**Universidad ORT Uruguay**

**Facultad de Ingeniería**

Aspectos de Seguridad - Obligatorio

Entregado como requisito para la obtención del crédito Aspectos de Seguridad

Segundo Semestre 2019

Marcela Ferraz - 200112

Mauricio Zito - 111725

Docentes: Santiago Paz, Felipe Sotuyo

Tabla de contenido

[Introducción 3](#_Toc26702124)

[Tecnología utilizada 3](#_Toc26702125)

[Parte 1 – Sistema seguro Bell Lapadula 4](#_Toc26702126)

[1.1 Desarrollo 4](#_Toc26702127)

[1.2 Instalación 6](#_Toc26702128)

[1.3 Prueba 9](#_Toc26702129)

[1.4 Conclusión 9](#_Toc26702130)

[Parte 2 – Canal Encubierto 10](#_Toc26702131)

[2.1 Desarrollo 10](#_Toc26702132)

[2.2 Instalación 10](#_Toc26702133)

[2.3 Prueba 10](#_Toc26702134)

[2.4 Conclusión 10](#_Toc26702135)

[Conclusión Final 10](#_Toc26702136)

# Introducción

En el marco de la materia Aspectos de seguridad, se solicita un trabajo práctico en caracter de obligatorio donde se implementará un modelo de seguridad basado en el planteado por la dupla de Billy Elliott Bell y Len LaPadula, consiste en dividir el permiso de [acceso](https://es.wikipedia.org/wiki/Acceso) de los usuarios a la información en función de etiquetas de seguridad. Por ejemplo, en sistemas militares norteamericanos, categorizándola en 4 niveles: no clasificado, confidencial, secreto y ultrasecreto.

El modelo define 2 reglas de control de acceso mandatorio (MAC):

Propiedad de seguridad simple: Un sujeto de un determinado nivel de seguridad no puede leer un objeto perteneciente a un nivel de seguridad más alto.

Propiedad \*: Un sujeto de un determinado nivel de seguridad no puede escribir un objeto perteneciente a un nivel de seguridad más bajo. (También llamada propiedad de confinamiento).

## Tecnología utilizada

El obligatorio fue implementado utilizando:

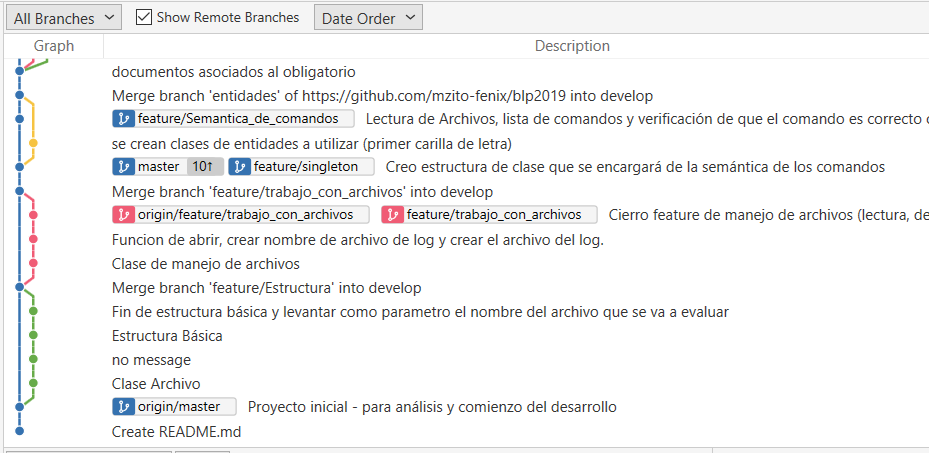
* Lenguaje Java
* IDE Netbeans versión 8.0.2

Version de Java

* JDK 1.8

Control de versiones, desarrollo compartido y organización de la tarea

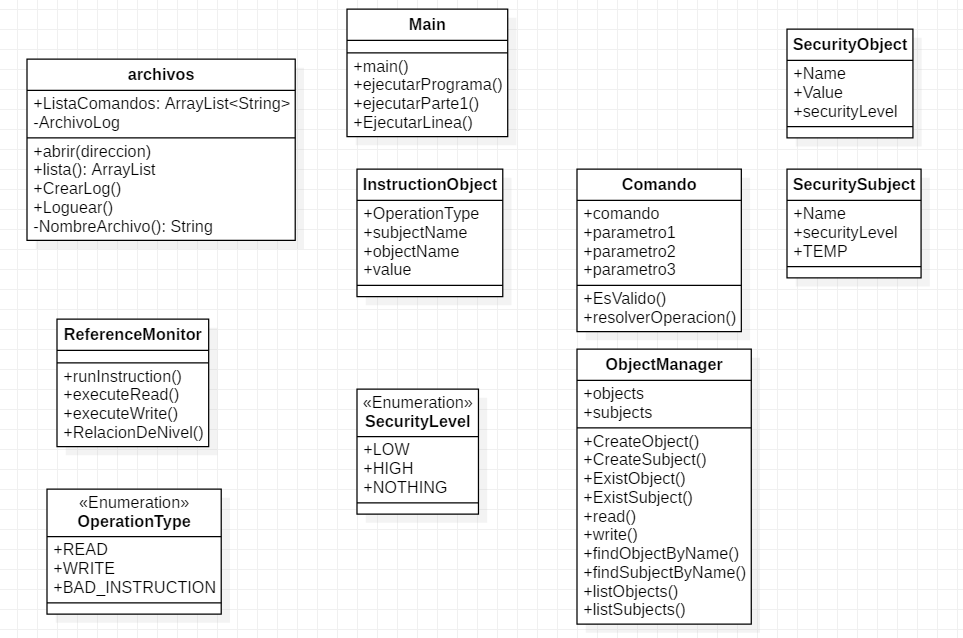
* Utilizamos github y sourcetree para poder trabajar en conjunto



## Parte 1 – Sistema seguro Bell Lapadula

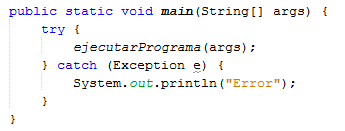
### Desarrollo

Listado de clases utilizadas



|  |  |  |  |
| --- | --- | --- | --- |
|  | archivos | Archivos | Clase que controla la lectura y escritura de archivos (lectura de instrucciones y escritura de log), interacción con el sistema operativo. |
| Entities | InstructionObject | Clase que resume información para la ejecución del comando |
| OperationType | Enumeración de tipos de operaciones.  READ, WRITE, BAD\_INSTRUCTION |
| ReferenceMonitor | Control de instrucciones, control de relación de permiso y ejecución. |
| SecurityLevel | Enumeración de tipos de niveles de seguridad  LOW, HIGH, NOTHING |
| SecurityObject | Representación de un Objeto del modelo Bell Lapadula |
| SecuritySubject | Representación de un Sujeto del modelo Bell Lapadula |
| Comando | Encargado de la verificación de correctitud y separación de las operaciones y parámetros para su posterior ejecución |
| Main | Main | Programa principal, funcion Main. |
| Management | ObjectManager | Controlador que contiene el comportamiento de una base de datos para registrar el listado de sujetos, objetos y sus estados |

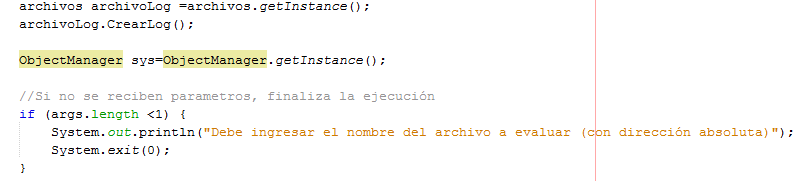
**La función principal**



La función “ejecutarPrograma” recibe los parámetros recibidos por linea de comandos directo del main.



Varias de las funciones fueron implementadas con el patrón “Singleton”, de ésta forma es posible utilizarlas de forma transversal desde diferentes secciones del proyecto.





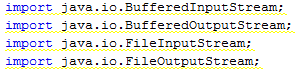
### Instalación

Para realizar la prueba, solo hay que copiar la carpeta “test” que se encuentra dentro del proyecto a cualquier carpeta del equipo destino.

|  |  |
| --- | --- |
|  | En la misma carpeta debe estar el archivo “InstructionList.txt” que es donde se encuentran las instrucciones a ejecutar. |

#### Sobre el Log

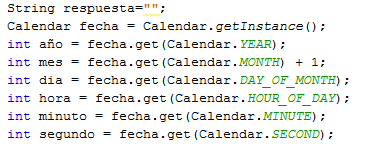
Se implementó una clase para realizar el log del sistema, utilizando las siguientes librerías.



Se compone de 2 partes

1. La creación del archivo de log

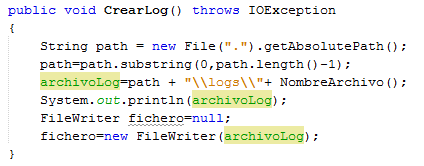
En la subcarpeta “logs” del proyecto se genera un archivo bajo el siguiente patrón de nombre



Se compone de la palabra “Log\_” seguido de Año, Mes, Dia, Hora, minuto y segundo y le asigna la extensión “.log” de ésta forma nos aseguramos que los logs se puedan acumular y no se pierdan registros de cada ejecución independiente.

**Crear Log**

La función CrearLog se encarga de crear un archivo de registro con nombre único y por única vez para toda la ejecución.

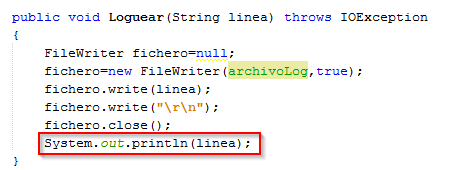




1. El Append de nueva linea al archivo de log

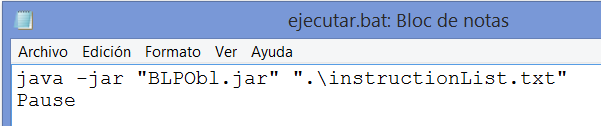
Es la función de la clase que se encarga de hacer “Append” al archivo utilizado y designado por la instancia de ejecución actual.

Tiene la doble funcionalidad de dejar registrado en un archivo la salida y tambien de mostrar en pantalla (por consola) lo que el programa va realizando.



#### 1.2.2 Ejecutar

Para correr el programa se debe ejecutar utilizando el archivo “Parte1.bat” que tiene el siguiente contenido.



### 1.3 Prueba

#### Verificación

|  |  |
| --- | --- |
| Archivo | Resultado de cada linea esperada |
|  |  |

|  |  |
| --- | --- |
| Resultado esperado | Resultado tras corrida |
|  |  |

### 1.4 Conclusión

La propuesta de la implementación de un sistema seguro con el concepto básico planteado por Bell y Lapadula fomenta la comprensión del método y las decisiones que se deben tomar para que la información vaya en un sentido seguro.

Si bien en ésta ocasión se trabajaba con números enteros, es posible implementar con documentos completos e incluso encriptados.

## Parte 2 – Canal Encubierto

### 

### Introducción

Un canal encubierto (del inglés covert channel), es un canal que puede ser usado para transferir información desde un usuario de un sistema a otro, usando medios no destinados para este propósito por los desarrolladores del sistema.

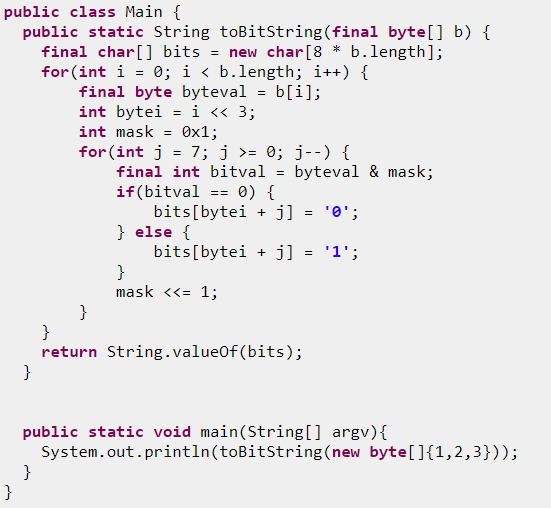
En ésta segunda parte del obligatorio se solicita el desarrollo de un canal encubierto para intentar transferir información entre 2 sujetos de diferentes niveles sorteando las reglas del modelo de Bell Lapadula.

### 2.1 Desarrollo

El mensaje recibido originalmente mediante el archivo XXXXXXXX y por parámetro del sistema, se debe descomponer en caracteres (bytes) y luego en bits (1 y 0) ya que la transferencia se hace en “goteos de bits” para rearmar le mensaje en el destinatario.

Investigación:

Como pasar un array de bytes en un array de bits:



Será necesario confeccionar una función para tales efectos, logrando una cadena de bits desde el texto original.

Fuente:

<http://www.java2s.com/Tutorials/Java/Data_Type/Array_Convert/Convert_byte_array_to_bit_string_in_Java.htm>

### 2.2 Instalación

\*\* Como instalar la app en una carpeta y probarla

### 2.3 Prueba

\*\* como hacer una prueba y mostrar fotos de una prueba realizada

### 2.4 Conclusión

\*\* Conclusión de la parte 2

## Conclusión Final

\*\* Conclusión del trabajo obligatorio completo