CO452 Programming Concepts

Lecture 5

Revision/Consolidation Lecture

In this revision lecture

- We shall recap over the key points of the past four weeks
- These concepts are **fundamental**
- You will be tested on these concepts in your TCA.

Week 1

Week 1

- Writing our First Program (Hello World)
- Declaring Variables
- Assigning values to variables
- Constants
- Input: InputReader
- Output
- GitHub

Question!

What is a computer?

The println() method

Use the **println()** method of the System class to output the text data placed within speech marks " " to screen

```
public class Program
{
    public static void main(String[] args)
    {
        System.out.println("Hello World");
    }
}
```

```
Hello World

_
```

How do we define variables?

1. First, we have to define the type of data that we want to store – **data type**

2. Then we have to select a **meaningful name** which represents that data well

```
String  name;
```

1. Data types

int	whole numbers e.g. 10, 75, 200
double	decimal numbers e.g. 2.56, 0.314, 20.75
String	a **class** that represents more than one letter
char	single letter or number or symbol
boolean	true or false

Input example: String data

We can use the getString method of InputReader to capture data from the keyboard.

```
public class Program
{
    public static void main(String[] args)
    {
        String name;
        System.out.println("Hello, what’s your name? ");
        name = InputReader.getString();
        System.out.println("Hello " + name + "!");
    }
}
```

```
Hello, what’s your name?

_
```

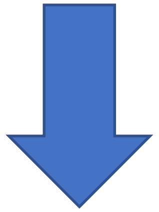
```
Hello, what’s your name?
Nick_
```

```
Hello, what’s your name?
Nick
Hello Nick!
_
```

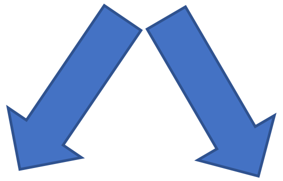
Week 2

Sequence, Selection and Iteration

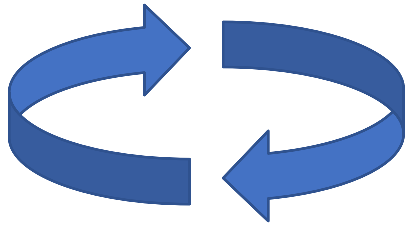
❖**Sequence** mandates that statements be executed in order (line by line)



❖**Selection** (conditional) statements will execute a **block of code once** when the condition is true



❖**Iteration** allows us to **repeat** statements within a block **whilst** the condition is **true**



If and else statement

The else block executes **if** the evaluation is **false**

```
if(mark >= 0 && mark <= 100)
{
   System.out.println("This is a valid mark");
}
else
{
    System.out.println("This is an invalid mark");
}
```

for loop

The for loop has three parts:

(1) variable initialisation (2) condition (3) increment

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed " + (count+1) + " times");
}
```

for loop

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Initialise count to 0

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Is 0 < 3 ?

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed… ");
}
```

Yes! (true) Therefore execute code in braces

for loop

After executing code in braces, increment count

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

count now has value of 1

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

We don’t re-initialise count back to 0!

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Is 1 < 3 ?

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed… ");
}
```

Yes! (true) Therefore execute code in braces

for loop

After executing code in braces, increment count

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Count now has the value of 2

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Is 2 < 3 ?

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed… ");
}
```

Yes! (true) Therefore execute code in braces

for loop

After executing code in braces, increment count

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Count now has
the value of 3

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

Is 3 < 3 ?

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed... ");
}
```

for loop

```
for(int count = 0; count < 3; count++)
{
  System.out.println("This loop has executed… ");
}
//continue with program…
```

No! (false) Therefore end loop and continue program

Week 3

Classes and objects

Classes are the **blueprint**

Unique objects can be created from this blueprint

Two parts of a class

Variables

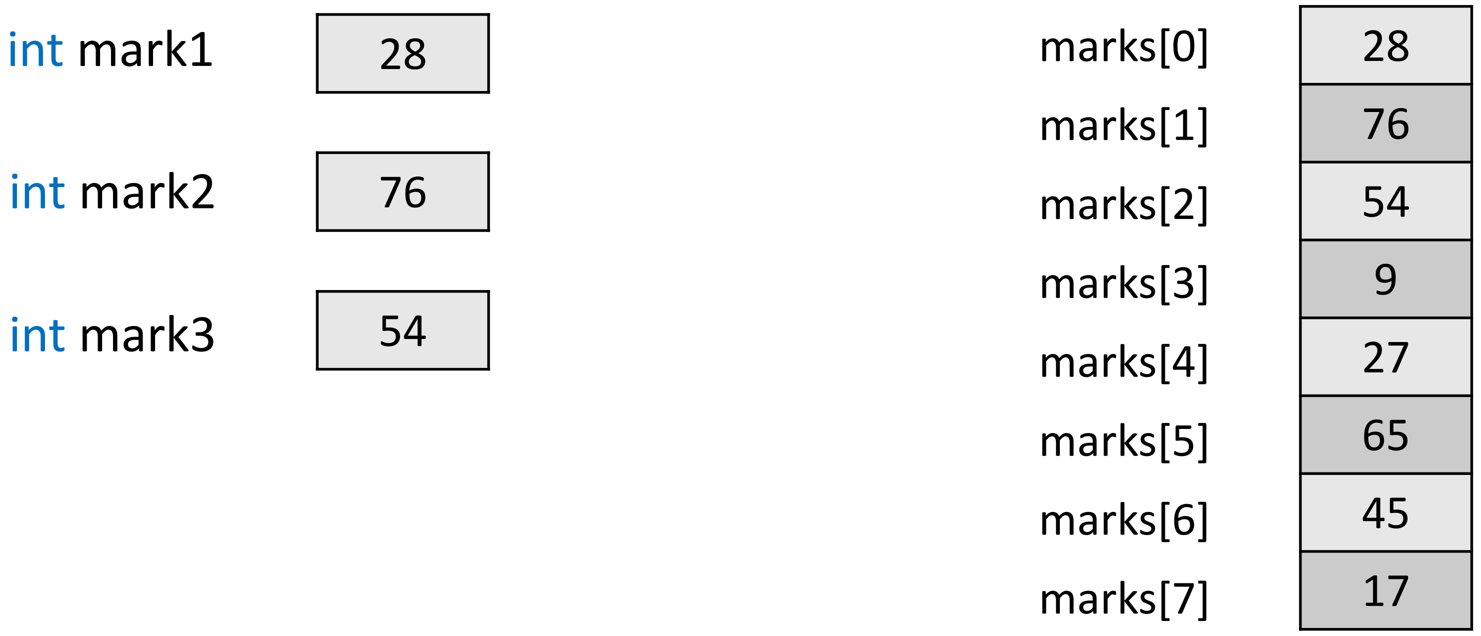
Methods

Week 4

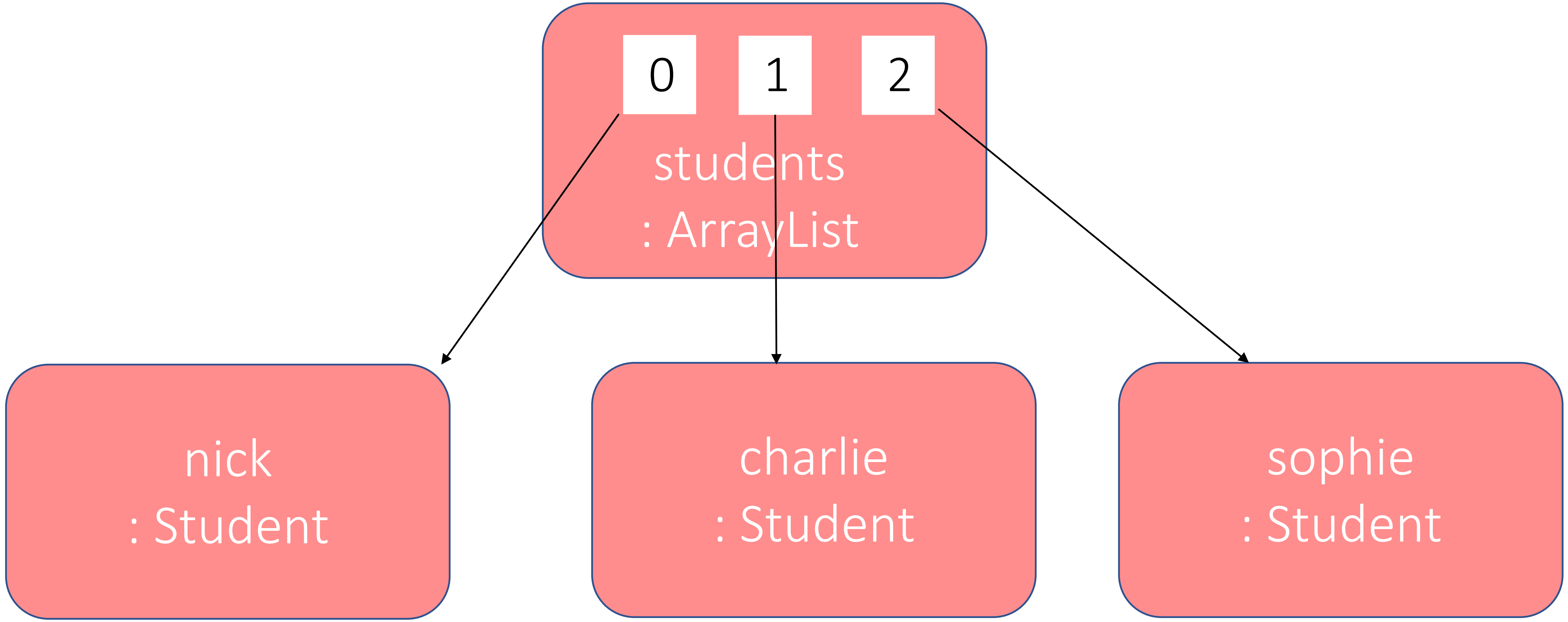
Visualisation of an Array

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

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

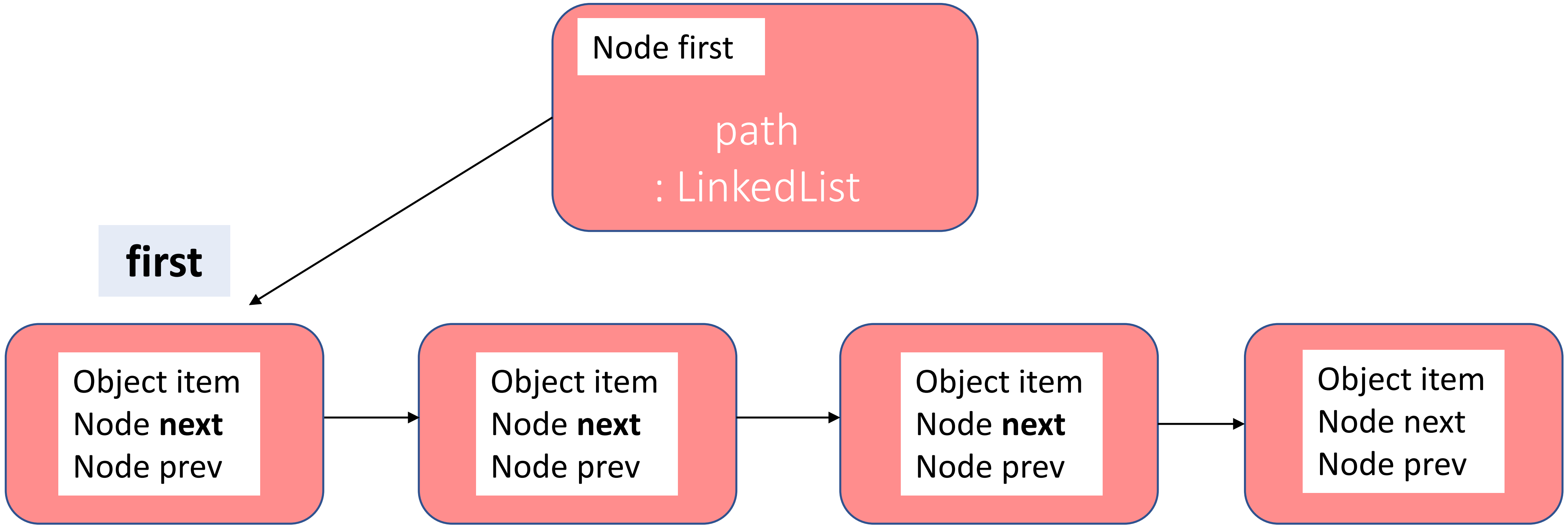


ArrayLists


0
1
2
students
: ArrayList
nick
: Student
charlie
: Student
sophie
: Student

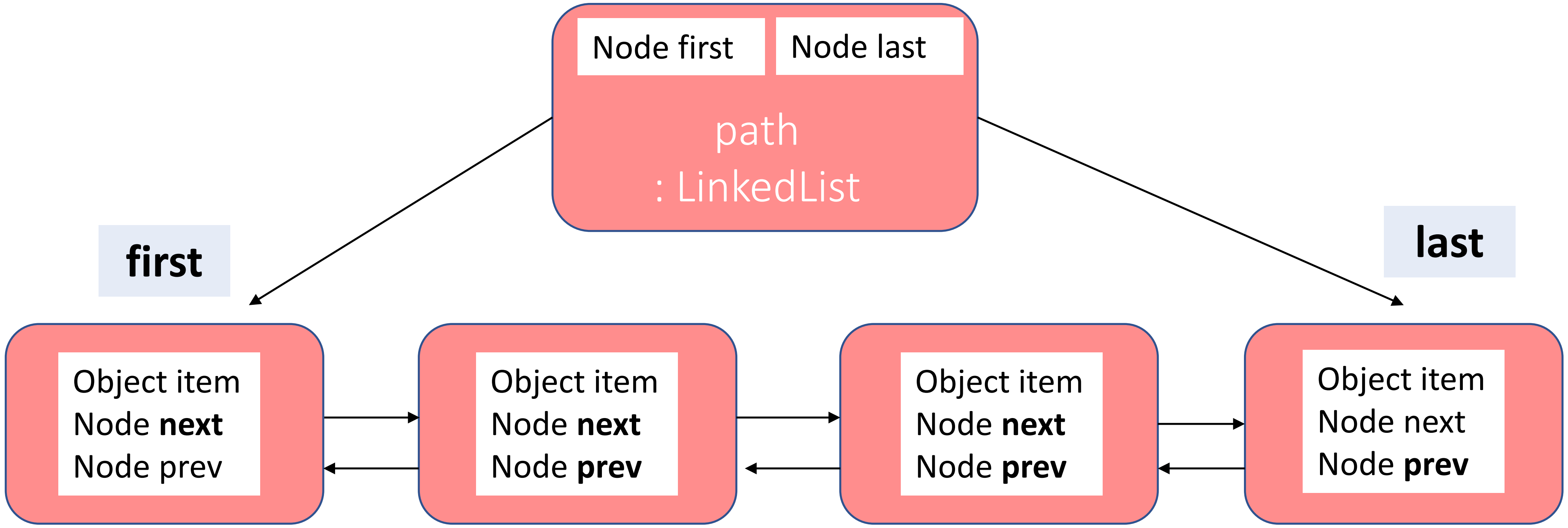
Visualisation of a (singly) LinkedList

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



Node first
Node last
path
: LinkedList
first
last
Object item
Node **next**
Node prev
Object item
Node **next**
Node **prev**
Object item
Node **next**
Node **prev**
Object item
Node next
Node **prev**