Tema 7- Modelo y Aplicación de Pila, Cola y Lista con Punto de Interés

Germán Moltó
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Tema 7- Modelo y Aplicación de Pila, Cola y Lista con Punto de Interés

Índice general:

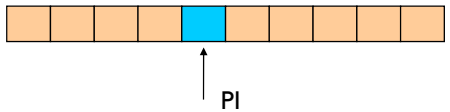
Objetivos

- Conocer diferentes patrones de acceso a datos.
- Enfatizar la separación entre especificación e implementación en una Estructura de Datos
- Comprender la importancia de la semántica de las operaciones de un modelo
- Aprender los modelos de Lista con Punto de Interés, Pila y Cola
- Resolver problemas utilizando única y exclusivamente los métodos definidos en los modelos

Lista Con Punto de Interés

- Colección homogénea de datos que solo se puede manipular accediendo secuencialmente al dato que ocupa el punto de interés.
- Modelo útil para realizar el acceso secuencial a una colección de elementos.

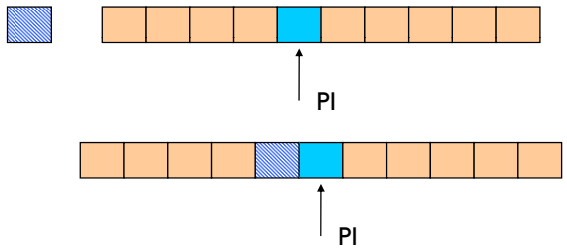
```
package modelos;
public interface ListaConPI<E> {
  void insertar(E x);
  void eliminar();
  void inicio();
  void fin();
  void siguiente();
  E recuperar();
  boolean esFin();
  boolean esVacia();
}
```



- Precondición: Las operaciones eliminar(), siguiente() y recuperar() solo se pueden ejecutar si el P.I. apunta dentro de la lista (si !esFin()).
- La operacion fin() situa el PI al final de la lista, **tras** el último elemento.

Lista Con Punto de Interés: insertar

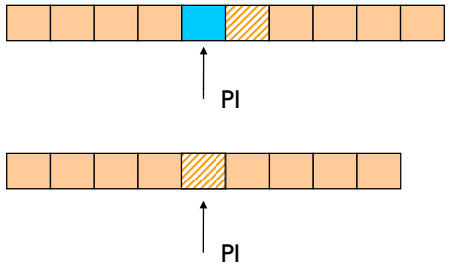
- Semántica de la operación insertar en ListaConPI:
  - El nuevo dato siempre se inserta **antes** del que ocupa el Punto de Interés.



- El PI no se ve alterado tras la operación de inserción. Sigue apuntando al mismo elemento.

Lista Con Punto de Interés: eliminar

- Semántica de la operación eliminar en ListaConPI:
  - Se elimina el elemento al que apunta el PI.



- Tras realizar el borrado, el PI apuntará al siguiente elemento de la colección (efecto secundario de la operación).

Ejemplo de Uso del Modelo ListaConPI

```
public class TestListaConPI{
  public static void main(String args[]){
   ListaConPI<String> l = new LEGListaConPI<String>();
   l.insertar(new String("patatas"));
   l.insertar(new String("limones"));
   l.insertar( "leche");
   l.inicio();
   String b = l.recuperar();
    l.eliminar();
    System.out.println(b);
  }}
```

1. ¿Qué resultado mostrará por pantalla la ejecución del programa?
2. ¿No hay que capturar ninguna excepción al llamar a eliminar?
3. ¿Por qué no escribimos ListaConPI en lugar de LEGListaConPI?

Ejemplo: Lista de la Compra

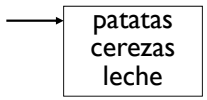
- Lista no ordenada sin elementos duplicados.
- Para insertar un nuevo elemento en la lista:
  - Recorrido secuencial ascendente de la lista desde el inicio, para detectar si el nuevo elemento es un duplicado.
  - Si llegamos al final sin encontrarlo, se añade el nuevo producto.
- Inserción al final, sin permitir elementos duplicados:

```
l.inicio();  esta = false;
while ( !l.esFin() && !esta ){
   if (l.recuperar().equals(x)) esta = true;
   else l.siguiente();
}
if (!esta) l.insertar(x);
```

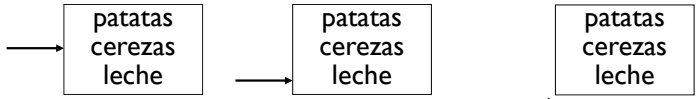
¿Por qué utilizamos equals en lugar de compareTo?

Ejemplo: Lista de la Compra (Traza)

- Insertar el producto *perejil* en la ListaConPI formada por {*patatas,cerezas,leche*}
  - Inicialización de la búsqueda (*esta = false, l.inicio()*):

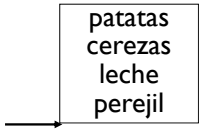


  - Búsqueda (*l.siguiente()*):



  - Resolución de la Búsqueda (*l.insertar(x)*):

¿Cual es el resultado de ejecutar l.esFin() tras el proceso?



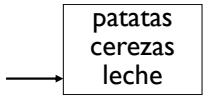
Lista Con Punto de Interés: Inserción **al principio**, sin permitir elementos duplicados

- Recorrido secuencial comprobando que el nuevo elemento NO existe, situar el PI al inicio y posterior inserción al final de la colección de elementos.

```
l.inicio(); esta = false;
while ( ! l.esFin() && !esta ){
        if (l.recuperar().equals(x)) esta = true;
        else l.siguiente();
}
if (!esta){
 l.inicio();
 l.insertar(x);
}
```

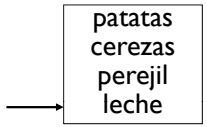

Ejercicio: Lista con Punto de Interés

- Dada la siguiente lista con punto de interés …



… sobre la que se realiza la operación: l.insertar(perejil). Dibuja el estado de la lista y comenta el resultado que se obtendría tras realizar la operación: l.recuperar().

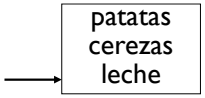
- Solución:



- l.recuperar() devolvería *leche* puesto que la inserción NO modifica el punto de interés.

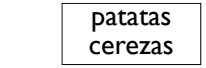
Ejercicio 2: Lista con Punto de Interés

- Dada la siguiente lista con punto de interés …



… sobre la que se realiza la operación: l.eliminar().

- Dibuja el estado de la lista y comenta el resultado que se obtendría tras realizar la operación: l.esFin().
- Solución:



- l.esFin() devolvería true.

Uso del Modelo Lista Con Punto de Interés

- El modelo de Lista con Punto de Interés se ha usado implícitamente a través de sus implementaciones contigua y enlazada.

ListaConPI

```
for (this.inicio(); !this.esFin(); this.siguiente()){
  procesar(this.recuperar());
}
```

Array

```
for (int i = 0; i < this.talla ; i++){
  procesar(this.array[i]);
}
```

Lista Enlazada

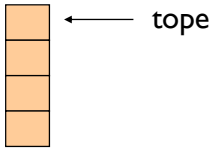
```
for (NodoLEG<E> aux = this.primero; aux != null; aux = aux.siguiente){
  procesar(aux.dato);
}
```


Estructura de Datos: Pila

- Una *Pila* es una colección homogénea de datos en la que:
  1. El acceso se realiza siguiendo un criterio LIFO (**L**ast **I**n **F**irst **O**ut), esto es, accediendo al dato que ocupa el *tope* de la pila, es decir, el último que se insertó.
- Ejemplos de Pilas:
  - Bandejas en un AutoServicio.
  - Hojas de papel en la bandeja de una impresora.
  - Pila de la recursión.

El interfaz Pila

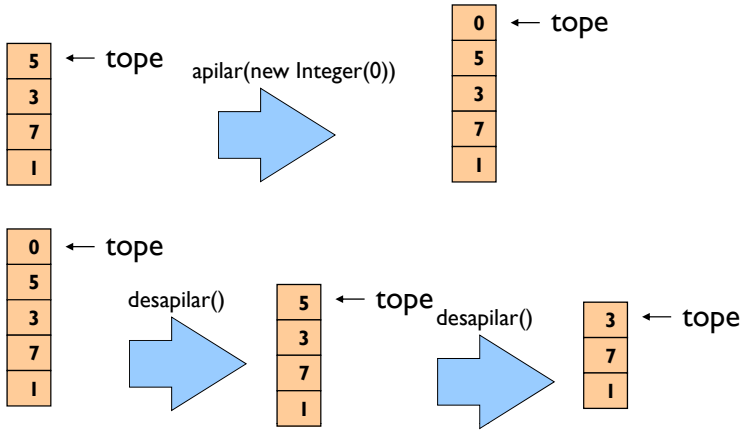
```
package modelos;
public interface Pila<E> {
        void apilar(E x);
        E desapilar();
        E tope();
        boolean esVacia();
    }
```



- Precondición: La ejecución de las operaciones tope() y desapilar() se deben ejecutar únicamente si la Pila no está vacia.
- Al ir insertando los elementos, se van apilando sobre el tope de la pila.

Comportamiento del Modelo Pila

- Pila de Ejemplo:



Uso de la Interfaz Pila: Análisis Sintáctico de Paréntesis

- Dada una expresión parentizada, diseñar un método Java que indique la correspondencia entre paréntesis abiertos y cerrados, así como los paréntesis sin pareja.
- Ejemplo 1: (a*(b+c)+d)

  Introduzca la expresión parentizada: (a*(b+c)+d)

  Paréntesis abierto en pos. 3 se cierra en pos. 7

  Paréntesis abierto en pos. 0 se cierra en pos. 10
- Ejemplo 2: (a+b))

  Introduzca la expresión parentizada: (a+b))

  Paréntesis abierto en pos. 0 se cierra en pos. 4

  Sin pareja el paréntesis cerrado en pos. 5

Análisis Sintáctico de Paréntesis

- Estrategia seguida:
  - Análisis de izquierda a derecha donde cada paréntesis cerrado debe corresponder con el último paréntesis abierto.

```
import modelos.*;
import lineales.*;
public TestParentesis {
   public static void main(String args[]){
      Scanner teclado = new Scanner(System.in);
      System.out.print("Introduzca la expresión parentizada:");
      String expresion = teclado.next();
      System.out.println(correspondenciaParentesis(expresion));
}
…
```


Análisis Sintáctico de Paréntesis (II)

```
private static String correspondenciaParentesis(String e){
   String res = ""; int talla = e.length();
   Pila<Integer> p = new ArrayPila<Integer>();
   for ( int i = 0; i < talla; i++ ) {
      if ( e.charAt(i) == '(' ) p.apilar(new Integer(i));
      else if ( e.charAt(i) == ')' ) {
       if ( !p.esVacia() ){
          res += "Paréntesis abierto en pos. "+ p.desapilar();
          res += " se cierra en la pos. "+i+"\n";
       }
      else res += "Sin pareja el paréntesis cerrado de la posicion "+i;
   }
   while ( !p.esVacia() )
      res += "Sin pareja el paréntesis abierto en pos. "+p.desapilar();
   return res;
   }
}
```

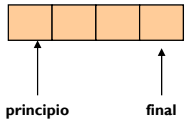
• ¿Cuál es el coste temporal asintótico del método?

Estructura de Datos: Cola

- Una *Cola* es una colección homogenea de elementos en la que:
  1. El acceso se realiza siguiendo un criterio FIFO (**F**irst **I**n **F**irst **O**ut), es decir, el primer elemento que fue insertado es el primero en ser accedido.
- Ejemplos de Colas:
  - Cola para entrar a la UPV por las mañanas.
  - Cola para renovarse el DNI.
  - Colas de trabajo pendientes de impresión.
  - Cola de tareas a ejecutar en un sistema de computación distribuido.

El interfaz Cola

```
package modelos;
public interface Cola<E> {
        void encolar(E x);
        E desencolar();
        E primero();
        boolean esVacia();
    }
```



- Precondición: Los métodos desencolar() y primero(), únicamente deben ser empleados si la cola no está vacía.
- Al igual que en la Pila y la ListaConPI, ningún método utiliza como parámetro las referencias principio y final (se relega a la fase de implementación).

Comportamiento del Modelo Cola

- Cola de Ejemplo: