Hoja de trabajo 2

1) Análisis

- **Requisitos del sistema:**
    - **Lectura de Expresiones Postfix**
        - El programa debe leer expresiones en notación postfix desde un archivo de texto llamado `datos.txt`.
        - Cada línea del archivo contendrá una expresión postfix válida.

    - **Evaluación de Expresiones Postfix**
        - Se debe procesar la expresión utilizando una pila.
        - Los operandos se almacenan en la pila hasta encontrar un operador.
        - Al encontrar un operador, se extraen los dos operandos más recientes y se evalúa la operación.
        - Se debe preservar el orden de los operandos (algunas operaciones no son conmutativas).

    - **Implementación de una Pila Genérica**
        - Diseño de un ADT de pila utilizando genéricos.
        - Implementación basada en un vector de tamaño variable (clase `Vector`).
        - Alternativa de implementación con `ArrayList`.

    - **Desarrollo del ADT Calculadora**
        - Debe permitir evaluar expresiones postfix.
        - Debe ser intercambiable entre diferentes programas principales.
        - Manejo de errores como división entre cero, caracteres inválidos o insuficiencia de operandos.

    - **Control de Versiones**
        - Uso de un repositorio de control de versiones (Git).
        - Evidencia de al menos tres versiones guardadas.

    - **Pruebas Unitarias con Junit**
        - Se deben incluir pruebas unitarias para la pila.
        - Se deben incluir pruebas unitarias para la calculadora.

- **Clases:**
    - **Pila<T>:** Representa una pila genérica, implementada con un vector de tamaño variable.
        - **Atributos:**
            - ✓ elementos (Vector<T>)-: Almacena los elementos de la pila.
            - ✓ tamanio (int): Cantidad de elementos en la pila.
        - **Métodos:**
            - ✓ push(T elemento): Agrega un elemento a la pila.
            - ✓ T pop(): Extrae el elemento en la cima de la pila.
            - ✓ T peek(): Devuelve el elemento en la cima sin extraerlo.
            - ✓ boolean isEmpty(): Indica si la pila está vacía.
            - ✓ int size(): Devuelve el número de elementos en la pila.

    - **Vector<T>:** Implementación de un vector de tamaño variable.
        - **Atributos:**
            - ✓ datos (T[]): Arreglo que almacena los elementos.
            - ✓ capacidad (int): Capacidad actual del vector.
            - ✓ tamanio (int): Cantidad de elementos en el vector.
        - **Métodos:**
            - ✓ void agregar(T elemento): Agrega un elemento al vector.
            - ✓ T obtener(int indice): Obtiene un elemento en una posición específica.
            - ✓ void eliminar(int indice): Elimina un elemento en una posición específica.
            - ✓ int getTamanio(): Devuelve el número de elementos almacenados.

    - **CalculadoraPostfix:** Evalúa expresiones postfix utilizando una pila.
        - **Atributos:**
            - ✓ pila (Pila<Integer>): Pila utilizada para la evaluación.
        - **Métodos:**
            - ✓ int evaluar(String expresion): Evalúa una expresión postfix y devuelve el resultado.
            - ✓ boolean esOperador(String token): Verifica si un token es un operador.
            - ✓ int operar(int a, int b, String operador): Realiza la operación aritmética.

    - LectorArchivo: Lee las expresiones postfix desde un archivo de texto.
        - **Atributos:**

- ✓ nombreArchivo (String): Nombre del archivo a leer.

➢ **Métodos:**

- ✓ List<String> leerExpresiones(): Lee y devuelve una lista de expresiones postfix.

- Main: Clase principal para la ejecución del programa.
  - ➢ **Métodos:**
    - ✓ public static void main(String[] args): Inicia el programa, lee las expresiones y las evalúa usando la calculadora.

2) UML