Estructuras Lineales

Recursividad en recursividad estructural

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

19 de julio de 2021

Código recursivo

Temas

Lista recursiva en Java

Código 1: Lista recursiva

```
1  public class Lista<E> {
2    private E dato;
3    private Lista<E> siguiente;
4
5    /** Construye una lista con un dato, seguida de otra lista. */
6    public Lista(E dato, Lista<E> siguiente) {
7      setDato(dato);
8      this.siguiente = siguiente;
9    }
10
11   public E getDato() { return dato; }
12
13   public Lista<E> getSiguiente() { return siguiente; }
14
15   public void setDato(E dato) {
16     if (dato == null) throw new NullPointerException("No datos nulos.");
17     this.dato = dato;
18   }
19 }
```

Temas

Consumer

- La **interfaz** `Consumer<T>` representa una operación que consume un argumento y no devuelve resultados. Se espera que su efecto sea colateral.

Consumer

Código 2: Lista recursiva

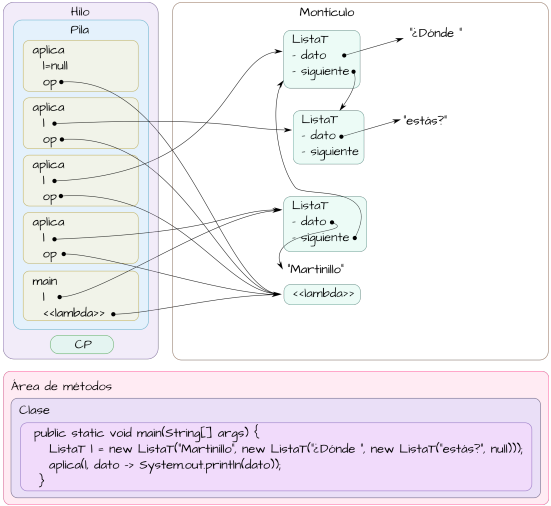
```
1  import java.util.function.Consumer;
2  public class Lista<E> {
3   // ...
4   /** Versión funcional de un método que trabaja con la lista. */
5   public static void imprimeLista(Lista<?> l) {
6     if(l == null) return;            // Caso base, escrito explícitamente.
7     else {
8       System.out.println(l.dato);
9       imprimeLista(l.siguiente);
10    }
11  }
12
13  public static <E> void aplica(Lista<E> l, Consumer<? super E> op) {
14    if(l != null) {
15      op.accept(l.getDato());
16      aplica(l.getSiguiente(), op);
17    }
18  }
19
20  public static void main(String[] args) {
21    Lista<String> l = new Lista<>("Martinillo", new Lista<>("¿Dónde␣", new Lista<>("estás?", null)));
22    aplica(l, dato -> System.out.println(dato));
23  }
24 }
```

Temas

```
1  public static <E> void
2         aplica(ListaT<E> l,
3                Consumer<E> op) {
4    if(l != null) {
5      op.accept(l.getDato());
6      aplica(l.getSiguiente(), op);
7    }
8  }
```

Código 3: Ejecución

```
Martinillo
¿Dónde
estás?
```



Inversión

Temas

Consumer

Código 4: Lista recursiva

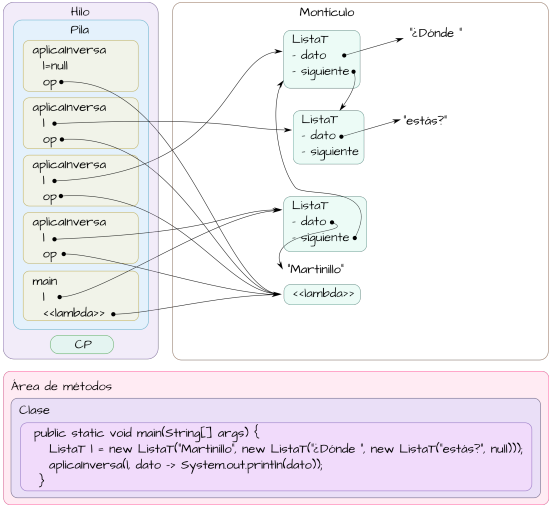
```
1  import java.util.function.Consumer;
2  public class Lista<E> {
3   // ...
4   public static <E> void aplicaInversa(Lista<E> l, Consumer<? super E> op) {
5     if(l != null) {
6       aplicaInversa(l.getSiguiente(), op);
7       op.accept(l.getDato());
8     }
9   }
10
11  public static void main(String[] args) {
12    Lista<String> l = new Lista<>("Martinillo", new Lista<>("¿Dónde␣", new Lista<>("estás?", null)));
13    aplicaInversa(l, dato -> System.out.println(dato));
14  }
15 }
```

Temas

```java
public static <E> void
        aplicaInversa(ListaT<E> l,
                      Consumer<E> op) {
  if(l != null) {
    aplicaInversa(l.getSiguiente(), op);
    op.accept(l.getDato());
  }
}
```

Código 5: Ejecución

```
estás?
¿Dónde
Martinillo
```



Otro ejemplo

Reducción

Código 6: Reduce suma y multiplicación

```
1  public static <E> E reduce(Lista<E> l, BinaryOperator<E> op, E valorInicial) {
2    if (l == null) {
3      return valorInicial;
4    } else {
5      return op.apply(l.getDato(), reduce(l.getSiguiente(), op, valorInicial));
6    }
7  }
8
9  public static void main(String[] args) {
10   Consumer<Object> impresora = dato -> System.out.println(dato);
11
12   System.out.println("Sea la lista: ");
13   Lista<Integer> lint = new Lista<>(-8, new Lista<>(9, new Lista<>(4, null)));
14   aplica(lint, impresora);
15
16   Integer res = reduce(lint, (n1, n2) -> n1 + n2, 0);
17   System.out.println("La suma de sus elementos es: " + res);
18
19   Integer resm = reduce(lint, (n1, n2) -> n1 * n2, 1);
20   System.out.println("La multiplicación de sus elementos es: " + resm);
21 }
```

Código 7: Ejecución

```
Sea la lista:
-8
9
4
La suma de sus elementos es: 5
La multiplicación de sus elementos es: -288
```

Bibliografía

Bibliografía I

Package java.util.function (s.f.). URL: https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html.

Licencia