ESTRUCTURAS DE DATOS
CURSO 2022-23.
Práctica 2: Pilas y Colas.

1. OBJETIVOS:

- Utilizar estructuras de datos Pila y *Stack* para resolver diferentes algoritmos.
- Utilizar estructuras de datos Cola para resolver diferentes algoritmos.

2. Apertura Proyecto.

2.1. Abrir el proyecto "*ED 2 Practica. PilasColas".*

Descargue el archivo "ED 2 Practica. PilasColas.zip" de *Moodle* y descomprímalo. A continuación, abra el proyecto con *IntelliJ* y compruebe los archivos Java que contiene:

- **Nodo**, **Pila** y **Cola** son las clases que implementan los TAD Pila y Cola a través de listas enlazadas, de forma similar a como se ha visto en clase.
- **Etiqueta, ListaEtiquetas, Principal y Pruebas** son clases completamente desarrolladas con las que se va a trabajar en esta práctica.
- **UtilizacionPila, UtilizacionStack y UtilizacionCola** son las clases en las que se tendrán que desarrollar una serie de métodos propuestos.

Definición del JDK para el proyecto.

Es muy posible que al tratar de abrir un fichero Java del proyecto, aparezca en la parte superior de la pantalla el mensaje *"Project SDK is not defined",* además de errores en el código fuente indicando que no se reconocen las clases de biblioteca (*System*, *String*, etc). Esto es debido a que el proyecto ha sido creado con una versión de JDK diferente a la que tiene el sistema en donde va a hacer la práctica.

El problema se soluciona fácilmente, definiendo de forma adecuada el JDK para el proyecto. Se proponen a continuación dos métodos diferentes:

1. Junto al mensaje de error *"Project SDK is not defined"* aparece un enlace *"Setup SDK"*. Al pulsarlo, aparecen los JDK de los que tiene constancia *IntelliJ*. Se elige uno cualquiera y el problema queda resuelto. En caso de que no aparezca ninguno, se utilizará el segundo método, que se propone a continuación.
2. la barra de menú de *IntelliJ*, pulsar ***File --> Project Structure***. En ***Project Settings***, ***Project***, existe una lista desplegable para ***Project SDK***. En ella aparecen los JDK de los que tiene constancia *IntelliJ*, se seleccionará uno cualquiera y el problema quedará resuelto.

En caso de que *IntelliJ* no tenga constancia de ningún JDK, se pulsará el botón ***New*** para definirle uno. Se selecciona la opción JDK, se le indica la ubicación del JDK en el sistema de archivos, por ejemplo, C:\Program Files\Java\jdk-11.0.5 y se pulsa OK.

Nota: En las clases *UtilizacionPila, UtilizacionCola*, y *UtilizaciónStack* se deberá sustituir el comentario correspondiente al autor *(@author)*, por los datos del alumno/a que desarrolla la práctica (incluyendo nombre, apellidos y número de matrícula).

Nota: Además de las pruebas propuestas, el alumno/a podrá realizar las pruebas adicionales que considere oportunas.

3. Clases Etiqueta, ListaEtiquetas, Principal y Pruebas.

3.1. Etiqueta.

Clase que se utilizará para la gestión de etiquetas en *html*. La clase se entrega completamente desarrollada, no hay que realizar ninguna modificación. Contiene dos atributos de tipo *String* (apertura y cierre). Además, contiene los siguientes métodos:

- `public Etiqueta (String dato)`. Constructor de la clase, crea las etiquetas de apertura y cierre a partir de un *String*.
- `public String getApertura()`. Devuelve la etiqueta de apertura.
- `public String getCierre()`. Devuelve la etiqueta de cierre.

3.2. ListaEtiquetas.

Clase que se utilizará para la gestión de una lista de etiquetas en *html*. La clase se entrega completamente desarrollada, no hay que realizar ninguna modificación. Contiene los siguientes atributos:

- `private final int MAXIMO`, que representa el tamaño del vector que almacena la lista.
- `private int numEtiquetas`, que utilizaremos para almacenar el número de elementos que contiene la lista.
- `private Etiqueta[] lista`, vector que contiene los elementos de tipo Etiqueta

Así como los siguientes métodos públicos:

- `public ListaEtiquetas(int maximo, String[] datos)`. Constructor de la clase, que prepara la lista para usarla. Recibe como argumentos el tamaño del vector y un *array* con los datos que vamos a insertar en la lista.
- `public void mostrar()`. Método que muestra los elementos que contiene la lista.
- `public boolean esCierre(String aux)`. Método que recibe un elemento, y si es una etiqueta de cierre, devuelve verdadero.
- `public boolean esApertura(String aux)`. Método que recibe un elemento, y si es una etiqueta de apertura, devuelve verdadero.
- `public boolean esEtiqueta(String dato)`. Método que recibe un elemento, y si es una etiqueta, devuelve verdadero.
- `public boolean emparejados (String abre, String cierra)`. Método que recibe dos datos de tipo *String* y comprueba si los dos son elementos de la misma etiqueta (uno de apertura y otro de cierre).

3.3. Clases Principal y Pruebas.

Estas dos clases contienen las pruebas que se van a utilizar para comprobar el funcionamiento correcto de los métodos que se desarrollarán en las clases *UtilizacionPila*, *UtilizacionCola* y *UtilizacionStack*.

Se entregan completamente desarrolladas, no es necesario modificarlas.

3.4. Clases Nodo, Pila y Cola.

Antes de empezar a desarrollar los métodos de los siguientes apartados, hay que modificar las clases Nodo, Pila y Cola para que tanto los elementos de la Pila como los de la Cola sean de tipo *String* en vez de ser de tipo *int*.

No se permite incluir métodos adicionales en ninguna de las clases.

4. Algoritmos con Pilas.

Realizar los siguientes algoritmos de la clase UtilizacionPilas.

4.1 Algoritmo *mostrarInverso*.

Método que muestra el contenido de la pila invertido (el fondo arriba y la cima abajo):

```
public void mostrarInverso(Pila pila)
```

Este método recibe una pila como argumento. Se mostrará el contenido de la pila invertido sin modificar la pila, ni utilizar ninguna estructura de datos auxiliar. Este método se deberá realizar obligatoriamente de manera recursiva.

4.2 Algoritmo *comprobarTexto.*

Método que realiza la comprobación de un texto en html:

```
public boolean comprobarTexto (ListaEtiquetas lista, String texto)
```

Este método recibe como parámetros un texto y una lista de etiquetas. Comprueba si todas las etiquetas del texto están balanceadas (cada etiqueta de apertura se corresponde con una de cierre).

Para ello, se creará una pila y se irán introduciendo en la misma las etiquetas de apertura que se vayan detectando. Cada vez que se localice una etiqueta de cierre, se comprobará si su etiqueta de apertura correspondiente está en la cima de la pila.

Si al finalizar de tratar el texto quedan elementos en la pila:

- Se mostrará el mensaje "En la pila quedan elementos:".
- A continuación, se mostrarán dichos elementos en orden inverso (utilizando el método *mostrarInverso*).
- Y se devolverá falso como resultado.

Para realizar este método, se pueden usar los métodos de la clase *String* que se considere necesarios, así como los métodos de la clase *ListaEtiquetas*.

5. Algoritmos con *Stack*.

Realizar los siguientes algoritmos de la clase UtilizacionStack.

5.1 Algoritmo *mostrar*.

Método que muestra el contenido de un *Stack* de forma similar a como muestra los datos el método *mostrar* de la clase Pila:

```
public void mostrar (Stack<String> pila)
```

Este método recibe una pila de tipo *Stack* como argumento. Se mostrará el contenido de la pila sin modificar la pila, ni utilizar ninguna estructura de datos auxiliar. Este método se deberá realizar obligatoriamente de manera recursiva.

5.2 Algoritmo *mostrarInverso*.

Método que muestra el contenido de un *Stack* invertido (el fondo arriba y la cima abajo):

```
public void mostrarInverso(Stack<String> pila)
```

Este método recibe una pila de tipo *Stack* como argumento. Se mostrará el contenido de la pila invertido sin modificar la pila, ni utilizar ninguna estructura de datos auxiliar. Este método se deberá realizar obligatoriamente de manera recursiva.

5.3 Algoritmo *comprobarTexto.*

Método que realiza la comprobación de un texto en html:

```
public boolean comprobarTexto (ListaEtiquetas lista, String texto)
```

Este método recibe como parámetros un texto y una lista de etiquetas. Comprueba si todas las etiquetas del texto están balanceadas (cada etiqueta de apertura se corresponde con una de cierre).

Para ello, se creará una pila de tipo *Stack<String>* y se irán introduciendo en la misma las etiquetas de apertura que se vayan detectando. Cada vez que se localice una etiqueta de cierre, se comprobará si está en la cima de la pila. Si no se corresponden, se dejará de comprobar el texto. Si al finalizar de tratar el texto quedan elementos en la pila:

- Se mostrará el mensaje "En la pila quedan elementos:".
- A continuación, se mostrarán dichos elementos en orden inverso (utilizando el método *mostrarInverso*).
- Y se devolverá falso como resultado.

Para realizar este método, se pueden usar los métodos de la clase *String* que se considere necesarios, así como los métodos de la clase *ListaEtiquetas*.

6. Algoritmos sobre Colas.

Realizar los siguientes algoritmos de la clase UtilizacionColas.

6.1 Algoritmo *leerTexto*.

Método que lee un texto y extrae las etiquetas *html* que contiene el mismo, guardándolas en una cola:

```
public Cola leerTexto (ListaEtiquetas lista, String texto)
```

Este método recibe como argumentos una lista de etiquetas *html* y un texto de tipo *String*. Primero se crea un objeto Cola, y luego se introducen en la misma todas las etiquetas detectadas. Por último, se devuelve como resultado el objeto Cola creado.

6.2 Algoritmo *comprobarHtml*

En este apartado se comprobará si el conjunto de etiquetas de una cola está balanceado:

```
public boolean comprobarHtml (Cola cola, ListaEtiquetas lista)
```

Este método recibe como parámetro una cola que contendrá las etiquetas de un texto (extraídas por medio del método *leerTexto*) y una lista de etiquetas.

La solución de este apartado será un algoritmo iterativo en el que está permitido utilizar una estructura de datos auxiliar del tipo Pila. Al finalizar la ejecución del método, la cola deberá quedar con el contenido inicial sin alterar.

Si al finalizar de tratar el texto quedan elementos en la pila auxiliar:

- Se mostrará el mensaje "En la pila quedan elementos:".
- A continuación, se mostrarán dichos elementos (utilizando el método *mostrar* de la clase Pila).
- Y se devolverá falso como resultado.

7. Entrega de la práctica.

Se entregará el proyecto *IntelliJ* resultante de hacer la práctica: “ED 2 Practica. PilasColas”, **que tendrá que tener exactamente ese nombre**.

Para entregarlo, se comprimirá el proyecto en un archivo, preferentemente ZIP, y se subirá a la plataforma. El nombre de dicho archivo ZIP será el mismo: “ED 2 Practica. PilasColas.zip”.

Importante: No olvide que el proyecto debe incluir todas las clases del proyecto inicial. No se pueden modificar los nombres de los métodos ni los argumentos de los mismos.