Informe Caso 1: Manejo de Concurrencia

ISIS2203 – Sección 03

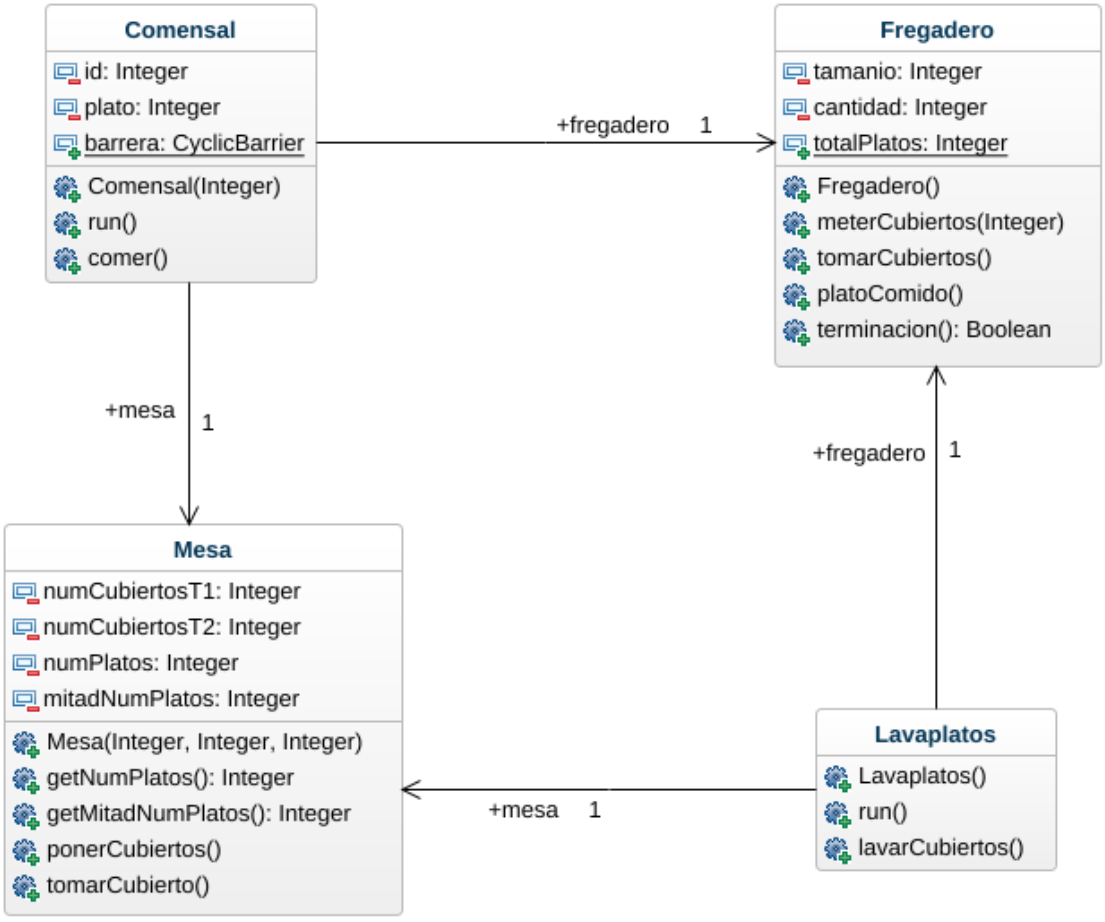
Grupo 03

Carlos Eduardo Ramírez Martínez - 201921729

Néstor Felipe González García – 201912670

Funcionamiento de la aplicación.

La aplicación, al comenzar su ejecución, lee e inicializa los datos almacenados en el archivo properties. Con estos valores, instancia los threads (comensales y lavaplatos) y los monitores (mesa y fregadero). A continuación, se muestra el modelo conceptual del caso:



El protocolo general que sigue la aplicación es:

1. El comensal toma un par de cubiertos de la mesa.
2. El comensal come un plato.
3. El comensal deja el par de cubiertos que usó en el fregadero.

4. El lavaplatos toma los cubiertos toma un par de cubiertos del fregadero
5. El lavaplatos lava un par de cubiertos.
6. El lavaplatos devuelve los cubiertos a la mesa.

Cabe aclarar que los threads de los comensales y del lavaplatos se están ejecutado concurrentemente.

De los atributos de la clase Comensal, es importante aclarar que los atributos que modelan las relaciones, además de ser públicos, son estáticos, esto para poder asignarles los valores leídos en el archivo properties. Al ser ejecutado un thread comensal este toma los cubiertos, come, deja los cubiertos y, en caso de haber comido la mitad de los platos, espera a que los demás comensales lleguen a este punto.

Para tomar los cubiertos el comensal hace uso del monitor Mesa para sincronizarse con otros comensales. En este método, intenta tomar los cubiertos en orden, es decir, primero un cubierto T1 y luego un cubierto T2, y en caso de no poder tomar alguno de los cubiertos debe devolver el cubierto tomado y esperar mediante un wait hasta que sea notificado de que el lavaplatos ha dejado cubiertos en la mesa y pueda volver a intentar tomar los cubiertos. Además, al volver a intentar tomar los cubiertos luego de despertarse del wait, llama al método tomarCubiertos recursivamente.

Luego al comer genera aleatoriamente el tiempo que le toma comer entre 3 y 5 segundos y se duerme mediante sleep durante ese tiempo generado. Después de esto, llama a un método sincronizado en el Fregadero para disminuir en 1 la cantidad de platos totales que faltan por comerse para poder indicar el fin del thread del Lavaplatos más adelante.

Luego de comer, intenta meter los cubiertos al fregadero mediante un método no sincronizado que revisa, mediante un ciclo, cuando hay disponibilidad en el fregadero para dejar los cubiertos y en caso de que no haya disponibilidad, hace una espera semiactiva con yield. Cuando finalmente hay disponibilidad, hay un bloque sincronizado dentro del método sobre el Fregadero donde aumenta en 1 la cantidad de parejas de cubiertos en el fregadero.

Finalmente, revisa si se encuentra en la mitad de los platos y si es el caso, emplea la barrera para hacer await y esperar a que todos los threads Comensal lleguen al mismo punto para poder avanzar.

Todo este proceso se repite hasta que cada comensal haya comido todos sus platos.

Por otro lado, el thread Lavaplatos al ejecutarse corre un ciclo infinito (condición true) el cual ejecuta una serie de funciones. Primero, el Lavaplatos intenta tomar un par de cubiertos del lavaplatos, si el fregadero no tiene cubiertos para lavar, el lavaplatos realiza una espera activa para volver a intentar tomar cubiertos.

Seguidamente, el Lavaplatos lava los cubiertos tomados, proceso el cual tarda un tiempo aleatorio entre 1 y 2 segundos. Finalmente, este devuelve los cubiertos a la mesa, función la cual está sincronizada en el monitor Mesa, este método nunca implica una espera, ya que la mesa siempre tiene espacio para nuevos cubiertos. De acuerdo con lo anterior, el Fregadero se comporta como un buffer para regular la entrada de cubiertos al lavaplatos, y la Mesa es un monitor que controla el acceso a los cubiertos.