Algoritmos y Estructuras de Datos

Indexed List, Pilas (LIFO) y Colas (FIFO)

Guillermo Román Díez

groman@fi.upm.es

Universidad Politécnica de Madrid

Curso 2021/2022

La librería aedlib

- En la asignatura usaremos una librería propia: "aedlib"
- "aedlib" implementa varios tipos abstractos de datos (TADs): listas indexadas, pilas, colas (ya estudiados en Programación II), conjuntos, arboles, grafos, etc...
- La librería está disponible en el Moodle de la asignatura: `Laboratorios y Ejercicios Individuales` → `aedlib.jar`
- La documentación de aedlib está disponible en: `http://costa.ls.fi.upm.es/entrega/aed/docs/aedlib/`

Motivación

Pregunta

¿qué es una lista indexada?

Motivación

Pregunta

¿qué es una lista indexada?

Pregunta

¿cuáles son las diferencias entra una lista y un array?

Motivación

Pregunta

¿qué es una lista indexada?

Pregunta

¿cuáles son las diferencias entra una lista y un array?

- Una lista es una colección de elementos en un cierto orden
- El orden no depende de los elementos en sí, sino de la posición que ocupan en la lista
- A nivel teórico puede tener infinitos elementos. Luego en la realidad...
- Decimos que una lista está indexada cuando se usa un **índice** para acceder a los elementos
  - Habitualmente se utiliza un índice numérico
- Tiene innumerables usos en programación

Interfaz `IndexedList<E>`

```
public interface IndexedList<E> extends Iterable<E> {

  public void add(int index, E e) throws IndexOutOfBoundsException;

  public E get(int index) throws IndexOutOfBoundsException;

  public boolean isEmpty();

  public int size();

  public E set(int index, E e) throws IndexOutOfBoundsException;

  public int indexOf(E search);

  public E removeElementAt(int i) throws IndexOutOfBoundsException;

  public boolean remove(E element);

  public Object [] toArray();
}
```

Implementaciones de `IndexedList<E>`

- La documentación detallada de `IndexedList<E>` está diponible en

  `http://costa.ls.fi.upm.es/entrega/aed/docs/aedlib/es/upm/aedlib/indexedlist/IndexedList.html`
- La clase `ArrayIndexedList<E>` implementa el interfaz `IndexedList<E>`
  - Utiliza internamente un array para el almacenamiento de los elementos
  - Dispone de dos constructores: uno sin parámetros y el constructor de copia
- Tanto `IndexedList<E>` como `ArrayIndexedList<E>` como se encuentran disponibles en el paquete: `es.upm.aedlib.indexedlist` de la librería `aedlib.jar`

Motivación

Pregunta

¿qué es una pila? ¿qué es una cola?

Motivación

Pregunta

¿qué es una pila? ¿qué es una cola?

Pregunta

¿qué significa LIFO? ¿y FIFO?

Motivación

Pregunta

¿qué es una pila? ¿qué es una cola?

Pregunta

¿qué significa LIFO? ¿y FIFO?

- Las **pilas** y las **colas** son TADs fundamentales con innumerables aplicaciones
- Se usan en multitud de aplicaciones
  - Pila de llamadas a métodos
  - Gestión del historial de acciones / navegación
  - Sistemas Operativos (round robin)
  - La JVM tiene una *arquitectura de pila*
  - ...

¿Qué es un LIFO?

- LIFO $\Rightarrow$ Last In First Out
- Una Pila, LIFO o Stack es un TAD contenedor que consiste en una secuencia lineal de elementos donde:
  - El último elemento **apilado (push)** es el primer elemento en ser **desapilado(pop)**
- No existe operación de búsqueda sobre los elementos
- Únicamente se puede consultar la cima de la pila
- No tiene límite de tamaño (teórico. . .)
  - Os suena **Stack Overflow**?

Interfaz LIFO<E>

```
public interface LIFO<E> extends Iterable<E> {
  public int size();

  public boolean isEmpty();

  public E top() throws EmptyStackException;

  public E pop();

  public void push(E elem);

  public Object [] toArray();
}
```

Implementaciones de `LIFO<E>`

- La documentación detallada de `LIFO<E>` está diponible en

  `http://costa.ls.fi.upm.es/entrega/aed/docs/aedlib/es/upm/aedlib/lifo/LIFO.html`
- La clase `LIFOList<E>` implementa el interfaz `LIFO<E>`
  - Utiliza internamente una lista para el almacenamiento de los elementos
  - Dispone de tres constructores: uno sin parámetros, otro que recibe un array de elementos y el constructor de copia
- La clase `LIFOArray<E>` también implementa el interfaz `LIFO<E>`
  - Utiliza internamente un array para el almacenamiento de los elementos
  - Dispone de cuatro constructores: uno sin parámetros, otro recibe un array de elementos, otro con una capacidad inicial y el constructor de copia
- `LIFO<E>`, `LIFOList<E>` y `LIFOArray<E>` se encuentran disponibles en el paquete: `es.upm.aedlib.lifo` de la librería `aedlib.jar`

¿Qué es un FIFO?

- FIFO $\Rightarrow$ First In First Out
- Es un TAD contenedor que consiste en una secuencia lineal de elementos donde el primer elemento **encolado (enqueue)** es el primer elemento en ser **desencolado (dequeue)**
- No existe una operación de búsqueda sobre los elementos
- Únicamente se puede consultar el primer elemento de la cola
- No tiene límite de tamaño (teórico)
  - Presenta las mismas limitaciones que la lista

Interfaz FIFO<E>

```
public interface FIFO<E> extends Iterable<E> {

  public int size();

  public boolean isEmpty();

  public E first() throws EmptyFIFOException;

  public void enqueue(E elem);

  public E dequeue() throws EmptyFIFOException;

  public Object [] toArray();
}
```

Implementaciones de `FIFO<E>`

- La documentación detallada de `FIFO<E>` está diponible en

  `http://costa.ls.fi.upm.es/entrega/aed/docs/aedlib/es/upm/aedlib/fifo/FIFO.html`
- La clase `FIFOList<E>` implementa el interfaz `FIFO<E>`
  - Utiliza internamente una lista para el almacenamiento de los elementos
  - Dispone de tres constructores: uno sin parámetros, otro que recibe un array de elementos y el constructor de copia
- La clase `FIFOArray<E>` también implementa el interfaz `FIFO<E>`
  - Utiliza internamente un array para el almacenamiento de los elementos
  - Dispone de cuatro constructores: uno sin parámetros, otro recibe un array de elementos, otro con una capacidad inicial y el constructor de copia
- `FIFO<E>`, `FIFOList<E>` y `FIFOArray<E>` se encuentran disponibles en el paquete: `es.upm.aedlib.fifo` de la librería `aedlib.jar`