Guatemala 25 de Enero de 2024
Adrián López - 231361
Luis Fernando Mendoza -

**Algoritmos y Estructura de Datos**
Hoja de Trabajo 2, Diseño de ADT

---

**Objetivos:**
a. Utilización de genéricos.
b. Diseño del ADT para pilas (stack).
c. Implementación de pilas con un vector de tamaño variable. (Clase VECTOR)
d. Control de versiones del programa.
e. Emplear JUnit para casos de prueba.

*"Su programa debe leer de un archivo de texto, una expresión en formato Postfix y producir el resultado de la misma. El archivo de texto se llama datos.txt y será proporcionado por su auxiliar al correr el programa. En cada línea del archivo de texto vendrá una expresión en notación postfix."*

**Diagrama de UML:**

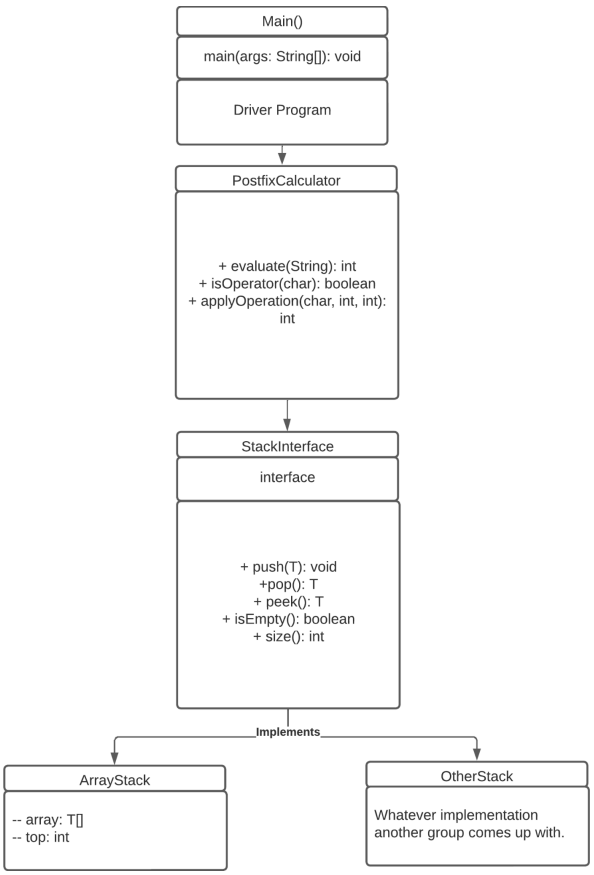
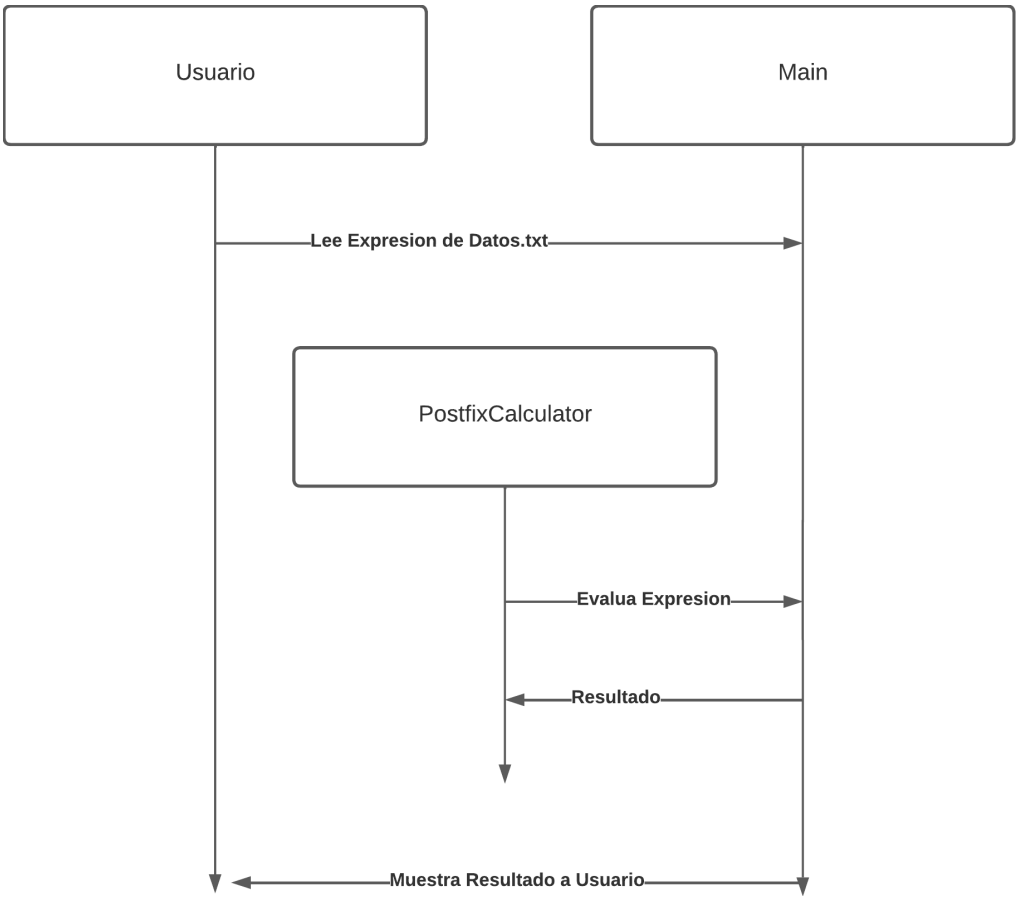


**Diagrama de Secuencia:**




**Evidencia de Desarrollo (GitHub):**
https://github.com/adirnnn/HDT2-ADT-Program.git

**Análisis:**
*Main.java:*
Propósito: Driver Program, el cual es el punto de entrada de la aplicación. Su función principal es leer una expresión de un archivo de texto y pasarla a PostfixCalculator para su evaluación y mostrar el resultado al usuario.

- Lee una expresión de un archivo de texto.
- Instancia un objeto PostfixCalculator.
- Evalúa la expresión usando el objeto PostfixCalculator.
- Muestra el resultado de la expresión al usuario.

Comentarios: Esta clase maneja la interacción con el usuario y coordina la evaluación de la expresión Postfix.

*PostfixCalculator.java:*
Propósito: Esta clase realiza la evaluación de expresiones en formato Postfix.

- evaluate(String expression): Recibe una expresión en formato Postfix como entrada y la evalúa para devolver el resultado.
- isOperator(char c): Verifica si un carácter es un operador aritmético.
- applyOperation(char operator, int operand1, int operand2): Aplica la operación correspondiente a dos operandos según el operador dado.

Comentarios: Esta clase implementa la lógica necesaria para evaluar expresiones Postfix utilizando una pila.

*StackInterface.java:*

Propósito: Esta interfaz define las operaciones básicas que debe tener una pila.

- push(T element): Inserta un elemento en la pila.
- pop(): Elimina y devuelve el elemento superior de la pila.
- peek(): Devuelve el elemento superior de la pila sin eliminarlo.
- isEmpty(): Verifica si la pila está vacía.
- size(): Devuelve el tamaño actual de la pila.

Comentarios: Esta interfaz define un contrato que deben cumplir las implementaciones de pilas.

*ArrayStack.java:*

Propósito: Esta clase proporciona una implementación de pila utilizando un arreglo de tamaño variable.

- push(T element): Inserta un elemento en la pila, redimensionando el arreglo si es necesario.
- pop(): Elimina y devuelve el elemento superior de la pila.
- peek(): Devuelve el elemento superior de la pila sin eliminarlo.
- isEmpty(): Verifica si la pila está vacía.
- size(): Devuelve el tamaño actual de la pila.

Comentarios: Esta clase implementa la interfaz StackInterface utilizando un arreglo dinámico para almacenar los elementos de la pila.