Universidad ORT Uruguay
Facultad de Ingeniería

Aspectos de Seguridad de Sistemas Informáticos

Obligatorio

Diciembre 2020

DESCRIPCIÓN DEL SISTEMA

InsecureSystem
SecureSystem
SecureSystemWithCovertChannel

Docente:
Ing. Felipe Sotuyo

Alumnos:
Esteban Muzio 92391
Gianfranco Drago 198490
Matias Hernández 169236

Índice

1. Introducción

1.1. Propósito del documento

El propósito del presente documento es proveer una especificación completa de los sistemas que fueron solicitados implementarse según la letra del obligatorio de la materia Aspectos de Seguridad de Sistemas Informáticos.

Se proveerá información detallada del propósito de cada uno de los siguientes sistemas:

- InsecureSystem
- SecureSystem
- SecureSystemWithCovertChannel

De los mismos, se comentará:

- La descripción del sistema
- El procedimiento de corrida que es necesario realizar para asegurar el correcto funcionamiento del mismo.
- La arquitectura del sistema.
- Las decisiones de diseño que el equipo tuvo que enfrentar.

1.2. Propósito de los sistemas

El propósito de los sistemas es poder entender de manera práctica las diferencias que existen en utilizar un sistema sin ningún nivel de seguridad (InsecureSystem) en comparación con otro sistema con un nivel de seguridad simple (SecureSystem).

También mostrar que pueden existir vulnerabilidades en los sistemas seguros. Para esto se realizará un canal encubierto para transmitir un mensaje desde un sujeto con un nivel de autorización superior a otro sujeto con un nivel de autorización inferior (SecureSystemWithCovertChannel).

1.3. Repositorio

Repositorio en Github: https://github.com/estebanequis/seguridad

2. InsecureSystem

2.1. Descripción

El propósito del InsecureSystem es implementar un sistema de seguridad simple, en Java, siguiendo las reglas de seguridad de Bell-LaPadula (BLP), seguridad simple, la propiedad * y la tranquilidad fuerte.

El objetivo es que sea un sistema simple no seguro.

Características del sistema:

- Los objetos en este sistema son variables enteras simples.
- Cada objeto tiene un nombre y un valor (inicialmente 0).
- Cada sujeto tiene un nombre y una variable entera TEMP que registra el valor que leyó más recientemente (también inicialmente 0).
- Los sujetos pueden realizar operaciones READ o WRITE en los objetos.
- Para un READ, el sujeto lee el valor actual del objeto y guarda ese valor en su variable TEMP (un READ posterior le pasará por arriba).
- Cuando un sujeto hace un WRITE, el valor del objeto se actualiza.
- La entrada a su sistema es un archivo de comandos.

Instrucciones habilitadas del sistema:

- READ nombre_sujeto nombre_objeto
- WRITE nombre_sujeto nombre_objeto valor

Para las instrucciones con error, se generará la siguiente instrucción especial:

- BadInstruction

Sujetos del sistema:

Sujeto	Nivel de autorización
hal	HIGH
moe	MEDIUM
lyle	LOW

Objetos del sistema:

Objeto	Nivel de sensibilidad
hobj	HIGH
mobj	MEDIUM
lobj	LOW

Ejemplo de entrada:

write lyle lobj 10
write hal hobj 90
como venimos por ahora?
read lyle lobj
read hal lobj
write lyle lobj 20
write hal lobj 200
read hal hobj
write lyle hobj 150
read hal hobj
read lyle lobj
read lyle hobj
write lyle hobj 50
leer lyle lobj
write lyle lucas 70
read hal hobj
write hal lobj
write hal hobj 200
Final de las instrucciones

Ejemplo de salida:

lyle writes value 10 to lobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 0
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 0

hal writes value 90 to hobj
The current state is:
lobj has value: 10

mobj has value: 0
hobj has value: 90
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 0

Bad Instruction
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 0

lyle reads lobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 10

hal reads lobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
hal has recently read: 10
moe has recently read: 0
lyle has recently read: 10

lyle writes value 20 to lobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 90
hal has recently read: 10
moe has recently read: 0
lyle has recently read: 10

hal writes value 200 to lobj
The current state is:
lobj has value: 200
mobj has value: 0

hobj has value: 90
hal has recently read: 10
moe has recently read: 0
lyle has recently read: 10

hal reads hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 90
hal has recently read: 90
moe has recently read: 0
lyle has recently read: 10

lyle writes value 150 to hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 150
hal has recently read: 90
moe has recently read: 0
lyle has recently read: 10

hal reads hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 150
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 10

lyle reads lobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 150
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 200

lyle reads hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 150

hal has recently read: 150
moe has recently read: 0
lyle has recently read: 150

lyle writes value 50 to hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 50
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 150

Bad Instruction
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 50
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 150

lyle writes value 70 to lucas
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 50
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 150

hal reads hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 50
hal has recently read: 50
moe has recently read: 0
lyle has recently read: 150

Bad Instruction
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 50
hal has recently read: 50

moe has recently read: 0
lyle has recently read: 150

hal writes value 200 to hobj
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 200
hal has recently read: 50
moe has recently read: 0
lyle has recently read: 150

Bad Instruction
The current state is:
lobj has value: 200
mobj has value: 0
hobj has value: 200
hal has recently read: 50
moe has recently read: 0
lyle has recently read: 150

2.2. Procedimiento de corrida

En la carpeta: ..\ASSI_Obl\InsecureSystem\inputs

- Colocar el archivo con la lista de instrucciones.

Desde la línea de comandos se debe posicionar en la carpeta: ..\ASSI_Obl\InsecureSystem\

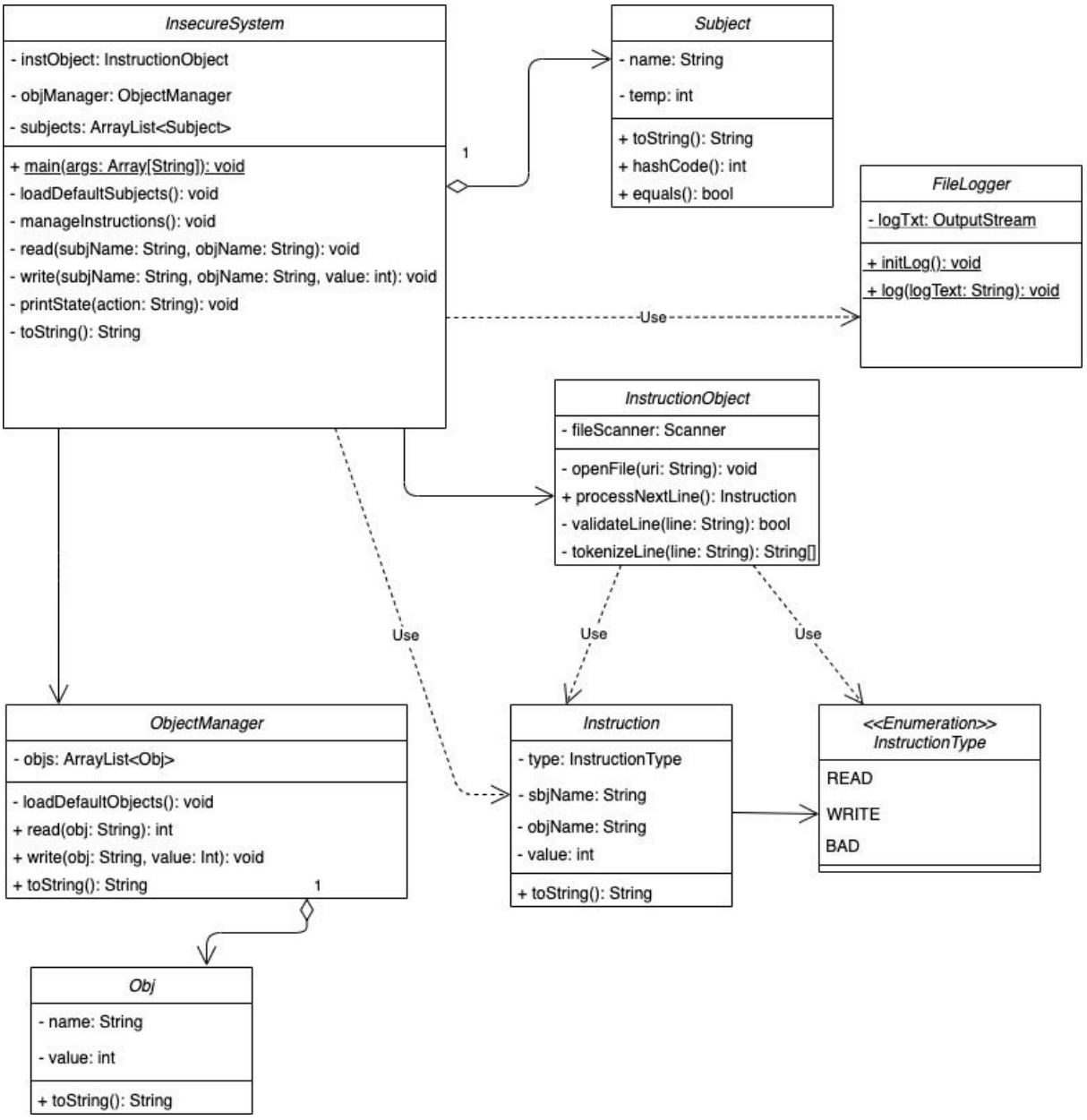
A continuación, se debe ejecutar el siguiente comando poniendo el nombre del archivo (junto con su extensión) al final del comando.

Ejemplo a continuación:
java -jar InsecureSystem.jar InstructionList.txt

En la carpeta: ..\ASSI_Obl\InsecureSystem\outputs

- Se verá el archivo log.txt, el cual mostrará en tiempo real las instrucciones y el estado que va quedando en el procedimiento.

2.3. Arquitectura del sistema



3. SecureSystem

3.1. Descripción

El propósito de SecureSystem es implementar una mejora sobre el sistema de InsecureSystem proveyendo una capa de seguridad asociado a los sujetos y objetos del sistema.

Para lograr esto, se realizarán diversas técnicas sobre los módulos y la lógica del sistema.

Características del sistema:

- Los sujetos tienen asociados niveles de autorización (HIGH, MEDIUM, LOW)
- Los objetos tienen asociados niveles de sensibilidad (HIGH, MEDIUM, LOW)
- Propiedad de tranquilidad fuerte: Las etiquetas no pueden ser cambiadas luego de su creación.
- Se utiliza Bell-LaPadula (BLP) para evaluar la autorización de las acciones de los sujetos sobre los objetos.
- Si la instrucción es sintácticamente incorrecta, genera una instrucción BadInstruction.
- Si el sujeto no puede realizar una lectura sobre un objeto, se guardará 0 en su variable TEMP.
- Se utilizará la variable TEMP del sujeto para determinar el último valor leído del mismo.

Instrucciones habilitadas del sistema:

- READ nombre_sujeto nombre_objeto
- WRITE nombre_sujeto nombre_objeto valor

Para las instrucciones con error, se generará la siguiente instrucción especial:

- BadInstruction

Sujetos del sistema:

Sujeto	Nivel de autorización
hal	HIGH
moe	MEDIUM
lyle	LOW

Objetos del sistema:

Objeto	Nivel de sensibilidad
hobj	HIGH
mobj	MEDIUM
lobj	LOW

Ejemplo de entrada:

write lyle lobj 10
write hal hobj 90
como venimos por ahora?
read lyle lobj
read hal lobj
write lyle lobj 20
write hal lobj 200
read hal hobj
write lyle hobj 150
read hal hobj
read lyle lobj
read lyle hobj
write lyle hobj 50
leer lyle lobj
write lyle lucas 70
read hal hobj
write hal lobj
write hal hobj 200
Final de las instrucciones

Ejemplo de salida:

lyle writes value 10 to lobj
The current state is:
hobj has value: 0
mobj has value: 0
lobj has value: 10
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 0

hal writes value 90 to hobj
The current state is:
hobj has value: 90

mobj has value: 0
lobj has value: 10
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 0

Bad Instruction
The current state is:
hobj has value: 90
mobj has value: 0
lobj has value: 10
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 0

lyle reads lobj
The current state is:
hobj has value: 90
mobj has value: 0
lobj has value: 10
hal has recently read: 0
moe has recently read: 0
lyle has recently read: 10

hal reads lobj
The current state is:
hobj has value: 90
mobj has value: 0
lobj has value: 10
hal has recently read: 10
moe has recently read: 0
lyle has recently read: 10

lyle writes value 20 to lobj
The current state is:
hobj has value: 90
mobj has value: 0
lobj has value: 20
hal has recently read: 10
moe has recently read: 0
lyle has recently read: 10

Bad Instruction
The current state is:
hobj has value: 90
mobj has value: 0

lobj has value: 20
hal has recently read: 10
moe has recently read: 0
lyle has recently read: 10

hal reads hobj
The current state is:
hobj has value: 90
mobj has value: 0
lobj has value: 20
hal has recently read: 90
moe has recently read: 0
lyle has recently read: 10

lyle writes value 150 to hobj
The current state is:
hobj has value: 150
mobj has value: 0
lobj has value: 20
hal has recently read: 90
moe has recently read: 0
lyle has recently read: 10

hal reads hobj
The current state is:
hobj has value: 150
mobj has value: 0
lobj has value: 20
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 10

lyle reads lobj
The current state is:
hobj has value: 150
mobj has value: 0
lobj has value: 20
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 20

lyle reads hobj
The current state is:
hobj has value: 150
mobj has value: 0
lobj has value: 20

hal has recently read: 150
moe has recently read: 0
lyle has recently read: 0

lyle writes value 50 to hobj
The current state is:
hobj has value: 50
mobj has value: 0
lobj has value: 20
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 0

Bad Instruction
The current state is:
hobj has value: 50
mobj has value: 0
lobj has value: 20
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 0

Bad Instruction
The current state is:
hobj has value: 50
mobj has value: 0
lobj has value: 20
hal has recently read: 150
moe has recently read: 0
lyle has recently read: 0

hal reads hobj
The current state is:
hobj has value: 50
mobj has value: 0
lobj has value: 20
hal has recently read: 50
moe has recently read: 0
lyle has recently read: 0

Bad Instruction
The current state is:
hobj has value: 50
mobj has value: 0
lobj has value: 20
hal has recently read: 50

moe has recently read: 0
lyle has recently read: 0

hal writes value 200 to hobj
The current state is:
hobj has value: 200
mobj has value: 0
lobj has value: 20
hal has recently read: 50
moe has recently read: 0
lyle has recently read: 0

Bad Instruction
The current state is:
hobj has value: 200
mobj has value: 0
lobj has value: 20
hal has recently read: 50
moe has recently read: 0
lyle has recently read: 0

3.2. Procedimiento de corrida

En la carpeta: ..\ASSI_Obl\SecureSystem\inputs

- Colocar el archivo con la lista de instrucciones.

Desde la línea de comandos se debe posicionar en la carpeta: ..\ASSI_Obl\SecureSystem\

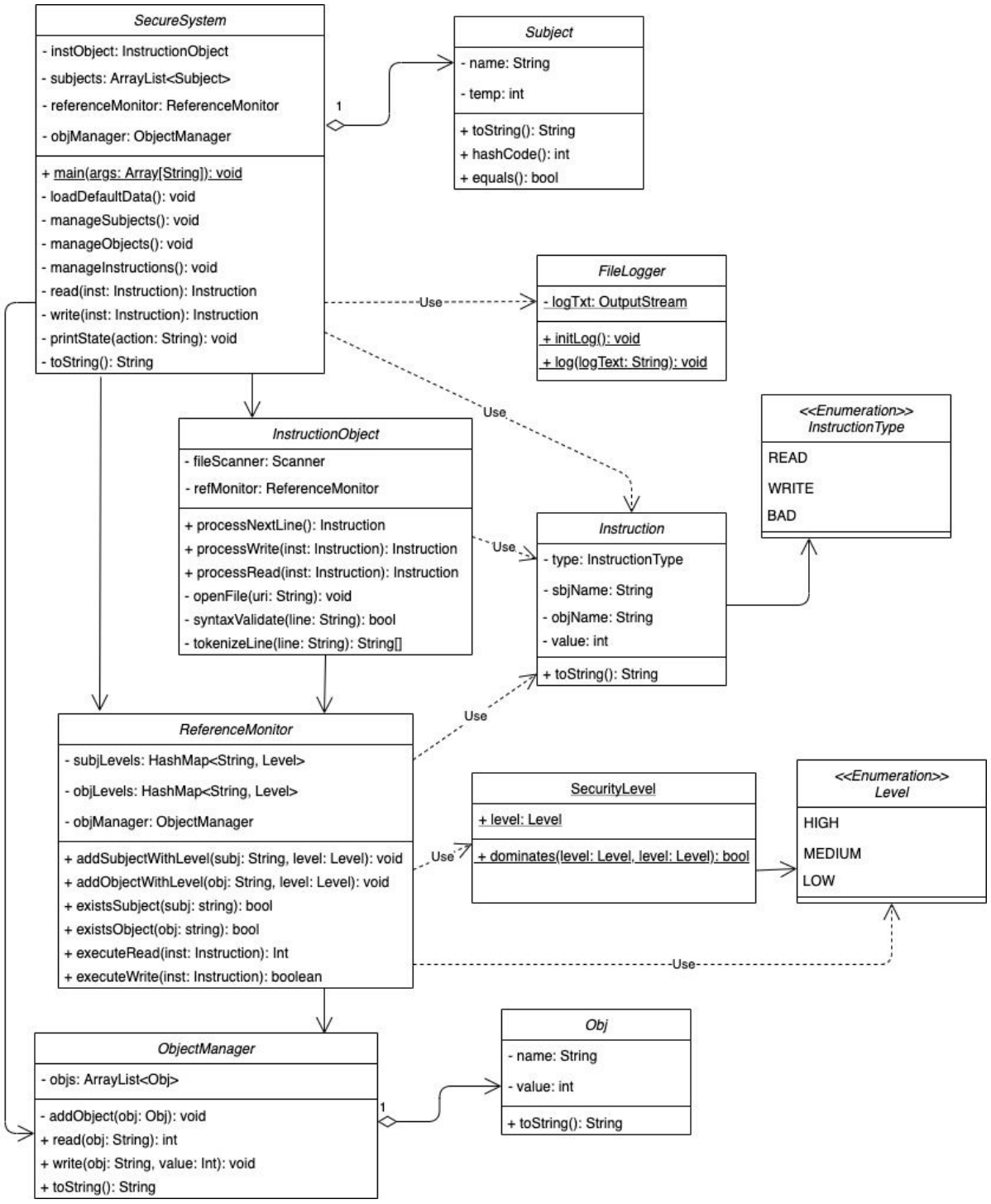
A continuación, se debe ejecutar el siguiente comando poniendo el nombre del archivo (junto con su extensión) al final del comando.

Ejemplo a continuación:
java -jar SecureSystem.jar InstructionList.txt

En la carpeta: ..\ASSI_Obl\SecureSystem\outputs

- Se verá el archivo log.txt, el cual mostrará en tiempo real las instrucciones y el estado que va quedando en el procedimiento.

3.3. Arquitectura del sistema



4. SecureSystemWithCovertChannel

4.1. Descripción

Se actualiza el sistema SecureSystem para que posea un canal encubierto entre el sujeto hal (de nivel de autorización HIGH) a lyle (de nivel de autorización LOW).

Sujetos del sistema:

Nombre	Nivel de autorización
hal	HIGH
moe	MEDIUM
lyle	LOW

Por las características de BLP, no es posible que lyle pueda leer archivos escritos por hal, entonces deben establecer un canal encubierto para poder realizar la transferencia de un mensaje secreto.

Nota: El mensaje secreto deberá estar colocado en la carpeta inputs con su texto correspondiente a ser enviado.

En este programa, se realiza la transferencia del mensaje provisto, a través de un canal encubierto de tiempo, con un sujeto realizando interferencias en el medio (moe) sin que anule o perturbe dicha transmisión encubierta.

Ejemplo de mensaje en la carpeta inputs: Esto es un mensaje secreto.

El archivo de secuencia mostrará la secuencia en la que operarán los sujetos del sistema (hal, moe, lyle), siendo H, M y L las letras correspondientes para hal, moe y lyle consecutivamente.

Ejemplo: HHHLLLLHHLHLLLHLHLHHLHLLLHLHLHLHHLHLHLHLHLL

En la carpeta Outputs se encontrarán archivos de texto con el resultado de la ejecución (en tiempo real) y con el log de instrucciones que se van generando según la secuencia asignada.

Característica del sistema:

- Se agregan las siguientes tres instrucciones
    - CREATE nombre_sujeto nombre_objeto
    - DESTROY nombre_sujeto nombre_objeto
    - RUN nombre_sujeto
- Instrucción CREATE
    - La semántica de CREATE es tal que un nuevo objeto es adicionado al estado con SecurityLevel igual al nivel del sujeto que lo crea. Inicialmente tiene un valor de 0. Si ya existe un objeto con el mismo nombre, en cualquier nivel, la operación no es válida, no será ejecutada
- Instrucción DESTROY
    - Eliminará el objeto indicado del estado, asumiendo que el objeto existe y que el sujeto tiene acceso de WRITE sobre el objeto, de acuerdo a la propiedad * de BLP. Si no, la operación no es válida, no será ejecutada
- Instrucción RUN
    - El objetivo de RUN es permitirle a lyle hacer cualquier procesamiento que sea necesario para obtener el valor del bit que le envían a hal, agregarlo a un byte que está creando, y si el byte está completo, escribirlo en la salida
    - Cada sujeto en la instrucción realizarán acciones diferentes
    - Tiene como objetivo tener un canal encubierto entre los sujetos que operan
        - Para ello, no será posible compartir variables entre los sujetos
    - En la ejecución de RUN, la información se va enviando bit a bit entre los sujetos
    - Para garantizar la existencia de un canal encubierto se provee un archivo de log para registrar las instrucciones realizadas por los sujetos

A continuación se ilustran los siguiente escenarios para tener un seguimiento con el canal encubierto:

Para enviar un bit en 0, hal ejecuta:

RUN hal
CREATE hal obj

Para enviar un bit en 0, hal ejecuta:

RUN hal
CREATE hal obj

lyle recibe el bit (mediante la presencia o ausencia del objeto), mediante la ejecución de:

CREATE lyle obj
WRITE lyle obj 1
READ lyle obj
DESTROY lyle obj
RUN lyle

En el caso de moe, siempre ejecuta lo siguiente:
WRITE moe obj 9
RUN moe

Con la instrucción RUN se puede lograr que lyle registre el bit en su estado interno, agregarlo al byte que crea, y sacar el byte si ha recibido el 8vo bit para ese byte

Ejemplo de entrada:

mensaje.txt:
Este es el texto para el canal encubierto.
Si logran transmitirlo, sin trampas, aprueban.
Suerte.

Ejemplo de salida:

recibido.txt:
Este es el texto para el canal encubierto.
Si logran transmitirlo, sin trampas, aprueban.
Suerte.

log.txt (solo pegamos las primeras líneas por lo extenso de la salida, dentro de la carpeta de entrega dirigiéndose a ..\ASSI_Obl\SecureSystemWithCovertChannel\outputs\log.txt se puede ver una salida completa):

RUN hal
CREATE hal obj
RUN hal
RUN hal
CREATE lyle obj
WRITE lyle obj 1
READ lyle obj
DESTROY lyle obj
RUN lyle
CREATE lyle obj
WRITE lyle obj 1
READ lyle obj
DESTROY lyle obj
RUN lyle
CREATE lyle obj
WRITE lyle obj 1
READ lyle obj
DESTROY lyle obj
RUN lyle
CREATE lyle obj
WRITE lyle obj 1

4.2. Procedimiento de corrida

En la carpeta: ..\ASSI_Obl\SecureSystemWithCovertChannel\inputs

- Colocar el archivo con el mensaje.
- Colocar el archivo con la secuencia.

Desde la línea de comandos se debe posicionar en la carpeta:
..\ASSI_Obl\SecureSystemWithCovertChannel\

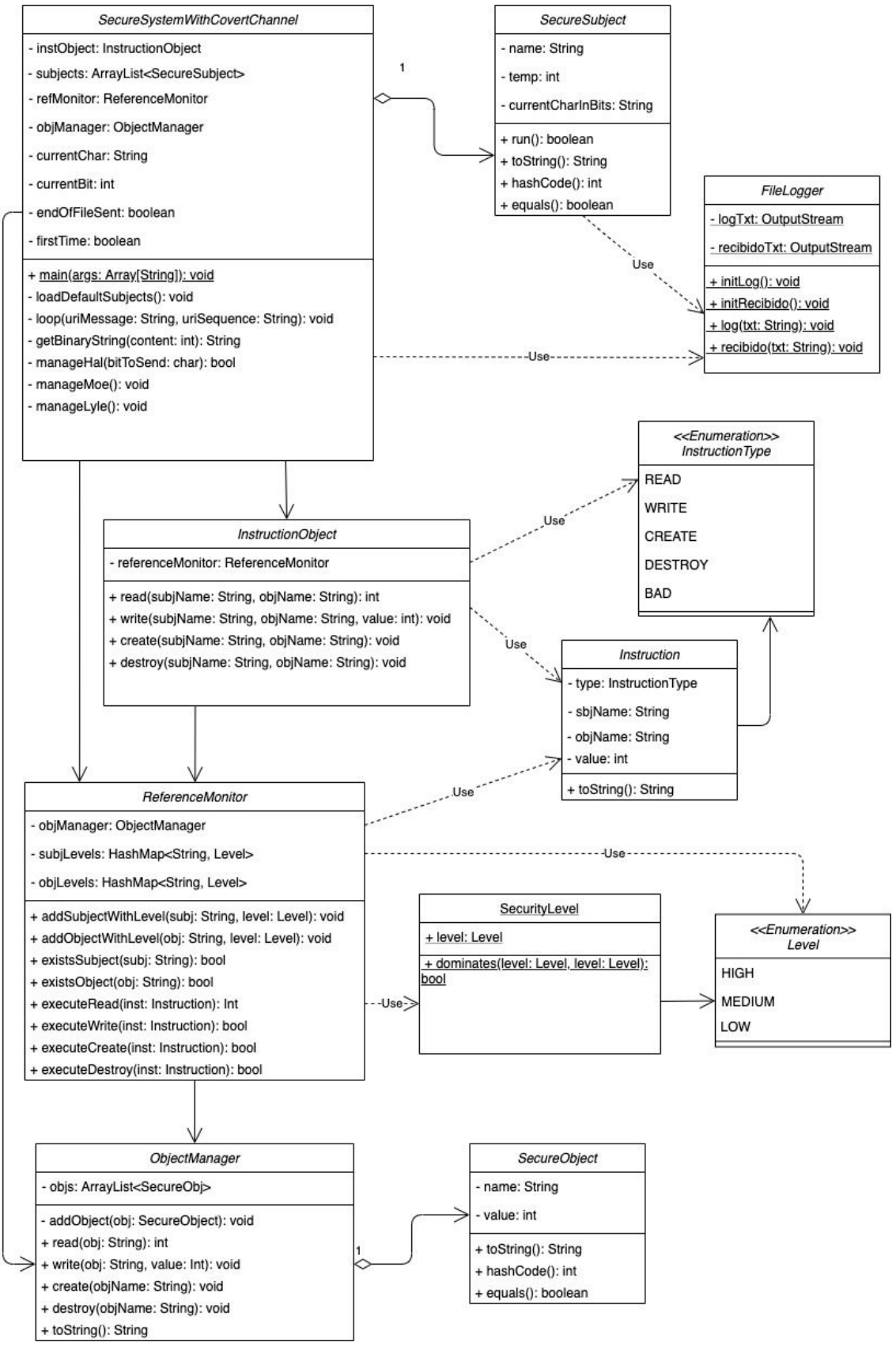
A continuación, se debe ejecutar el siguiente comando poniendo los nombres (junto con sus extensiones) al final del comando.

Ejemplo a continuación:
java -jar SecureSystemWithCovertChannel.jar mensaje.txt secuencia.txt

En la carpeta: ..\ASSI_Obl\SecureSystemWithCovertChannel\outputs

- Se verá el archivo log.txt, el cual mostrará en tiempo real las instrucciones que se van ejecutando en el procedimiento.
- Se verá el archivo recibido.txt, el cual mostrará en tiempo real los caracteres que lyle va recibiendo.

4.3. Arquitectura del sistema



4.4. Protocolo

Conceptualización de alto nivel del protocolo de transmisión de bits por el canal encubierto usando el reloj del sistema:

(el protocolo técnicamente se encuentra en el código de la solución)

Para que cada uno de los dos participantes en el protocolo sepan si es o no su turno para hacer algo, utilizamos el módulo 2 del tiempo de forma tal de obtener siempre uno de entre dos posibles valores.

Procedimiento de lyle:

- (Si el módulo 2 del tiempo es distinto de 1)
    - Deja el tiempo tal que el módulo 2 del tiempo sea igual al que recibió.
- (Si el módulo 2 del tiempo es igual a 1)
    - Se fija lo que leyó en el objeto obj utilizando su propia variable TEMP para determinar lo que le mandó hal.
        - (Si la variable TEMP tiene asignado el valor 0)
            - Significa que hal le mandó un bit en 0, entonces lyle lo agrega al final de su variable currentCharInBits.
                - (Si la variable currentCharInBits llegó a 8 bits)
                - Imprime en recibido.txt el carácter y deja el tiempo tal que el módulo 2 del tiempo sea 0.
                - (Si la variable currentCharInBits no llegó a 8 bits)
                - deja el tiempo tal que el módulo 2 del tiempo sea 0.
        - (Si la variable TEMP tiene asignado el valor 1)
            - Significa que hal le mandó un bit en 1, entonces lyle lo agrega al final de su variable currentCharInBits.
                - (Si la variable currentCharInBits llegó a 8 bits)
                - Imprime en recibido.txt el carácter y deja el tiempo tal que el módulo 2 del tiempo sea 0.
                - (Si la variable currentCharInBits no llegó a 8 bits)
                - deja el tiempo tal que el módulo 2 del tiempo sea 0.

Procedimiento de moe:

- No participa del protocolo.

Procedimiento de hal:

- (Si el módulo 2 del tiempo es distinto de 0)
    - Deja el tiempo tal que el módulo 2 del tiempo sea igual al que recibió.
- (Si el módulo 2 del tiempo es igual a 0)

- (Si tiene que escribir 0)
    - Espera hasta que el módulo 2 del tiempo sea 1 y crea el objeto obj.
- (Si tiene que escribir 1)
    - Espera hasta que el módulo 2 del tiempo sea 1