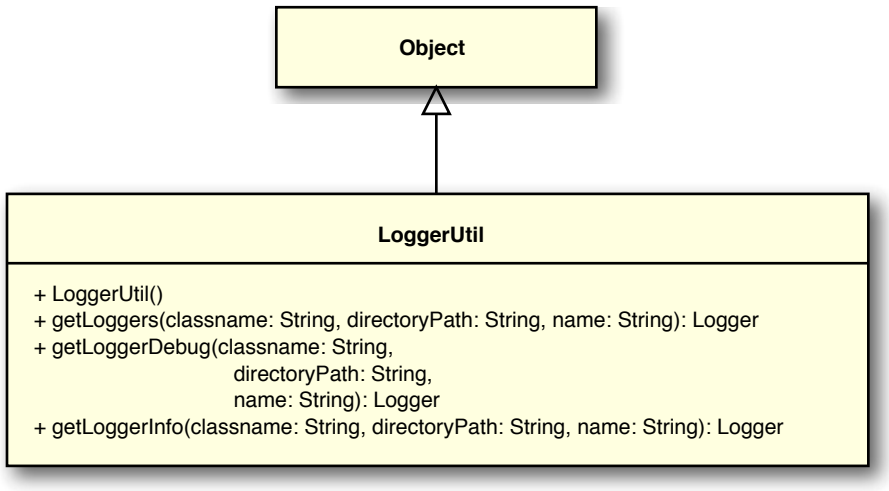
4.3. LoggerUtil

```
public class LoggerUtil {

  // Costructor Publico
  public LoggerUtil();

  // Metodos Estáticos y Publicos
  public static Logger getLoggerDebug(String classname,
                                      String directoryPath,
                                      String name)
    throws IOException;
  public static Logger getLoggerInfo(String classname,
                                     String directoryPath,
                                     String name)
    throws IOException;
  public static Logger getLoggers(String classname,
                                  String directoryPath,
                                  String name)
    throws IOException;
}
```

Metodos internos de java.lang.Object: `equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`



4.3.1. getLoggerDebug(String, String, String)

```
public static Logger getLoggerDebug(String classname,
                                     String directoryPath,
                                     String name)
  throws IOException;
```

This method return a logger to write in a debug file and the console.

Parámetros	
classname	It is the name of the class that appears in each entry of the log file.
directoryPath	It is the path of the directory where the log file will be stored.
name	It is the name of the log file.
return	Logger It is the configured logger to use in any part of the source code.

Excepciones

`IOException`

4.3.2. getLoggerInfo(String, String, String)

```
public static Logger getLoggerInfo(String classname,
                                   String directoryPath,
                                   String name)
  throws IOException;
```

This method return a logger to write in an info file and the console.

Parámetros	
classname	It is the name of the class that appears in each entry of the log file.
directoryPath	It is the path of the directory where the log file will be stored.
name	It is the name of the log file.
return	Logger It is the configured logger to use in any part of the source code.

Excepciones

`IOException`

4.3.3. getLoggers(String, String, String)

```
public static Logger getLoggers(String classname,
                                String directoryPath,
                                String name)
  throws IOException;
```

This method return a logger to write in two files (debug and info) and the console.

Parámetros	
classname	It is the name of the class that appears in each entry of the log file.
directoryPath	It is the path of the directory where the log file will be stored.
name	It is the name of the log file.
return	Logger It is the configured logger to use in any part of the source code.

Excepciones

`IOException`

Apéndice A. Valores de campos constantes

A.1. util.*

Tabla A.1. Constants

COLON	":"
SEMICOLON	";"
SPACE	" "

Índice

S