COMP 210: Data Structures

ArrayList and LinkedList

Java's `ArrayList` and `LinkedList` classes both implement the `List` interface.

ArrayList of References

An ArrayList is always a list of references to objects, not a list of actual object.

```
Index   ArrayList       Objects


        ---------        -------------
  0     |    *--|--------|  object 1 |
        ---------        -------------
  1     |    *--|----------->  object 2 somewhere else
        ---------
  2     |    *--|----> object 3 somewhere different
        ---------
  3     |    *--|-------> object 4 yet another place
        ---------        -------------
  4     |    *--|--------|  object 5 |
        ---------        -------------
  5     |       |
        ---------
  6     |       |
        ---------
  .         .
  .         .
  .         .
```

The most common code pattern for stepping through an ArrayList is:

```
for ( MyObject obj : myList )
{
    obj.doSomething();
}
```

or the older for loop style:

```
for ( int i = 0; i < myList.size(); i++ )
{
    MyObject obj = myList.getElement(i);
    obj.doSomething();
}
```

LinkedList of Nodes

A LinkedList is a list of nodes, where each node contains a reference to an object representing an element in the list and a reference (or *pointer*) to the next node in the list.

```
start
 |    ____________    ____________    ____________    ____________
 ---->|     |  --|--->|     |  --|--->|     |  --|--->|     |  / |
      |  *  |    |    |  *  |    |    |  *  |    |    |  *  | /  |
      ---|--------    ---|--------    ---|--------    ---|--------
         |               |               |               |
         |  _________    |  _________    |  _________    |  _________
         -> | obj 1 |    -> | obj 2 |    -> | obj 3 |    -> | obj 4 |
            ---------       ---------       ---------       ---------
```

Each LinkedList Node is a *tuple* (pairing) of (1) a reference to an object and (2) a reference to the next node in the list, often called a *pointer*. The last node in the list has a `null` reference as its “next” node.

The most common code pattern for stepping through a LinkedList is:

```
for ( MyObject obj : myList )
{
    obj.doSomething();
}
```

Note that the for-each style loop is the same for a Java `ArrayList` or `LinkedList`. You can use this type of loop for any collection that implements the `Iterable` interface, as both `ArrayList` and `LinkedList` do.

Another diagram

```
                                            _________
                                         -->| obj 3 |
                                         |  ---------
                      ____________       |
                   -->|     |  --|--     |
start              |  |  *  |    | |     |
 |    ____________ |  ---|-------- |     |            ____________
 ---->|     |  --|-|     |         |     |         -->|     |  / |
      |  *  |    |       |         |     |         |  |  *  | /  |
      ---|--------       |         |  ___|________ |  ---|--------
         |               |         -->|  |  |  --|-|     |
         |  _________    |            |  *  |    |       |  _________
         -> | obj 1 |    |            ------------       -> | obj 4 |
            ---------    |                                  ---------
```

```
|   _________
-> | obj 2 |
    ---------
```

Although we often draw neat diagrams of linked lists like the first one above, the nodes and the objects may be placed anywhere in memory. Each reference specifies where a node's object can be found or where the next node can be found.

Alyce Brady, 27 September, 2025