Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

IC2101 - Programación Orientada a Objetos | Grupo 20

Proyecto #1 del curso:
Simpletron

Estudiantes:
Valery Mishel Carvajal Oreamuno | 2022314299
Anthony Josué Rojas Fuentes | 2018027141

Profesor:
Carlos Martín Flores Gonzalez

Jueves 06 de Octubre
II semestre 2022

I. Introducción.

El Simpletron es una máquina simple que ejecuta programas escritos en lenguaje máquina de Simpletron (LMS). El objetivo de este proyecto es buscar implementar esta máquina, la cual podemos ver como un simulador de computadora, que ejecutará las instrucciones dadas cumpliendo ciertos requisitos expuestos en el documento de instrucciones recibido para completar el trabajo.

Para explicar brevemente la funcionalidad que esperamos de este simulador, vamos a tener un acumulador que nos ayuda guardando información o, mejor dicho, datos utilizados para cálculos y demás. Por otro lado, el programa se va a manejar en 1000 “palabras”; cada palabra siendo un número de cinco dígitos en notación decimal con un signo “+” o “-” y se va a hacer referencia a ellas según su número de ubicación, dicho número se encuentra entre 0 y 999.

También, el programa debe ser cargado en memoria antes de ser ejecutado y su primera instrucción (donde inicia) siempre será colocada en 0. Cada ubicación en la memoria del Simpletron va a ser: el signo, los primeros dos dígitos del número serán la instrucción y los últimos tres son los operandos.

Por último, los códigos de operación de LMS fueron tomados del escrito que contenía las instrucciones del proyecto, mientras que el funcionamiento del programa se basó en el documento mencionado y las recomendaciones del profesor. En resumen, nuestra meta con este programa es que logre leer datos numéricos, calcular lo que se le indica e imprimir los resultados deseados en respecto a las instrucciones recibidas del usuario.

II. Estrategia de la solución

Como estrategia de la solución decidimos que se iban a implementar un total de cinco clases, siendo estas las siguientes:

1. Instruction: En esta clase se encuentran ciertos métodos necesarios en el programa que transforman variables a int o string según sea requerido.

2. IOUtils: Esta clase lee las entradas y se encarga de hacer todas las verificaciones necesarias, retornando los mensajes de error correspondientes.

3. Memory: En la clase Memory definimos todos los getters y setters necesarios para el programa que diseñamos.

4. Operation: Aquí, se implementaron todas las operaciones (matemáticas y otras) requeridas en el programa según las instrucciones del proyecto.

5. Simpletron: Esta actuó como nuestra clase principal, ya que contiene el main y permite que nuestro programa funcione correctamente según lo solicitado.

Con las clases ya definidas, analizamos el programa para desarrollarlo de modo que cumpla con las expectativas y trabajar cada clase de la manera más óptima posible. Aparte, establecimos los métodos de cada clase y su función.

III. Detalles de implementación

Teniendo un esquema general de la estructura de nuestro proyecto, pasamos al código; creamos las clases expuestas anteriormente e iniciamos a trabajarlas poco a poco. A continuación se explica a detalle el funcionamiento de cada clase.

1. Instruction: En esta clase podemos encontrar variados métodos, tales como: *operandToInstruction()*, este recibe un entero que va de -999 a 999 llamado "operand", lo transforma en String y lo devuelve con el formato correcto para guardarlo en memory.

   El siguiente es *instructionToOperand()* que recibe un String correspondiente a una de las instrucciones asignadas en la memoria de Simpletron llamado "instruction", después, toma los últimos 3 dígitos de la instrucción y retorna el operando en formato int. Este puede ser positivo o negativo, dependiendo del signo al inicio de la instrucción, y siempre debe tener un mínimo de 3 dígitos.

   Luego, tenemos a *instructionToOperationCode():*, la cual recibe un String que es una de las instrucciones asignadas en la memoria de Simpletron llamado "instruction", toma los dos primeros dígitos de la instrucción después del signo, que corresponde al operationCode, y lo devuelve como int positivo de dos números.

   Por último, *instructionToSign():* recibe un String correspondiente a una de las instrucciones asignadas en la memoria de Simpletron llamado "instrucción", luego, toma el primer carácter (el signo) y devuelve un booleano True si el signo es +, o False si el signo es -.

2. IOUtils: En esta clase, encontramos todas las validaciones de las entradas en dos métodos. Más específicamente, de la siguiente manera:
   - El método *readInput()* hace las siguientes validaciones: Si el input es nulo, si no es de longitud 6 o si el signo es diferente de "+" o "-" muestra el mensaje "Error: El código de operación no es válido.". Si el signo es "-", revisa si los dos primeros dígitos son diferentes a "00" y el número no es "99999"; de darse ese caso, envía el mismo mensaje de error.
   - Por último, el método *readOperand()* revisa si la entrada está entre -999 y 999; si es mayor o menos que ese rango, envía el siguiente mensaje: "Error: Desbordamiento de memoria".

3. Memory: Se crea un String (memorylist) y tenemos los siguientes métodos:
   - *getMemory()*: Obtiene "pos" que es una instrucción de la posición de memoria y retorna el memorylist[pos].
   - *getMemorySign()*: Nos da el signo de una instrucción en una posición de memoria indicada en la entrada "pos".
   - *getMemoryOperationCode()*: Retorna el operationCode de "pos" (una posición de memoria).
   - *getMemoryOperand()*: Devuelve un operando de la posición de memoria que obtiene como parámetro.

- *setMemory()*: Nos da el número (operando) en la posición de memoria asignada.
- *setMemoryInstruction()*: Devuelve la instrucción en la entrada "pos".
- *memoryDump()*: Imprime todo lo guardado en memoria hasta el momento de forma descendente. Este método es utilizado para probar el programa durante el proceso de elaboración del mismo.

4. Operation: Como se mencionó anteriormente, en esta clase se muestran todas las operaciones necesarias en el programa. Todas estas se explican a continuación:
    - *executeOperation()*: Recibe un objeto "memoria" de tipo Memory y ejecuta todas las operaciones en esa memoria. Dentro de este método tenemos que mientras se esté ejecutando el programa se va a cumplir todo lo siguiente:
        - Si el acumulador es mayor a 999 se manda un mensaje de error y se cierra el simpletron (se deja de ejecutar).
        - Si el dato introducido es "-99999" se detiene el programa.
        - En el caso que ninguna de las anteriores suceda, se utiliza un condicional de selección llamada switch que trabaja con casos. Estos casos son los siguientes:
            - Caso 0: Nos dice que es un número que se quiere guardar, no una instrucción de operación.
            - Caso 10: Lee un número de máximo 3 dígitos y lo guarda en la dirección de memoria que indica el operando.
            - Caso 11: Imprime el número que está en el espacio de memoria indicado por el operando.
            - Caso 20: Agrega al acumulador el número indicado por el operando.
            - Caso 21: Guarda el número del acumulador en la dirección de memoria indicada.
            - Caso 30: Agrega al acumulador el valor de la dirección de memoria asignada con el operando.
            - Caso 31: Resta el valor indicado al acumulador.
            - Caso 32: Multiplica el acumulador por el valor asignado.
            - Caso 33: Divide el acumulador entre el valor del operando, haciendo la validación de la división entre 0 e indicando el error de ser necesario.
            - Caso 34: Calcula el acumulador módulo el número del operando.
            - Caso 35: Eleva el acumulador al número indicado.
            - Caso 40: Salta al espacio de memoria indicado.
            - Caso 41: Salta al espacio de memoria indicado solo si el número es negativo.
            - Caso 42: Salta al espacio de memoria indicado solo si el número es 0.

- Caso 43: Salta al espacio de memoria indicado solo si el número es positivo.
- Caso 44: Termina la ejecución del programa.
- Caso 45: Resetea el acumulador (lo pone en 0).
- Caso 50: Salta al espacio de memoria indicado y puede ser usado para crear un ciclo después de inicializar el cycle controller.
- Caso 51: Pone el cycle controller en el número indicado por el operando.

Si la entrada es inválida, manda el siguiente mensaje de error: “Error: El código de operación no es válido”.

5. Simpletron: Esta clase contiene la función main() que es el método principal del programa y llama a todas las otras clases y objetos. En general, ejecuta el programa Simpletron.

IV. Restricciones o suposiciones

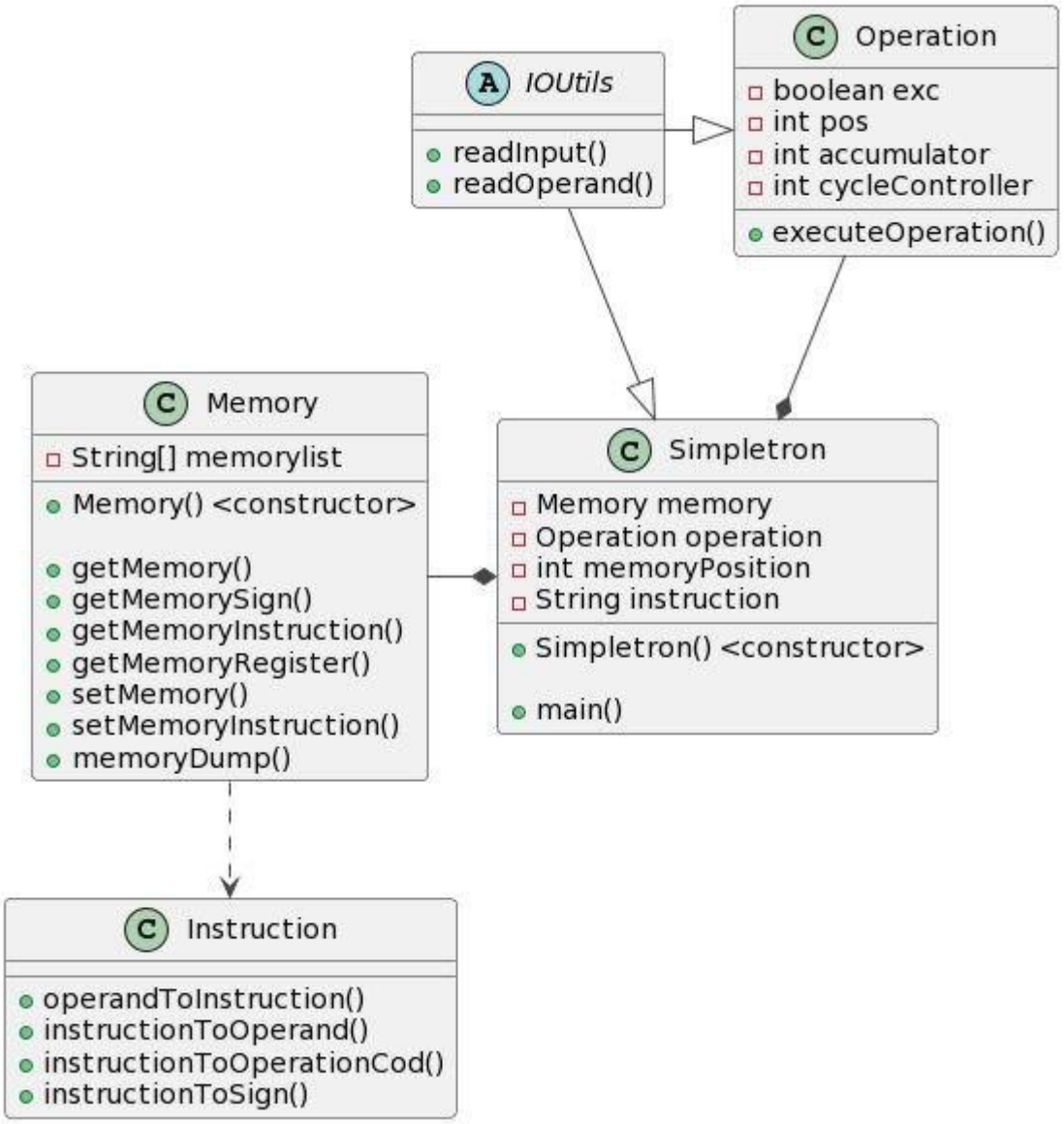
Durante el proceso, nos encontramos con algunas restricciones en el programa, lo que nos llevó a implementar diversas validaciones para que funcione de manera óptima. Entre estas tenemos: la división entre 0, el tipo de dato (int, string…) según sea necesario en la situación y el caso en que se ingresara una instrucción inválida o un dato que el programa no permite (por ejemplo: una letra).

Se tomó la suposición de que era necesario permitir que si el usuario lo programa como tal con los comandos de simpletron, le sea posible crear un ciclo sin salida, de forma semejante a como se puede hacer en la mayoría de los lenguajes de programación con un ciclo while(true).

V. Problemas encontrados

Al igual que en cualquier otro trabajo, nos encontramos con problemas en el camino. Hablando de este proyecto en específico, existieron bastantes momentos en que el código nos daba un error o no funcionaba de la manera que esperábamos. Incluso, hubo momentos en los cuales pensamos que el código de una operación estaba bien y tuvimos problemas en comprender cómo solucionarlo. Sin embargo, logramos solucionar todas estas situaciones analizando el código y modificándolo con el objetivo de mejorarlo y, por ende, eliminar cualquier problema presente en él.

VI. Diagramas de clases



VII. Conclusiones y recomendaciones

Personalmente, opinamos que este trabajo fue un buen ejercicio para reforzar los conocimientos adquiridos mediante el curso e, incluso, para comprender conceptos (o implementaciones nuevas de ellos) de una manera entretenida y diferente. Aparte, con los ejemplos dados y la explicación de los profesores, logramos tener una mejor perspectiva sobre el objetivo del proyecto y, con esto, tener presente su importancia e impacto en nuestro desarrollo como ingenieros.

De la misma manera, el trabajo nos permitió practicar nuestras habilidades de programación y a su vez mejorarlas. Además, nos ayudó a comprender de manera clara cómo utilizar java, qué características nos ofrece y, en general, el modo de sacar el mejor provecho de la herramienta para adaptarla a nuestras necesidades.

Bibliografía

[1]
Java® Platform, Standard Edition & Java Development Kit Version 11 API Specification. [Online]. Available: https://docs.oracle.com/en/java/javase/11/docs/api/index.html.

[2]
Oracle, *JDK 11 Documentation*. [Online]. Available: https://docs.oracle.com/en/java/javase/11/.

[3]
M. Flores, ‘Proyectos_Simpletron.’ 2022.

[4]
G. De Referencia and D. L. Plantuml, ‘Diagramando UML con PlantUML’. [Online]. Available: https://plantuml.com/es/guide.