CO452
Programming Concepts

Lecture 6

Arrays and Strings

Differences between collections

In this lecture we'll explore some of the differences between the following data structures, and also discuss the String class in more detail

- List
- LinkedList
- ArrayList
- Arrays

List

Java’s List

A List describes a collection of objects which are ordered.

In Java, the List is built as an interface, which cannot be instantiated (as it’s not a class).

However, List contains abstract forms of methods which are overridden in the ArrayList and LinkedList classes, providing the implementation (method bodies) which enable the behaviour of the data structure.

LinkedList

Linked List Introduction

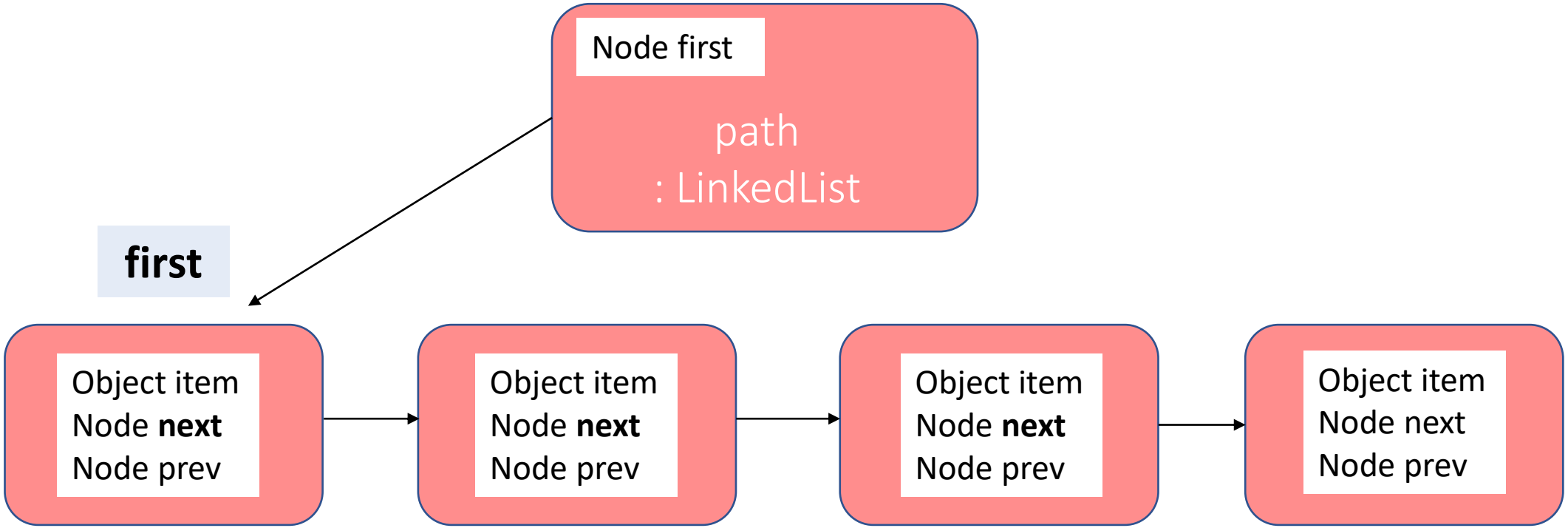
An instantiation of a singly Linked List will typically hold a pointer to the first object in the list.

The first object in the list then holds a pointer to the next object in the list and that object points to the next and so on. Therefore, objects are accessed sequentially starting from the first object (in a singly Linked List).

Objects are appended (added to the end; the last) or prepended (added to start of the list; the first).

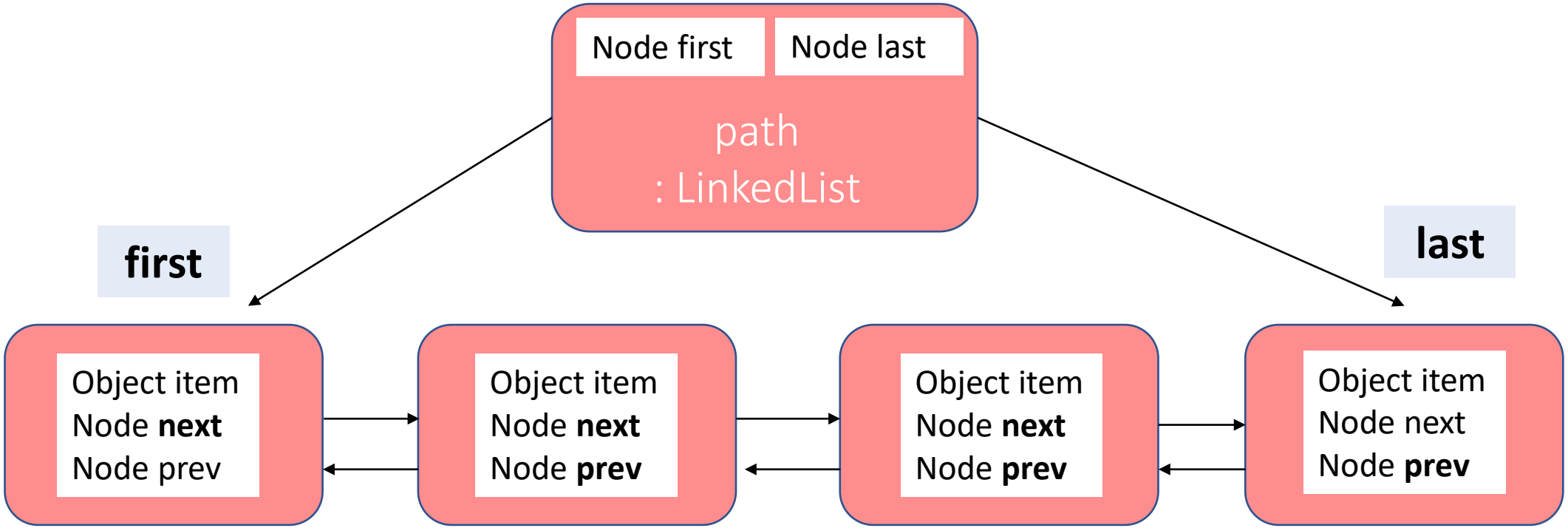
Visualisation of a (singly) LinkedList

A singly linked list would a pointer to the first item
and the individual nodes point to the next node in sequence



Visualisation of a doubly LinkedList

Java's LinkedList is also an example of a doubly linked list where objects also hold pointers to the previous object as well as next



ArrayList

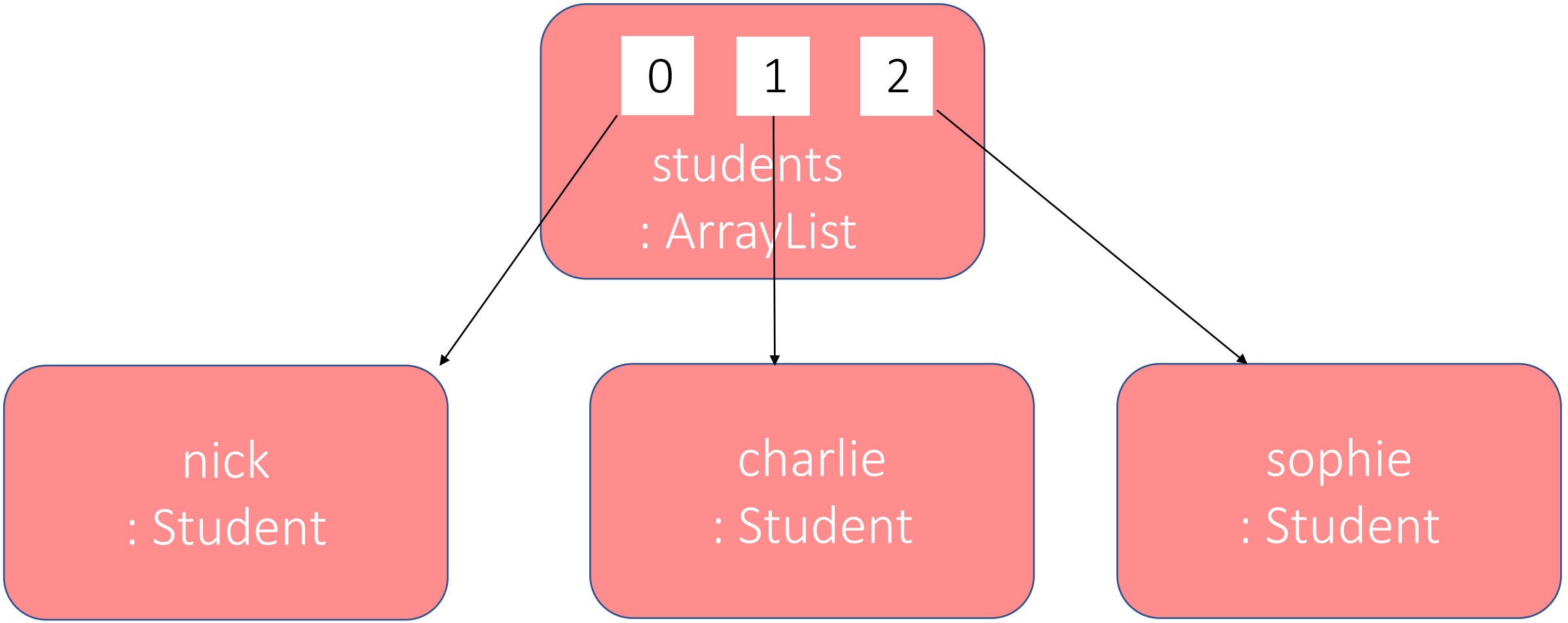
Recap on ArrayList

An instantiation of an ArrayList allows objects to be added (appended) to the end of the list.

Objects added do have an index position.

Can find objects by iterating through the list, or if the index is known, can be accessed directly through the index position.

Visualisation of an ArrayList



Array

Array

The items in an array are called **elements.**

We specify how many elements an array will have when we declare the **size** of the array (if '**fixed-size**'), unlike flexible sized collections (ArrayList).

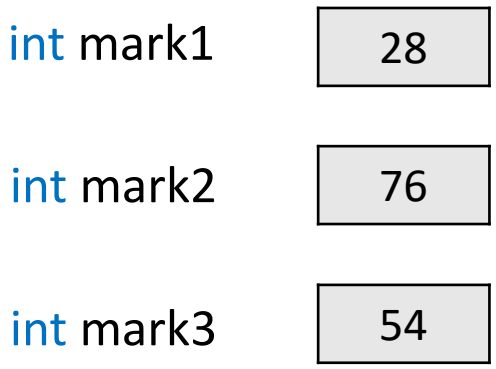
Elements are numbered and can referred to by number inside the [ ] is called the **index.** This is used when data is input and output.

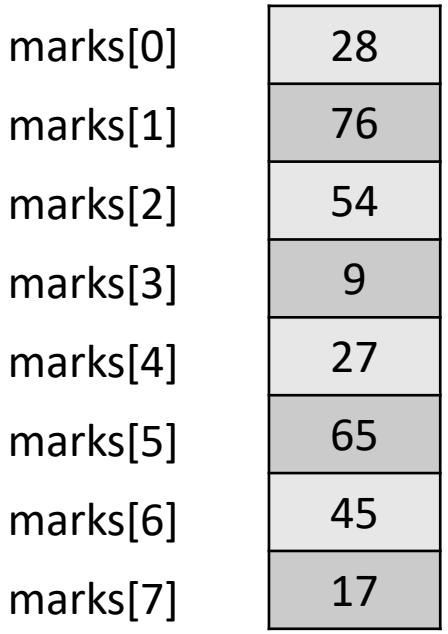
Can only store data if it **matches the type** the array is declared with.

Visualisation of an Array

An Array is a structure that can hold multiple values in individual elements (positions)

```
int[] marks = new int[8];
```





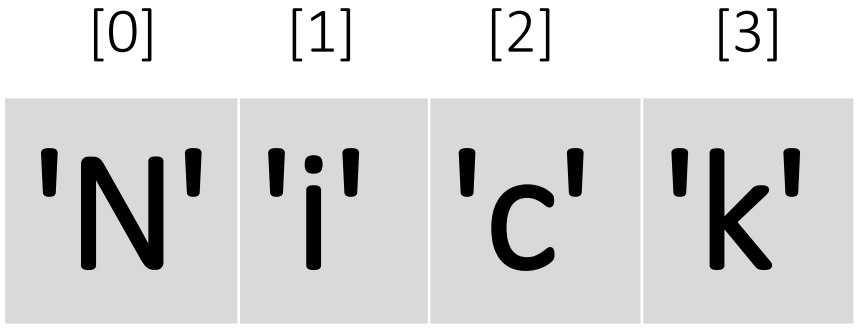
Strings

A String object is an array

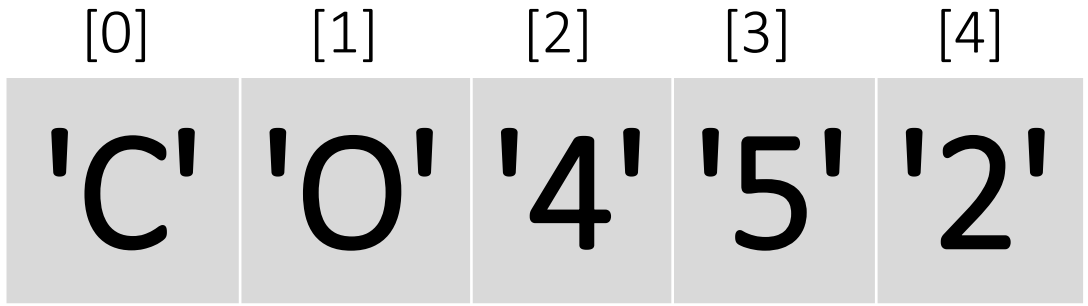
A String object is an **immutable array of characters**.

Each character has a numbered position in the array (index):

String name = "Nick";



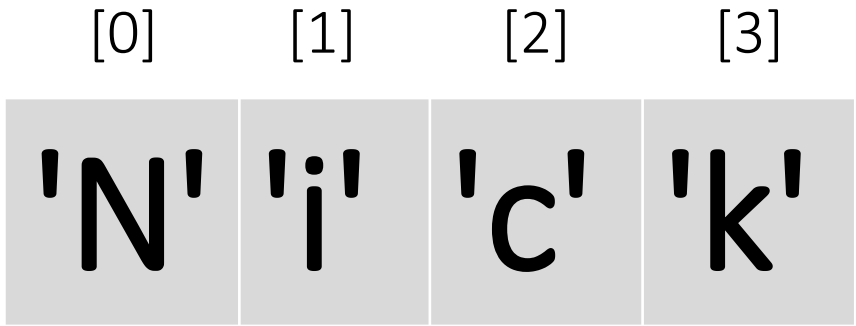
String code = "CO452";



Referring to characters in a String

You can refer to letters of a String through the index value. In Java you can pass the index value as a parameter to the method **charAt()**

String name = "Nick";



System.out.println(name.**charAt**(0)); // displays 'N'

String methods

Reminder on the equals() method

Whilst the equality operator (==) can be applied to primitive data (int, char, boolean), Strings are classes, so **the equality operator would compare memory addresses of String objects** rather than the values stored in each object

Use the method **equals** to compare the values stored at String variables rather than comparing memory addresses

```
if(name.equals("Nick"))
```

toUpperCase() and toLowerCase()

Methods which remove ‘casing’ can make validation easier when performing comparisons:

```
String name = "Nick";
System.out.println(name.toUpperCase());     // NICK
System.out.println(name.toLowerCase());      // nick
```

contains()

The contains method of the String library can be called to evaluate whether a part of one String exists in another. Be aware though, that the comparison is case sensitive.

```
String name = "Nick Day";

System.out.println(name.contains("Day"));       // true
System.out.println(name.contains("day"));       // false (case sensitive)

String title = "Programming Concepts";

System.out.println(title.contains("min"));      // true
```

ASCII

Under the American Standard Code for Information Interchange (ASCII), each letter, number and symbol on a keyboard has a decimal integer value. These can be used to evaluate and compare characters.

0-9 keys : ASCII decimal values : 48 - 57

A-Z keys : ASCII decimal values : 65 - 90

a-z keys : ASCII decimal values : 97 - 122

Other useful String methods

compareTo()

hashCode()

indexOf()

join()

length()

split()

trim()

valueOf()

Full documentation available at:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html