**Andrés Manrique Ardila 201713672**

**Juan Felipe Parra 201714259**

**{a.manrique, jf.parrac}@uniandes.edu.co**

**Infraestructura Computacional**

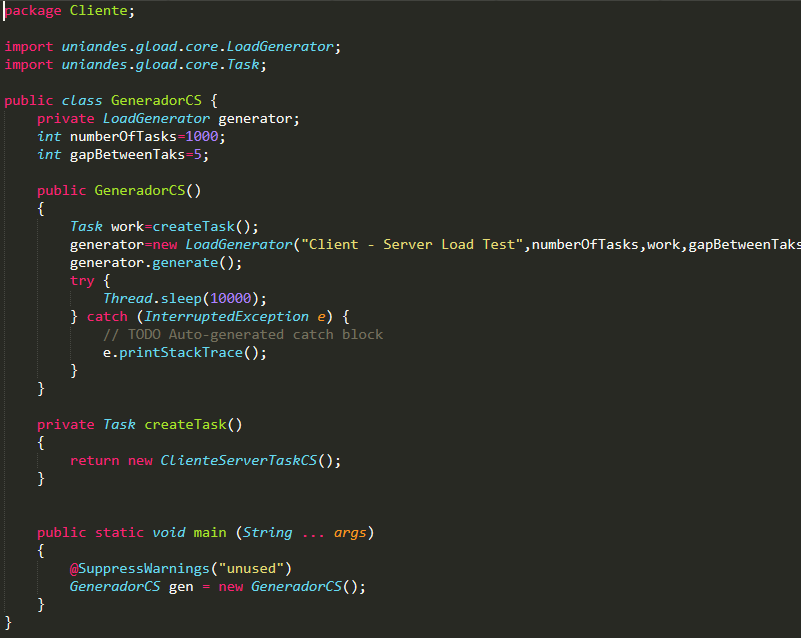
**Universidad de los Andes**

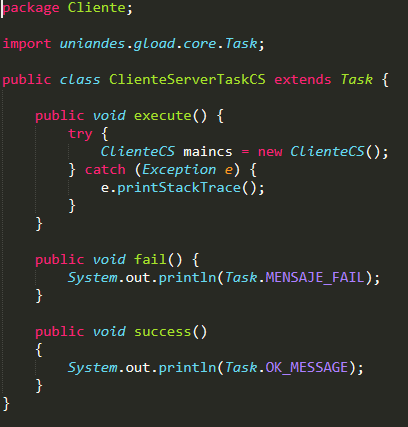
**Caso 3**

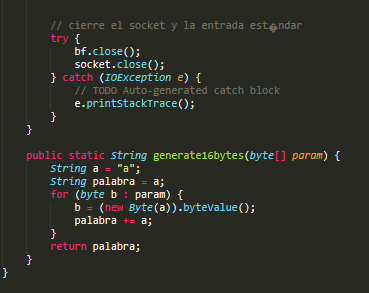
1. Descripción de la implementación

Para la configuración con seguridad:

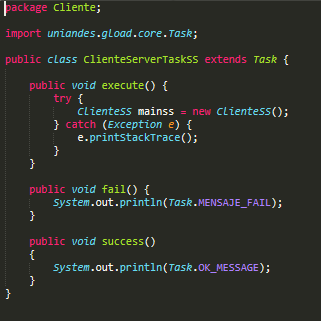
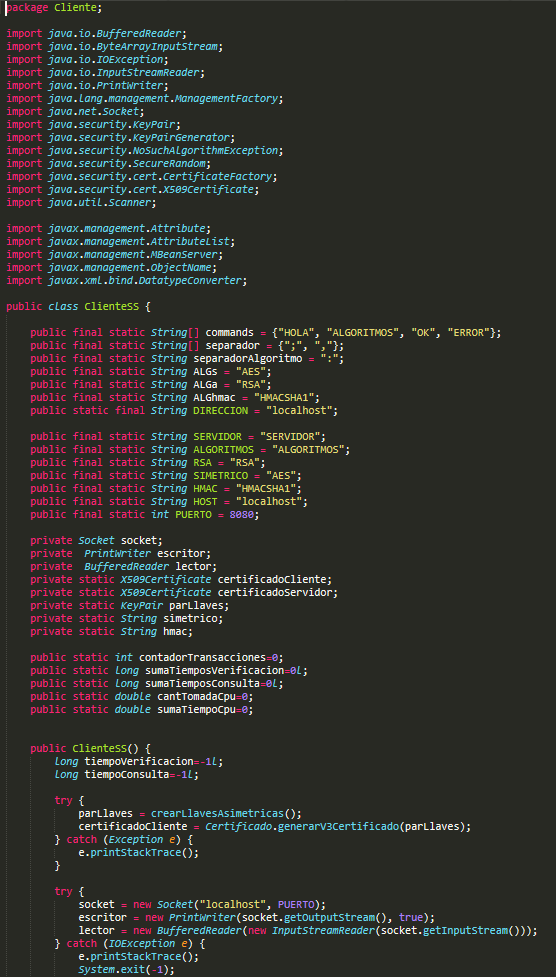
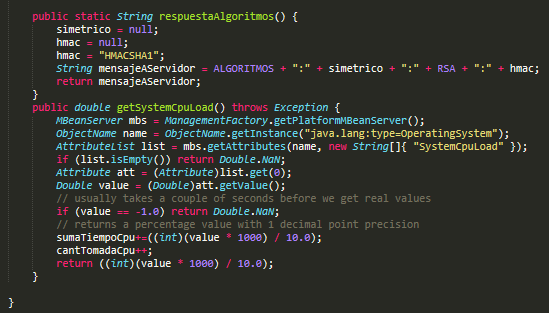
* Se implementó la clase GeneradorCS, la cual se encarga de la generacion de los 1000 task-clientes con seguridad.

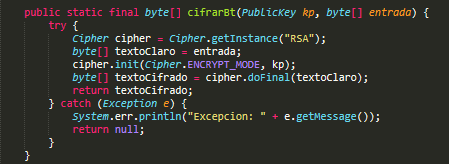


* La clase ClienteServerTaskCS, encargada de la creación de un cliente por task 
* Cambios en el cliente con seguriadad para realizar correctamente el protocolo estipulado en el caso2, ahora de manera automatica(sin interaccion del usuario) 

Para el cliente sin seguridad

* Se creo un generador de 1000 task para llevar a cabo la concurrencia de clientes sin seguridad
* Clase para cada task del cliente sin seguridad 
* Automatización y correcciones menores en el cliente sin seguridad   
* Para el cifrado asimétrico se creo el metodo de cifrar con parametro de bytes y retorno de bytes



1. Identificación de la plataforma

|  |  |
| --- | --- |
|  | |
| Arquitectura | 64 bits |
| # de núcleos | 4 |
| Vel del procesador | 2.6 GHz |
| RAM | 16 |
| Espacio JVM |  |

1. Comportamiento de la aplicación con diferentes estructuras de administración de la concurrencia
   1. Resultados carga 400 peticiones
   2. Resultados carga 200 peticiones
   3. Resultados carga 80 peticiones
   4. Conclusion
      1. Para el uso de la cpu se nota un aumento en los 3 escenarios de carga de un aumento de aproximadamente 50 % en el caso de 2 threads respecto a 1 thread.
      2. El comportamiento denotado para el uso de cpu se ve reflejado en igual proporción para el tiempo promedio de una transacción para los 3 escenarios.
      3. Como resultado de este análisis, es posible concluir que la mejor manera de configurar un servidor concurrente con seguridad es de la manera monothread.
2. Comportamiento de la aplicación ante diferentes niveles de seguridad
   1. Se espera que el porcentaje de cpu utilizado sea menor en una aplicación sin seguridad, pero a su vez tenga un tiempo promedio de transacción más alto.
   2. En promedio la aplicación con seguridad consumió 1% mas de cpu que la aplicación sin seguridad. Lo cual es una diferencia muy diminuta con un gran beneficio para la integridad de la información.