UNIVERSIDAD DEL VALLE DE GUATEMALA
CC 2016 -Estructura de Datos

Sección 10

Ing.Michaelle Alexander Pérez Riz

Hoja de Trabajo #4

June Herrera - 231038
Jonathan Zacarías - 231104

Guatemala, 25 de febrero de 2024

Análisis:

- Requerimiento
    - ¿Qué acciones debe poder hacer su programa?
        - Leer una expresión infix desde un archivo.
        - Convertir la expresión infix a postfix utilizando el algoritmo Shunting Yard.
        - Evaluar la expresión postfix utilizando una calculadora.
        - Permitir la elección de la implementación de la pila (stack) y la lista utilizada en el programa.
        - Mostrar el resultado de la evaluación de la expresión infix.
    - ¿Con qué datos va a trabajar?
        - Expresiones matemáticas en formato infix, almacenadas en un archivo de texto.
        - Pilas (stacks) implementadas con ArrayList, Vector, o List.
        - Listas implementadas con SinglyLinkedList o DoublyLinkedList.
        - Operandos y operadores de expresiones matemáticas.
    - Descripción de clases:Métodos y Atributos de clases:
      ArrayListStack<T>:

      Métodos:
      push(T item): Agrega un elemento al stack.
      pop(): Retira y devuelve el elemento superior del stack.
      peek(): Devuelve el elemento superior del stack sin retirarlo.
      isEmpty(): Verifica si el stack está vacío.
      size(): Devuelve el tamaño del stack.
      Calculator:

      Métodos:
      evaluateExpression(String postfixExpression, UVGStack<Integer> stack, List<Integer> list): Evalúa una expresión postfix.
      performOperation(int operand1, int operand2, char operator): Realiza una operación matemática.
      CalculatorProgram:

      Métodos:
      main(String[] args): Punto de entrada del programa.
      readInfixExpressionFromFile(String filename): Lee una expresión infix desde un archivo.
      infixToPostfix(String infixExpression): Convierte una expresión infix a postfix.
      Otros métodos auxiliares relacionados con la ejecución del programa.
      CalculatorSingleton:

      Métodos:

getInstance(): Devuelve la única instancia de la calculadora.
DoublyLinkedList<T>:

Implementa la interfaz List.
Métodos:
isEmpty(): Verifica si la lista está vacía.
addFirst(T item): Agrega un elemento al inicio de la lista.
removeFirst(): Retira y devuelve el primer elemento de la lista.
List<T>:

Métodos:
isEmpty(): Verifica si la lista está vacía.
addFirst(T item): Agrega un elemento al inicio de la lista.
removeFirst(): Retira y devuelve el primer elemento de la lista.
ListFactory<T>:

Métodos:
createList(String implementation): Crea una instancia de lista según la implementación especificada.
ListStack<T>:

Implementa la interfaz UVGStack utilizando la clase java.util.Stack.
SinglyLinkedList<T>:

Implementa la interfaz List.
Métodos:
isEmpty(): Verifica si la lista está vacía.
addFirst(T item): Agrega un elemento al inicio de la lista.
removeFirst(): Retira y devuelve el primer elemento de la lista.
StackFactory<T>:

Métodos:
createStack(String implementation): Crea una instancia de stack según la implementación especificada.
UVGStack<T>:

Métodos:
isEmpty(): Verifica si el stack está vacío.
push(T item): Agrega un elemento al stack.
pop(): Retira y devuelve el elemento superior del stack.
peek(): Devuelve el elemento superior del stack sin retirarlo.
size(): Devuelve el tamaño del stack.
VectorStack<T>:

Implementa la interfaz UVGStack utilizando la clase java.util.Vector.
CalculatorProgramTest:

Contiene pruebas JUnit para validar el funcionamiento de los métodos infixToPostfix y evaluateExpression.

7.

El patrón Singleton ofrece ventajas como control de acceso, conservación de recursos y prevención de inconsistencias de datos. Sin embargo, también presenta desventajas como acoplamiento fuerte, dificultad para extender la funcionalidad y problemas de concurrencia. En el contexto de esta hoja, su uso puede ser adecuado si se necesita una única instancia de una clase en todo el programa y no se esperan cambios en esta instancia durante la ejecución.

Diagrama de Secuencia UML:

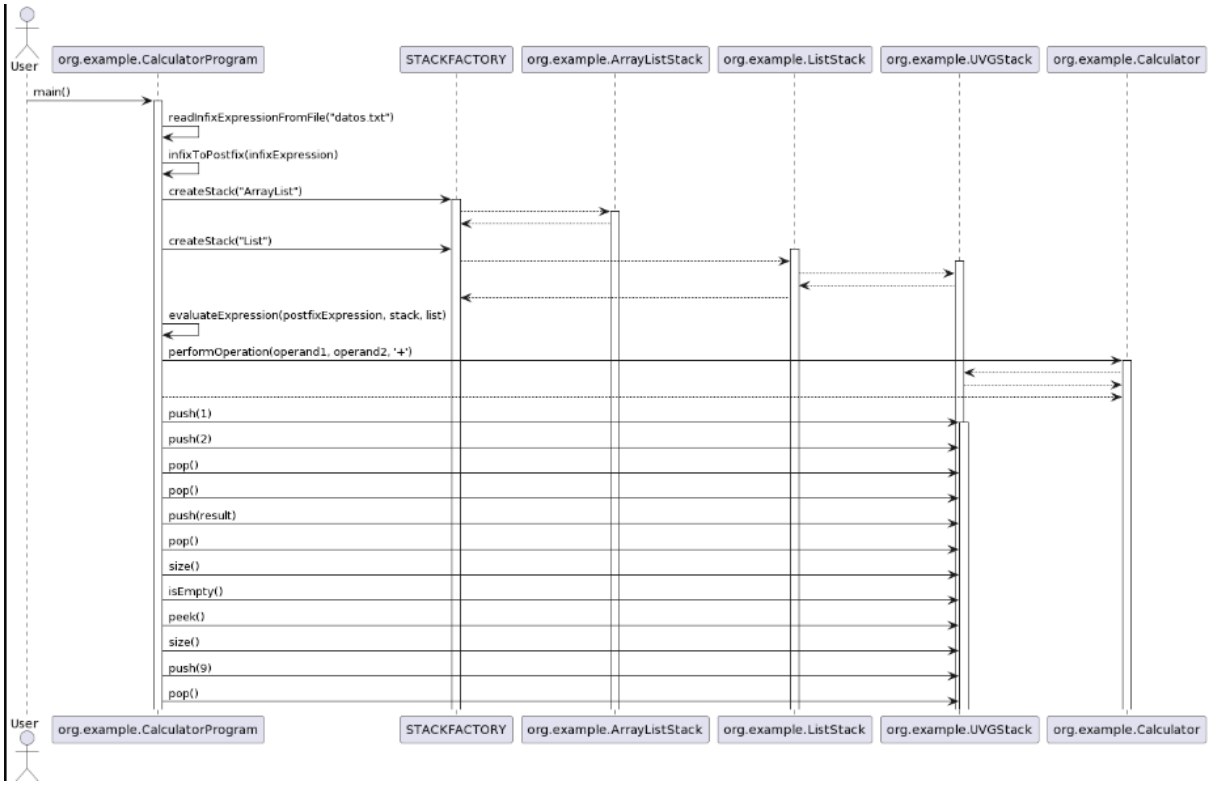
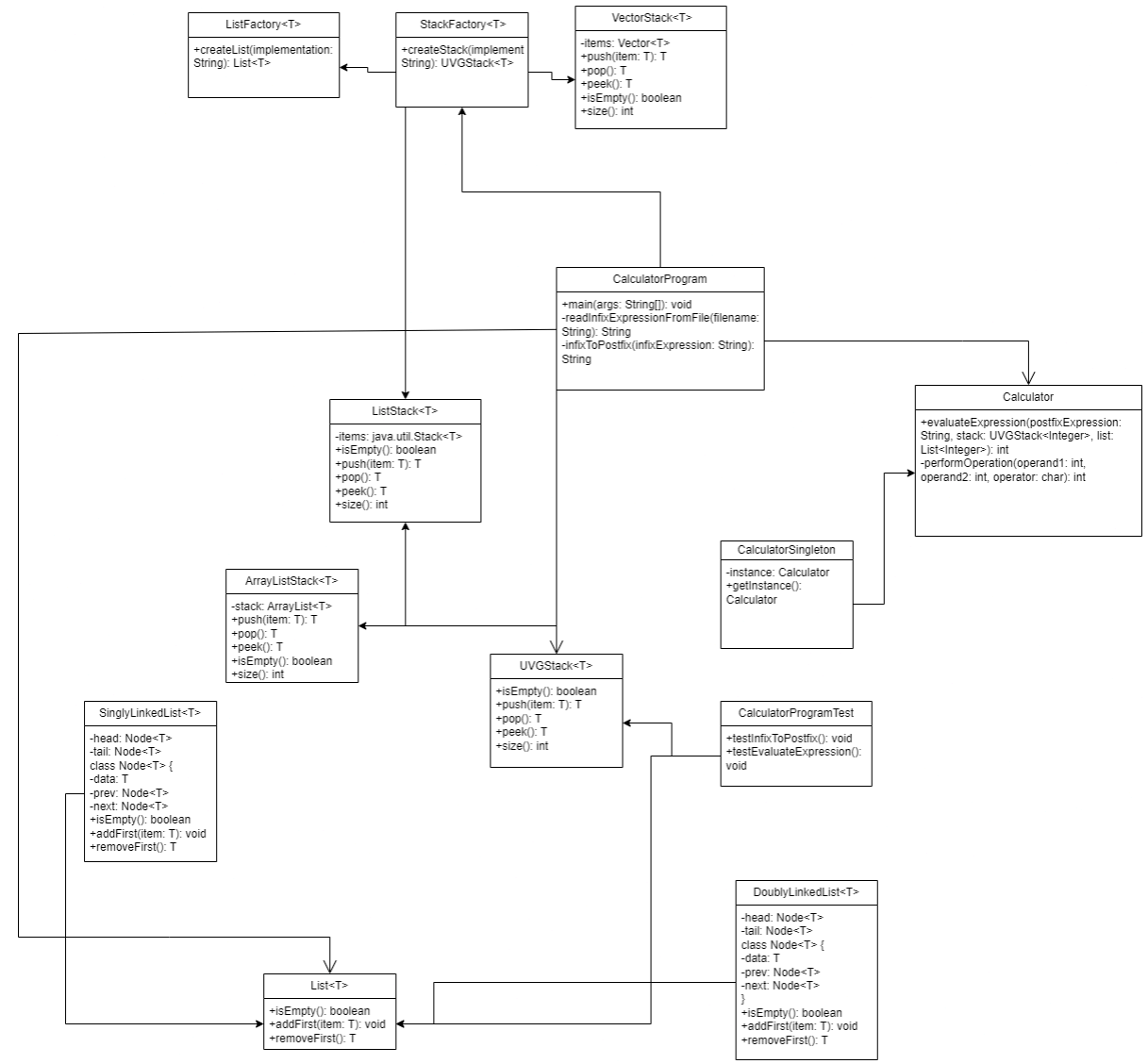


Diagrama de Clases UML:



https://drive.google.com/file/d/1rRi_oJBY_5B-DTxegc775WTvnDzVMWdT/view?usp=sharing