Ejercicios Tema 8

Ejercicios Adaptados de Apuntes y Exámenes de EDA
Germán Moltó
gmolto@dsic.upv.es
Estructuras de Datos y Algoritmos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Traza Pila

- Supóngase que el tamaño del array sobre el que se guardan los datos de la pila es 3. ¿Cual sería su contenido y el valor del atributo *tope* después de cada una de las siguientes instrucciones?

```
Pila<Integer> p = new ArrayPila<Integer>();
p.apilar(new Integer(4));
p.apilar(new Integer(5));
p.apilar(new Integer(6));
p.apilar(new Integer(7));
Integer i = p.desapilar();
```

- ¿Cual sería el resultado de ejecutar p.toString() tras realizar las anteriores operaciones?

borraBase y topeBase en Pila

- Enriquecer el interfaz **Pila** con dos nuevas operaciones, cuyas operaciones serán:
  - public E **borraBase**(), que borra el dato situado en la base de una Pila.
  - public void **topeBase**(), que intercambia el dato que ocupa el tope de una pila con el que ocupa su base.
- Se pide: Realizar la implementación de ambos métodos en la clase ArrayPila
- ¿Cuál es el coste temporal asintótico de cada método?

Reemplazar Ocurrencias en Pila

- Se desea enriquecer el modelo de Pila con una nueva operación que reemplace todas las ocurrencias de un objeto dado por otro objeto dado, devolviendo el número de substituciones realizadas.
- Se pide:
  - Enriquecer el modelo de Pila con la nueva operación.
- Realizar la implementación de la operación en la clase ArrayPila de dos maneras diferentes
  - Utilizando únicamente las operaciones definidas en el modelo de Pila.
  - Asumiéndose conocida la implementación de la Pila y accesible en la clase ArrayPila.

Traza Cola

- Supóngase que el tamaño del array sobre el que se guardan los datos de la cola es 4. ¿Cual sería su contenido y los valores de los atributos primero,fin y tallaActual después de cada una de las siguientes instrucciones?

```
Cola<Integer> q = new ArrayCola<Integer>();
q.encolar(new Integer(5));
q.encolar(new Integer(3));
q.encolar(new Integer(7));
q.encolar(new Integer(9));
Integer y = q.desencolar();
q.encolar(new Integer(8));
Integer y = q.desencolar();
q.encolar(new Integer(0));
q.encolar(new Integer(1));
```

- ¿Cual sería el resultado de ejecutar q.toString() tras realizar las anteriores operaciones?

Localiza en LEGListaConPI

- Añadir a la clase LEGListaConPI la operación **localiza**(x) que si x está en la lista situa el PI sobre su primera aparición y sinó lo situa en fin()
- Realizar dos implementaciones diferentes:
  1. Utilizando únicamente los métodos definidos en la interfaz ListaConPI.
  2. Empleando los atributos de la implementación.

Método anterior en ListaConPI

- Dada la interfaz ListaConPI y la clase LEGListaConPI tal y como han sido definidas en clase de teoria:
- Se desea añadir un nuevo método, ***anterior*** que sitúe en el PI de una lista sobre el elemento anterior al PI actual.

1. ¿Con las definiciones de esas clases, es posible implementar el método anterior? ¿Cómo se implementaría?
   - ¿Es posible hacerlo en tiempo constante?
2. Indíquese qué modificaciones deberían realizarse en las clases ListaConPI y LEGListaConPI para permitir implementar el método anterior con coste constante.

Borrar en LEGListaConPI

- Añadir a la clase LEGListaConPI la operación ***borra***(x) que si x está en la lista borra su primera aparición y sinó lo situa en fin()
- Realizar la implementación utilizando únicamente los métodos definidos en la interfaz ListaConPI

Contar Apariciones de Cola

- Se desea añadir al modelo de *Cola* un nuevo método *contarApariciones* que cuente las apariciones de un determinado objeto x en la cola sin alterar esta.
- Para ello se va a extender el modelo con una nueva interfaz llamada *ColaExtendida*, y como se dispone de la implementación del modelo con la clase *ArrayCola* se desea también extender esta clase añadiendo la nueva operación.
  1. Implementar la interfaz *ColaExtendida*
  2. Implementar la clase *ArrayColaExtendida* suponiendo que se conoce y se tiene accesible la implementación interna de ArrayCola

Ampliando funcionalidad de ListaConPI

- Se desea añadir al modelo de ListaConPI dos nuevos métodos:
  - Vaciar la ListaConPI
  - Calcular la talla de la ListaConPI
- Para ello se va a extender el modelo con una nueva interfaz llamada ListaConPIPlus que será implementada en la clase LEGListaConPIPlus.
- Dado que se dispone de la implementación de la ListaConPI en la clase LEGListaConPI, se pide:
  - Implementar la interfaz ListaConPIPlus.
  - Implementar la clase LEGListaConPIPlus sabiendo que se dispone de acceso a la implementación.

Implementación de Modelos con Lista Enlazada

- Diséñese la clase LEGCola, que implementa la interfaz Cola mediante una LEG
- Estúdiese el coste Temporal asintótico para los métodos de la interfaz en LEGCola.
- Diséñese la clase LEGPila, que implementa la interfaz Pila utilizando una LEG.

El Modelo Secuencia (1/2)

- Sea la siguiente definición de EDA Lineal:
  - Una EDA Secuencia es una Colección Homogénea de Datos tal que el acceso a uno de ellos solo se puede realizar indicando su posición, que viene determinada por el orden en el que fue insertado
  - Se considera que el primer Dato insertado ocupa la posición cero de la Secuencia.

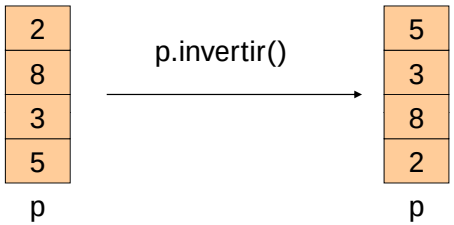
El Modelo Secuencia (2/2)

```
public interface Secuencia<E>
{
  void insertar(E x); //Inserta al final
  E borrar(E x) throws ElementoNoEncontrado;
  int indiceDe(E x); //Devuelve -1 si no está
  E recuperar(int indice);
  boolean esVacia();
  int talla();
}
```

- Diseña las clases ArraySecuencia y LEGSecuencia para implementar esta interfaz mediante una Representacion Contigua y una Enlazada respectivamente
- Discútase cual de ellas resulta más adecuada que la otra para implementar Secuencia de manera eficiente.

Invierte Pila

- Enriquecer el modelo de Pila con una nueva operación que se encargue de invertir una Pila.



- Implementar la operación en la clase ArrayPila<E>, disponiendo de acceso a la implementación.

Cambia Signo en Pila

- El siguiente método permite cambiar el signo a los elementos de una Pila de objetos Integer (Pila<Integer>):

```
public void cambiarSigno(){
    if ( !this.esVacia() ){
        int topeInt = this.desapilar().intValue();
        this.cambiarSigno();
        this.apilar(new Integer(-topeInt));
    }}
```

- Reescríbelo accediendo a la implementación de la clase LEGPilaInteger asumiendo que tiene como atributos (NodoLEG<Integer> tope; int talla)

Método anterior en LEGListaConPI (1/2)

Sea la siguiente clase genérica **LEGListaConPI** una Implementación eficiente de la interfaz ListaConPI.

```
public class LEGListaConPI<E> implements ListaConPI<E> {
protected NodoLEG<E> pri, ant, ult;  protected int talla;
public LEGListaConPI(){ pri = ult = ant = new NodoLEG<E>(null); talla = 0; }
   public void insertar(E e){…}        public void eliminar(){…}
   public void inicio(){…}             public  void siguiente(){…}
   public void fin(){…}                public E recuperar() {…}
   public  boolean esFin(){…}          public  boolean esVacia(){…}
   public String toString(){…}
}
```

Método anterior en LEGListaConPI (2/2)

1. Diseñar en la clase **LEGListaConPI** un método **anterior** que, en tiempo *lineal* y usando *sólo los atributos* de la clase, sitúe el Punto de Interés de una Lista sobre el elemento anterior al que actualmente lo ocupa, por lo que si éste resulta ser su primer elemento el método no está definido y hay que advertirlo lanzando la Excepción **ElementoNoEncontrado**.

2. Suponiendo que **LEGListaConPI** tiene Nodos Doblemente Enlazados, i.e. objetos de la clase **NodoLDEG**, escribir el código del método **anterior** e indicar su coste.