Parcial 1

Luisa María Vivas

1. SINCRONIZACIÓN DE PROCESOS

**A.** En el siguiente gráfico podemos ilustrar el problema del caso, donde existen dos tipos de carros (Carro Izquierdo, Carro Derecho) y por el puente solo se admite que pasen carros en un sentido y con una capacidad máxima de 10 coches simultáneamente.



B.

Pedir Recurso: Para nuestro problema sucede que el puente solo admite autos en un sentido por tanto cuando se pide el recurso es definir qué tipo de carro requiere pasar por el puente, es decir que requiere de este recurso. Cuando esto sucede el otro carro debe esperar su turno.

Devolver recursos: En el momento que el peso de coches es superado entonces se debe devolver el recurso y de esta manera cuando termine de pasar los carros Izquierdo entregar los recursos para los otros coches de derecha.

Las estructuras planteadas maximizan el uso de recursos, lógicamente y computacionalmente, debido a que si este método lo hubiéramos realizando con otro método de sincronización visto en clase como son la espera activa seguramente consumirían más recursos innecesariamente debido a que operaciones como While(condición) aumentan la complejidad algorítmicamente y está utilizando más recursos. A diferencia de la espera activa, los semáforos se encargan de maximizar los recursos disponibles, esto quiere decir que verifican el tiempo que toma cada hilo en ejecutarse dentro del procesador, de ahí que si un hilo del proceso toma demasiado tiempo, detiene el proceso para que otros procesos puedan usar el recursos de tal forma que siempre se esté utilizando hasta que deja pasar nuevamente el hilo detenido.

Para evitar la inanición, debemos comprender se debe realizar una serie de bloqueos según los temas de semáforos, es decir cuando un tipo de carro está pidiendo un recurso e inicialmente este está disponible, lo que se hace que es permite que se coloque los 10 autos y en ese mismo instante que sucede esto se debe bloquear el otro tipo de carro, cuando esto sucede debe esperar para luego en el momento que el recursos esté disponible quien pase a hacer uso sea el otro tipo de carro.

Para el desarrollo de la solución se definen las siguientes variables:

* Int Cantidad: esta variable se encarga de definir la cantidad de autos que pueden pasar por el puente, en este caso 10 es el máximo.
* Int numCochesDer: lleva el registro del número de carros derecha que pasan por un determinado t por el puente.
* Int numCochesIzqu: lleva el registro del número de carros izquierda que pasan por un determinado t por el puente.
* semaforoDer: se encarga de que los carros pasen en el sentido derecho y de esta manera evitar la inanición del paso de los autos en este sentido.
* semaforoIzqu: se encarga de que los carros pasen en el sentido izquierdo y de esta manera evitar la inanición del paso de los autos en este sentido.
* mutexD: este semáforo mutexD es utilizado para sincronizar la utilización de la variable semaforoDer.
* mutexI: este semáforo mutexI es utilizado para sincronizar la utilización de la variable semaforoIzqu.
* puente: este semáforo representa el puente disponible que permite que los autos solo pasen en un sentido.
* muPuente: variable que es utilizada para sincronizar la utilización de la variable que representa el puente y cuantos coches puede cruzarlo.

C. Implementación solución:

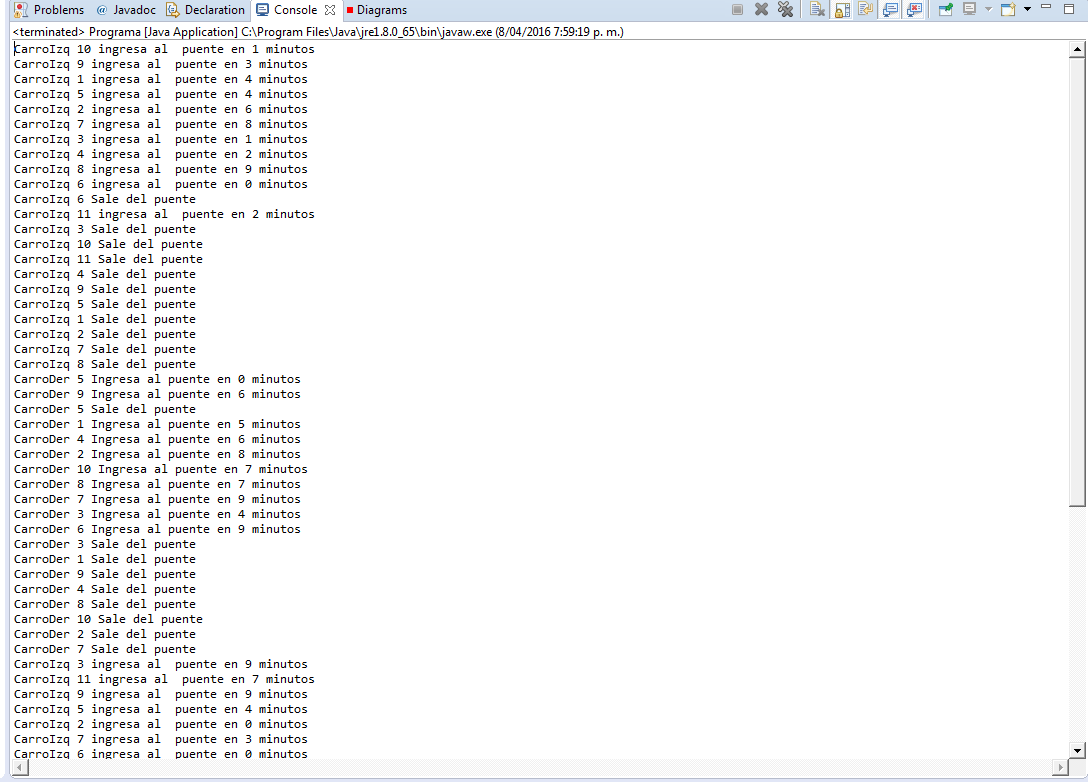
La implementación y desarrollo se llevó a cabo utilizando 4 clases:

Puente: en esta clase están definidas todas las variables de los semáforos y métodos utilizados para el paso de los carros del puente.

En esta clase se definieron 4 métodos llamados turnoDer(), turnoIzqu(), finIzqu(), finDer(). Respectivamente los métodos se encargan de que cada tipo de auto pase buscando habilitar y bloquear el paso según los recursos necesitados.

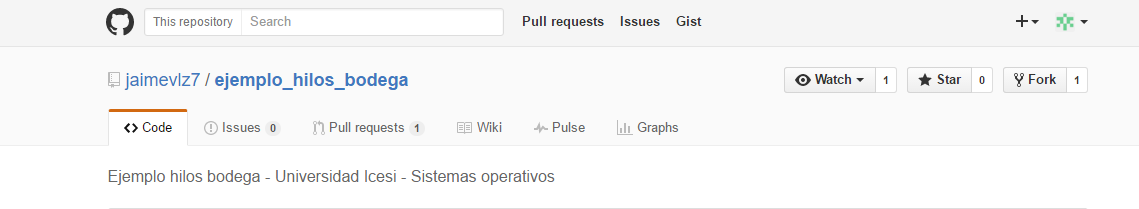
CochezIzq y CochesDer son clases que contienen los hilos y representan un coche individual.

Programa: Realiza las pruebas del ejercicio, en esta clase el caso de prueba se inicializan 12 autos para habilitar el paso en una dirección, entonces lo que hace el método tunoDer() se encarga de que ingresen los autos de la derecha y cuando pasan los 10 autos , los otros 2 quedarían en espera hasta que algún carro haya bajado y se bloquea el tipo de carro izquierda hasta que un auto al menos desocupe el recurso. También se debe ver en la prueba que solo ingresan un tipo de autos pueden ir en una dirección y no verse carros mezclados, es decir subir carro izquierda y subir carro derecha para que existe una sincronización del ingreso al puente como se puede ver a continuación.

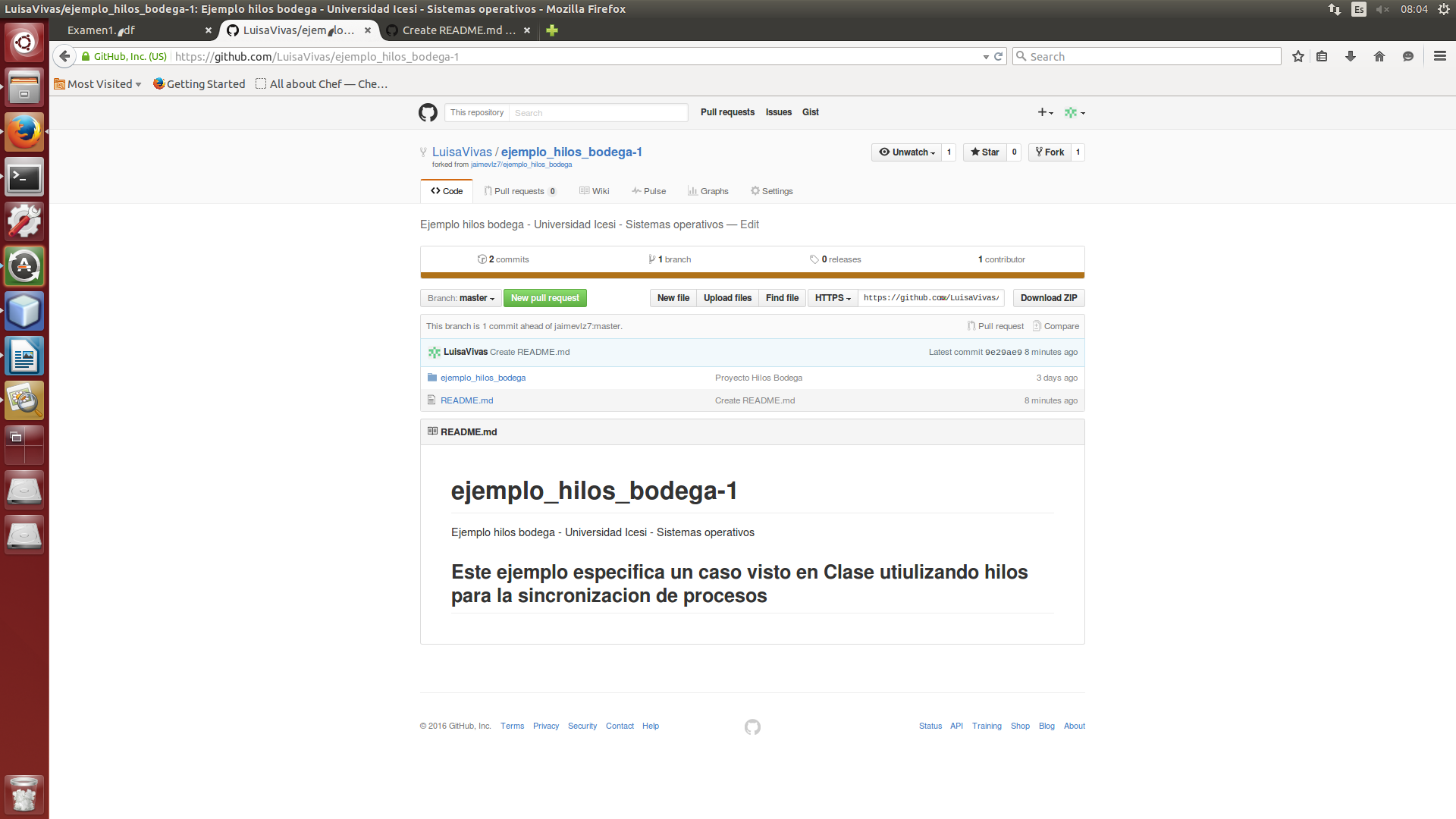


**3.**

Realice un fork a un repositorio de mi compañero Jaime Vélez llamado ejemplo\_hilos\_bodega



y modifique su README.md puesto que no tenía descripción del ejercicio subido por él.



Seguido a esto realice un comentario con las adiciones propuestas por mí de la siguiente manera.

Cree un nuevo pull request donde y en el campo de texto le indico que agregue del caso la especificación correcta para la compresión de quien está leyendo.

