Practica UC2 Estructura de Datos

I. Objetivos de la Práctica

1. **Diseñar y documentar el TDA (Tipo de Dato Abstracto):**

   Definir la estructura de datos (por ejemplo, una lista enlazada simple), sus componentes, restricciones y el detalle de cada método (nombre, entradas, salidas, pre y post condiciones).

2. **Implementar la estructura en Java:**

   Codificar la estructura de datos y sus métodos en NetBeans siguiendo buenas prácticas de programación.

3. **Realizar un reporte utilizando el Debugger de NetBeans:**

   Analizar y documentar la ejecución de cada método, identificando el flujo y los cambios en las variables.

4. **Gestionar el código mediante GitHub integrado en NetBeans:**

   Configurar, versionar y subir el proyecto a GitHub desde NetBeans, y compartir el repositorio con el docente.

II. Estructura y Distribución de la Práctica

La práctica se divide en cuatro secciones principales. Cada sección incluye pasos detallados y actividades recurrentes para cada método de la estructura de datos.

A. Sección 1: Diseño del TDA y Especificación de Métodos

1. **Definir el TDA:**
   - **Ejemplo:** Lista enlazada simple.
   - **Elementos:**
     - **NodoInicial:** Nodo que representa la lista
     - **TamañoLista:** Contador de elementos
   - **Restricciones:**
     - La lista está vacía si el puntero es nulo.
     - No se permiten datos nulos en la inserción.
   - Utilicen el siguiente formato:

TDA: Lista enlazada simple circular

Elementos:

Elemento	Descripción	Restricciones
inicial	Representación del primer elemento de la lista	● Puede ser nulo. ● Los elementos de la lista tienen que ser solo números enteros. ● Los números de cada elemento pueden tener una extensión máxima de 50 dígitos.
tam	Representa el número total de elementos en la lista	ninguno

Métodos:

Nombre	Utilidad	Entrada	Salida	precondición	Postcondición
isEmpty	Determina si la lista está vacía	No aplica	true si está vacía, false en caso contrario	Ninguna	Devuelve si la lista está vacía
Size	Regresa la cantidad de elementos que conforman la lista	No aplica	El número de elementos >= 0	Ninguna	Ninguna
addFirst	Añade elementos al principio de la lista	El elemento (e), que se desea añadir a la lista	Una confirmación de si el elemento e pudo ser añadido	Verificar que el elemento e sea de un tipo de dato valido para la lista	La lista L con el nuevo elemento e, si era un elemento valido
addLast	Añade elementos al final de la lista	Elemento a añadir	Confirmación de adición	Debe ser un tipo válido	Elemento agregado al final

removeFirst	Elimina el elemento al inicio de la lista	No aplica	Elemento eliminado	La lista no debe estar vacía	Se reduce el tamaño de la lista
removeLast	Elimina el ultimo elemento de la lista	No aplica	Elemento eliminado	La lista no debe estar vacía	Se reduce el tamaño de la lista
contains	Busca el elemento (e) dentro de la lista	Elemento a buscar	true si está, false si no	Lista inicializada	No modifica la lista
getElement	Obtiene un elemento en un índice	Índice de búsqueda	Elemento en la posición	Índice válido dentro del tamaño	No modifica la lista
Clear	Elimina todos los elementos de lista	No aplica	Lista vacía	Lista inicializada	Todos los elementos eliminados
toString	Muestra en consola cada uno de los elementos de la lista	No aplica	No aplica	No aplica	No aplica

2. **Especificación de Cada Método (a implementar gradualmente):**
    - **Formato Requerido:**
        - **Nombre del Método:** Ej. insertarAlFinal, eliminarPorValor.
        - **Entradas:** Descripción y tipo de cada parámetro.
        - **Salida:** Valor de retorno (ej. booleano para indicar éxito).
        - **Precondiciones:** Condiciones que deben cumplirse antes de ejecutar el método.
        - **Postcondiciones:** Cambios en la estructura después de la ejecución.

Nota: Esta documentación se debe actualizar a medida que se añadan nuevos métodos a la estructura.

B. Sección 2: Implementación en Java usando NetBeans

1. Creación del Proyecto

- **Abrir NetBeans y crear un nuevo proyecto:**
    - Menú: Archivo > Nuevo Proyecto
    - Seleccionar: **Java > Aplicación Java**
    - **Nombre del Proyecto:** ED_UC2_ID (sustituir "ID" por tu identificador o apellido según lo indique el docente)
    - Definir la ubicación y finalizar el asistente.
- **Estructura del Proyecto:**
    - Crear las carpetas: nodos, ed_ineales y ed_noLineales, donde agregaran las clases de las diferentes estructuras de datos. Por ejemplo, LES y dentro los métodos para la lista.

2. Codificación de la Estructura de Datos

- **Crear las Clases Necesarias:**
    - **Según el método que se les pida deberán realizar la implementación correspondiente.**
- **Implementación Gradual:**
    - Escribe cada método siguiendo el formato definido en la Sección 1.
    - Agrega comentarios en formato JavaDoc para explicar entradas, salidas, pre y post condiciones en el mismo Código.
    - Realiza pruebas en el método main de forma incremental conforme se añade cada método.

C. Sección 3: Reporte y Uso del Debugger en NetBeans

1. Preparación para la Depuración

- **Configurar Breakpoints:**
    - Ubica breakpoints en:
        - El inicio de cada método.
        - Antes y después de operaciones críticas (por ejemplo, cambios en enlaces de la lista).
    - **Cómo hacerlo en NetBeans:**
        - Haz clic en el margen izquierdo del editor junto a la línea de código deseada.

2. Documentación del Proceso de Debugging

- **Registro de Ejecución:**
    - Por cada breakpoint, anota:
        - Línea de código ejecutada.
        - Valores actuales de las variables relevantes.
        - Confirmación de que se cumplen las precondiciones.
        - Cambios en la estructura de datos (postcondiciones).
        - Observaciones de comportamientos inesperados o errores.
- **Capturas de Pantalla:**

- Toma imágenes del estado del debugger cada vez que avances una línea de codigo (variables, pila de llamadas) para incluirlas en el reporte.

- **Preguntas Guía (para cada método):**
  - ¿Qué valores tienen las variables antes de ejecutar el método?
  - ¿Cómo cambia la estructura al finalizar la ejecución?
  - ¿Se cumple la lógica esperada en cada paso?
  - ¿Qué se observa en la consola de NetBeans (errores, advertencias)?
- **Formato del Reporte:**
  - Organiza el reporte en secciones correspondientes a cada método.
  - Incluye tablas o listas con las respuestas a las preguntas guía.

D. Sección 4: Gestión del Proyecto en GitHub desde NetBeans

1. Configuración de Git en NetBeans

- **Activar el Control de Versiones:**
  - En NetBeans, abre el proyecto y ve a Team > Git > Initialize Repository…
  - Selecciona la carpeta raíz del proyecto para inicializar Git.
- **Realizar el Primer Commit:**
  - Menú: Team > Git > Commit…
  - Selecciona todos los archivos, añade un mensaje (ej. “Commit inicial: creación del proyecto ED_UC2_ID”) y confirma.

2. Creación y Configuración del Repositorio Remoto en GitHub (desde NetBeans)

- **Crear el Repositorio en GitHub:**
  - Accede a GitHub a través del navegador y crea un nuevo repositorio con el nombre ED_UC2_ID.
  - No es necesario agregar archivos de configuración inicial (README, .gitignore) desde GitHub, ya que se gestionará desde NetBeans.
- **Configurar la Conexión Remota en NetBeans:**
  - En NetBeans, ve a Team > Git > Remote > Set Remote…
  - Introduce la URL del repositorio (por ejemplo, https://github.com/tu_usuario/ED_UC2_ID.git).
  - Autentícate si es necesario.

- **Realizar el Primer Push:**
    - Menú: Team > Git > Remote > Push…
    - Selecciona la rama (por lo general, master o main) y realiza el push de los cambios.

3. Actualización Continua y Compartir el Repositorio

- **Subida de Avances:**
    - Cada vez que se implemente o se modifique un método, realiza:
        1. Commit en NetBeans: Team > Git > Commit…
        2. Push de los cambios: Team > Git > Remote > Push…
- **Compartir el Repositorio con el Docente:**
    - En GitHub, desde la página del repositorio:
        - Ve a la pestaña Settings > Manage Access.
        - Agregarme como colaborador usando mi correo sergio.castellanos90851@potros.itson.edu.mx
    - También puedes enviar el enlace directo del repositorio si tienen dudas.

III. Consejos Adicionales y Buenas Prácticas

- **Documentación Continua:**
  Mantén actualizada la documentación del TDA y el reporte de debugging con cada avance. Esto facilitará la revisión y comprensión del proceso.
- **Uso Consistente del Debugger:**
  Aprovecha el debugger en cada sesión de prueba para identificar errores a tiempo y entender el flujo de ejecución.