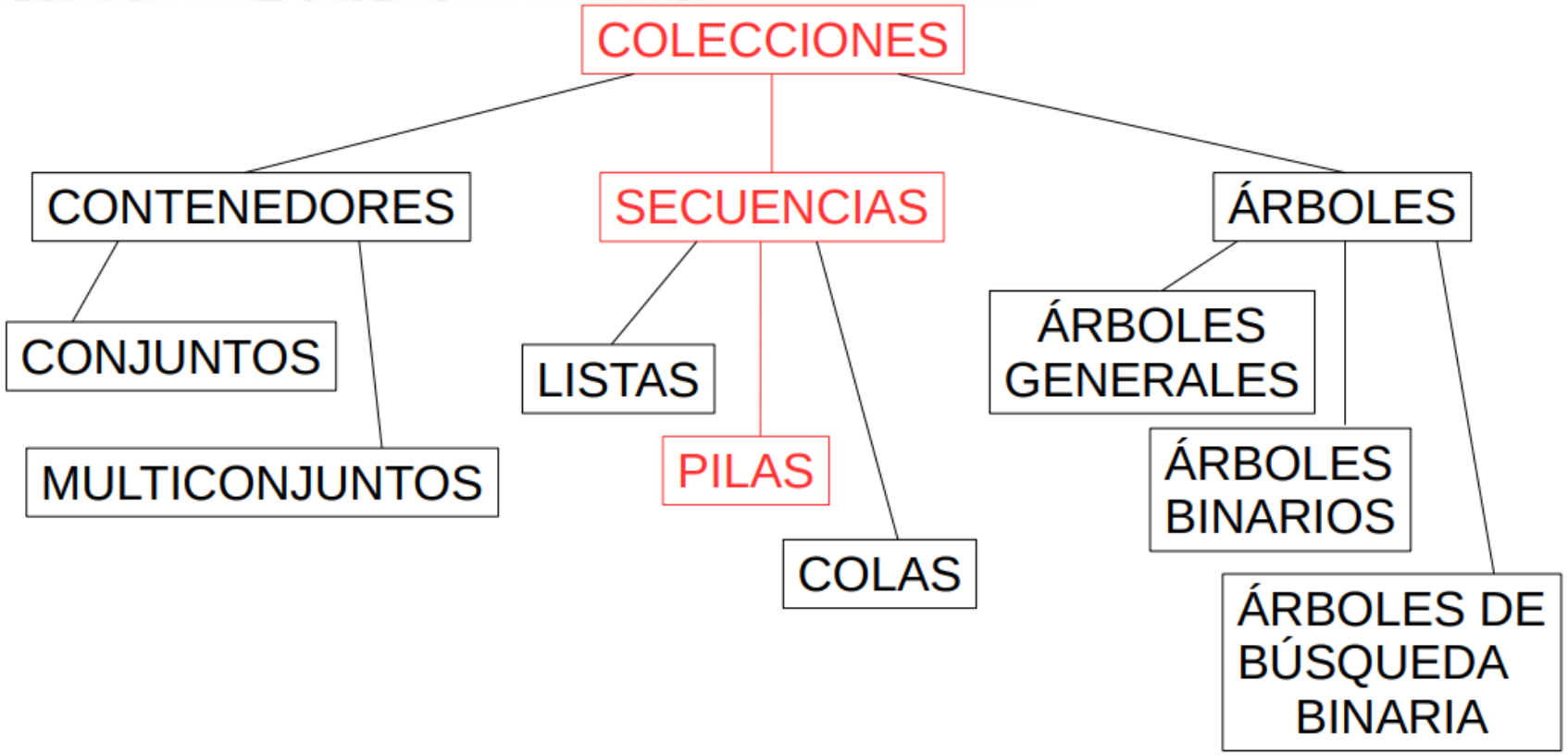
Tema 5 Pilas y Colas

ESTRATEGIAS DE PROGRAMACION Y ESTRUCTURAS DE DATOS

CA Guadalajara (UNED)

Índice

Implementación de Pilas



Pilas

Acceso por un único punto → cima

Se organizan inversamente a su inserción

- Política LIFO (Last In First Out)

No se puede modificar la cima → borrar + insertar

Recorrido destructivo

Operaciones:

- Apilar, Desapilar y consultar cima

Pilas: Constructores

```
public class Stack<E> extends Sequence<E> implements StackIF<E> {

  /* Constructor por defecto: crea una pila vacía */
  public Stack(){ super(); }

  /* Constructor por copia: delega en el constructor por copia *
   * de la secuencia                                           */
  public Stack(Stack<E> s) { super(s); }
```

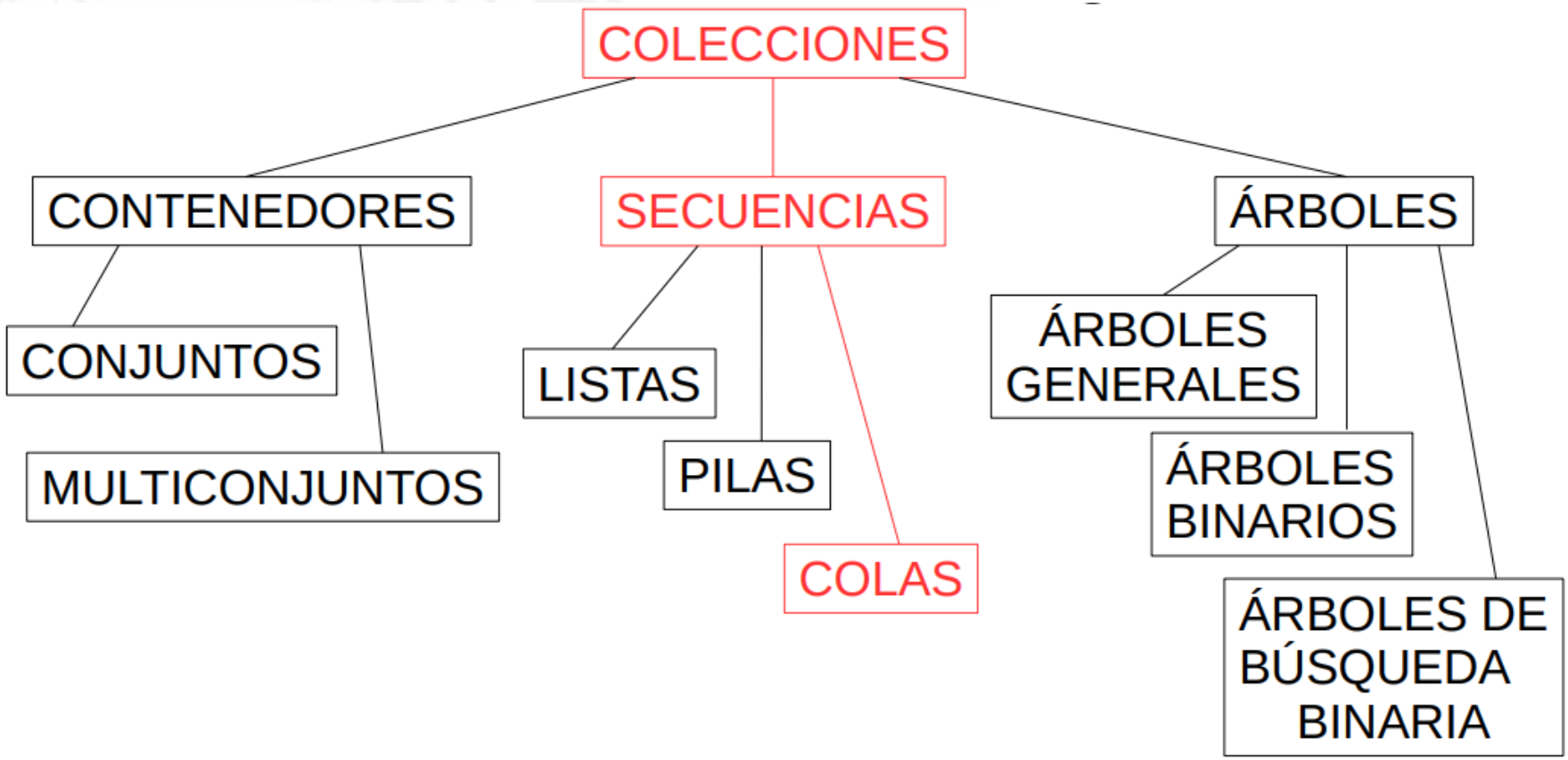
Pilas: Acceso y Modificación

```
/* Devuelve el elemento en la cima de la pila */
public E getTop() { return this.firstNode.getValue(); }

/* Añade un nuevo elemento a la cima de la pila */
public void push(E elem) {
  NodeSequence newNode = new NodeSequence(elem);
  if(!isEmpty()){
    newNode.setNext(this.firstNode);
  }
  this.firstNode = newNode;
  this.size++;
}

/* Elimina el elemento situado en la cima de la pila */
public void pop() {
  this.firstNode = this.firstNode.getNext();
  this.size--;
```

Implementación de Colas



Implementación de Colas

Acceso por dos puntos → inserción y borrado

Se organizan directamente a su inserción

- Política FIFO (First In First Out)

No se puede modificar sus elementos accesibles

- Copiar y destruir TODO el contenido

Recorrido destructivo

Operaciones:

- Encolar, desencolar y consultar primero

Colas: Constructores

```
public class Queue<E> extends Sequence<E> implements QueueIF<E> {

  private NodeSequence lastNode;

  /* Constructor por defecto: crea una cola vacía */
  public Queue(){
    super();                /* Construimos la secuencia vacía */
    this.lastNode = null; /* No hay último nodo */
  }
```

Colas: Constructores

```
/* Constructor por copia: delega en el constructor por copia *
 * de la secuencia                                            */
public Queue(Queue<E> s) {
  super(s);  /* Copiamos la secuencia de la cola original */
  /* Recorremos la secuencia de la cola copia para encontrar */
  /* su último nodo                                          */
  if ( this.isEmpty() ) {
    this.lastNode = null;
  } else {
    NodeSequence node = this.getNode(this.size);
    this.lastNode = node;
  }
}
```

Colas: Operaciones

```
/* Devuelve el primer elemento de la cola */
public E getFirst() { return this.firstNode.getValue(); }

/* Añade un nuevo elemento al final de la cola */
public void enqueue(E elem) {
  NodeSequence newNode = new NodeSequence(elem);
  if(isEmpty()){ this.firstNode = newNode; }
  else{ this.lastNode.setNext(newNode); }
  this.lastNode = newNode;
  this.size++;
}

/* Elimina el primer elemento de la cola */
public void dequeue() {
  this.firstNode = this.firstNode.getNext();
  this.size--;
  if(this.size == 0){ this.lastNode = null; }
}
                                        ...
```

Colas: Operaciones

```
  /* Vacía la cola */
  public void clear() {
    super.clear();          /* Vaciamos la secuencia */
    this.lastNode = null; /* No hay último nodo */
  }
}
```

Tema 5 Pilas y Colas

ESTRATEGIAS DE PROGRAMACION Y ESTRUCTURAS DE DATOS

CA Guadalajara (UNED)