**HOJA DE TRABAJO 4**

**(ANÁLISIS Y DISEÑO)**

**Descripción de clases**

* **Stack <T>:** Establece el funcionamiento de una pila genérica
* **AbstractStack <T>:** Define las operaciones que debe realizar la pila sin implementarlas
* **ArrayListStack <T>:** Implementa las operaciones definidas en el abstract usando un ArrayList como método de almacenamiento de los datos resultantes.
* **VectorStack <T>:** Implementa las operaciones definidas en el abstract usando un Vector como método de almacenamiento de los datos resultantes.
* **Main:** Se encarga de permitir y manejar la interacción del usuario con el programa

**Descripción de Atributos**

* **Stack <T>:** No tiene
* **AbstractStack <T>:** No tiene
* **ArrayListStack <T>:**
* **Private ArrayList<T> stack:** Es la estructura de datos que actuará como almacén para los datos resultantes de las operaciones.
* **VectorStack <T>:**
* **Private Vector <T> stack:** Es la estructura de datos que actuará como almacén para los datos resultantes de las operaciones.
* **Main:**
* **Scanner scanner:** Permite al usuario ingresar datos
* **AbstrackStack<String> stack:** Almacena las instancias de la pila.

**Descripción de Métodos**

* **Stack <T>:**
* **void push(T item):** Permite al usuario agregar una pila instanciada.
* T pop(): Muestra en pantalla cual es el primer el elemento de la pila y elimina dicho elemento. Si la pila está vacía le muestra al usuario un mensaje que dice “Pila vacía”.
* T peek(): Muestra en pantalla cual es el primer el elemento de la pila. Si la pila está vacía le muestra al usuario un mensaje que dice “Pila vacía”.
* boolean isEmpty(): Verifica si una pila contiene elementos
* **AbstractStack <T>:**
* **push(T item):** Método astracto
* **T pop():** Método astracto
* **T peek():** Método astracto
* **boolean isEmpty:** Método abstracto
* **ArrayListStack <T>:**
* **push(T item):** Permite agregar un nuevo al ArrayList en la última posición.
* **T pop():** Imprime en pantalla y elimina el primer el elemento que contiene el ArrayList
* **T peek():** Imprime en pantalla el primer el elemento que contiene el ArrayList
* **boolean isEmpty():** Analiza el ArrayList y devuelve un valor booleano (verdadero) si el ArrayList está vacío, en el caso contrario devuelve un valor booleano (falso)
* **VectorStack <T>:**
* **push(T item):** Permite agregar un nuevo al vector en la última posición.
* **T pop():** Imprime en pantalla y elimina el primer el elemento que contiene el vector.
* **T peek():** Imprime en pantalla el primer el elemento que contiene el vector.
* **boolean isEmpty():** Analiza el vector y devuelve un valor booleano (verdadero) si el vector está vacío, en el caso contrario devuelve un valor booleano (falso).
* **Main:**
* Scanner: Permite la entrada de datos desde la consola.
* AbstractStack<String> stack: Representa una pila genérica de tipo String, cuya implementación se elige en tiempo de ejecución entre ArrayListStack y VectorStack.
* boolean running: Controla la ejecución del bucle que gestiona las operaciones de la pila.

Métodos

* public static void main(String[] args): Crea un objeto Scanner para recibir la entrada del usuario. Permite al usuario elegir la implementación de la pila (ArrayListStack o VectorStack). Proporciona un menú interactivo con las siguientes operaciones:
* Push: Agrega un elemento a la pila.
* Pop: Elimina el elemento superior de la pila.
* Peek: Muestra el elemento superior sin eliminarlo.
* isEmpty: Verifica si la pila está vacía.
* Salir: Termina la ejecución del programa. Utiliza un bucle while para permitir múltiples operaciones hasta que el usuario decida salir. Cierra el objeto Scanner al finalizar la ejecución.

**UML:**

Diagrama

El contenido generado por IA puede ser incorrecto.