Tecnológico de Monterrey

Bulls 96

Oscar Fernandez Moreno A07013362
Jorge Manzo A01633991
Linda Nayeli Abundis López A01636416

Sección de trabajo individual

Oscar Fernandez Moreno

Cree el repositorio inicial, manejé los commits y pull requests de todo lo que se subía, además que resolví todos los merge conflicts que se iban presentando a lo largo del proyecto.

Realicé la parte de manejo de distintos consumidores y productores sobre un buffer así como la lógica de los hilos para que fuera fácil detenerlos en un futuro, además de hacer el funcionamiento de la lógica para las entradas:

- Número productores (1 - 10).
- Número consumidores (1 - 10).
- Tiempo de espera (0 ms a 10,000 ms) para productor/consumidor antes de su siguiente ciclo.
- Tamaño del buffer (1 - 100).

Dentro de la lógica para hacer la parte de los hilos implementé una manera de guardar la información para hacer las:

- Lista de tareas realizadas (operación, resultado e identificador del consumidor)
- Lista de tareas por hacer (operación e identificador del productor)

Además de hacer la parte lógica hice el despliegue de los datos en la GUI de manera ordenada de acuerdo a como fueron llegando las peticiones de consumir o producir en donde se actualizan de acuerdo a como se hagan dichas llamadas.

Conecté la parte de la GUI con lo que se hizo de Scheme, además de integrar todas las partes para la entrega final.

Jorge Manzo

Mi aportación principal fue la implementación de Scheme al programa, aunque tuvo ciertos inconvenientes en el proceso, y el manejo de la generación de operaciones y sus respectivas soluciones.

También implementé las funciones para generar las operaciones aleatorias a resolver y resolverlas tanto usando Racket/Scheme como manualmente

- Racket/Scheme: Enviar el String de la operación a Racket usando el Buffer Writer y leer el resultado usando el Buffer Reader. Al tener la respuesta en Un String (i.e >5 524), limpiar el String para poder convertirlo a Double de ahí poderse enviar a la tabla.
- Manual: Leer caracter por caracter, dividir el string inicial, identificar el operador (+,-,/,*) y aplicar la operación a los operandos

Agregué la clase de Scheme.java para poder utilizar las funciones de manera más óptima

Linda Nayeli Abundis López

En la parte de actuadores, trabajé en que el botón de inicio validara las entradas, creara los hilos (productores/consumidores) e iniciara la ejecución.

En el botón de paro trabajé en que detuviera/matara los hilos.

Trabajé en la validación de todos los inputs

- Número productores (1 - 10).
- Número consumidores (1 - 10).
- Tiempo de espera (0 ms a 10,000 ms) para productor/consumidor antes de su siguiente ciclo.
- Tamaño del buffer (1 - 100).
- Rango de valores para las operaciones en scheme: [0, 9].
- Operadores para las operaciones en scheme: +, -, *, /.

y en no permitir que el usuario ingrese algún carácter diferente a un número.

Trabajé en la conexión de los inputs con las clases.

En la parte de “Número de tareas por hacer”, trabajé en la realización de la barra de progreso y el porcentaje.

En la parte de “Número de tareas realizadas”, trabajé en la sección de número de tareas realizadas (cantidad).

Revisé algunos commits y ayudé en la solución de errores de mis compañeros.

Puntos Extra

Pasos para probar el Programa (Windows)

Requerimientos :

- Racket

Pasos Para modificar el programa (para que funcione correctamente)

1. Descargar el proyecto por Github
2. Moverse a la rama de Racket

3. Abrir la clase Scheme.java
4. En la línea #24, modificar el Path del File y el Path del ejecutable al lugar de instalación correspondiente de Racket
5. Correr el programa de GUI

Notas Importantes : Se recomienda que al realizar la instalación de Racket en Windows no se cambie la dirección de instalación, de esta manera no habrá que realizar ninguna modificación al código del programa