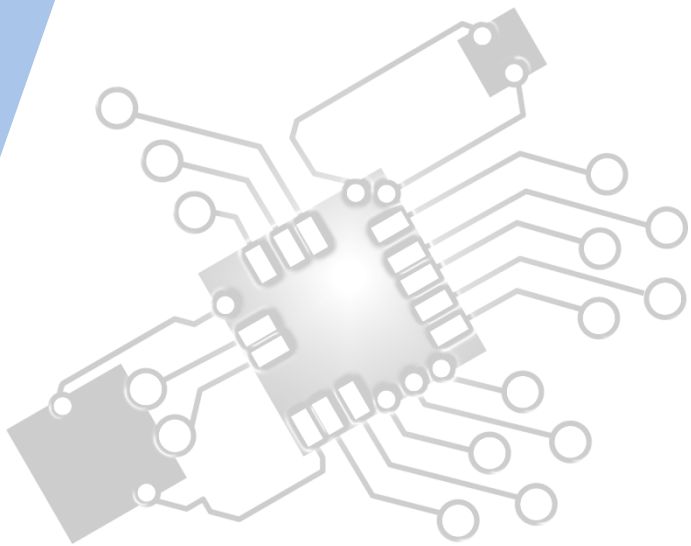
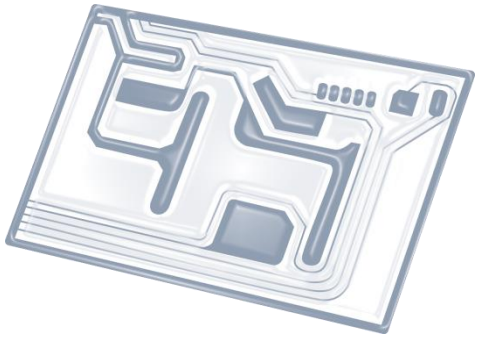
Advanced program development

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL only D.4 Overview

D.4 Advanced program development
D.4.1 Define the term recursion
D.4.2 Describe the application of recursive algorithms
D.4.3 Construct algorithms that use recursion
D.4.4 Trace recursive algorithms
D.4.5 Define the term object reference
D.4.6 Construct algorithms that use reference mechanisms
D.4.7 Identify the features of the abstract data type (ADT) list
D.4.8 Describe applications of lists
D.4.9 Construct algorithms using a static implementation of a list
D.4.10 Construct list algorithms using object references
D.4.11 Construct algorithms using the standard library collections included in JETS
D.4.12 Trace algorithms using the implementations described in assessment statements D.4.9–D.4.11.
D.4.13 Explain the advantages of using library collections
D.4.14 Outline the features of ADT's stack, queue and binary tree
D.4.15 Explain the importance of style and naming conventions in code

1: System design

2: Computer Organisation

3: Networks

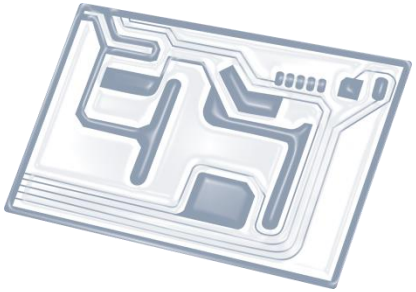
4: Computational thinking

5: Abstract data structures

6: Resource management

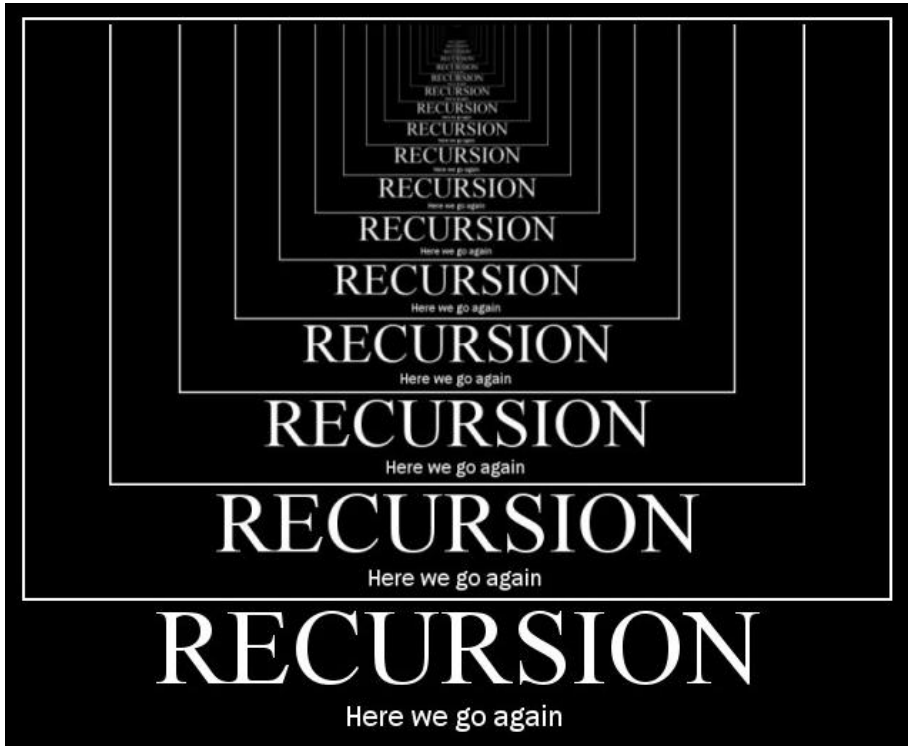
7: Control

D: OOP



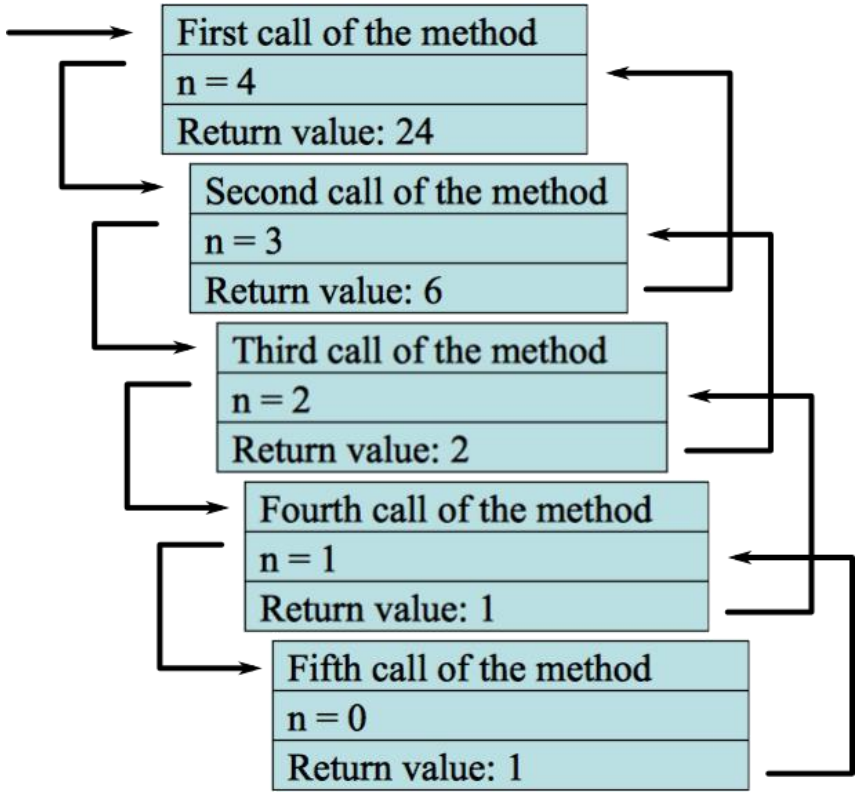
Topic D.4.1

Define the term: **recursion**



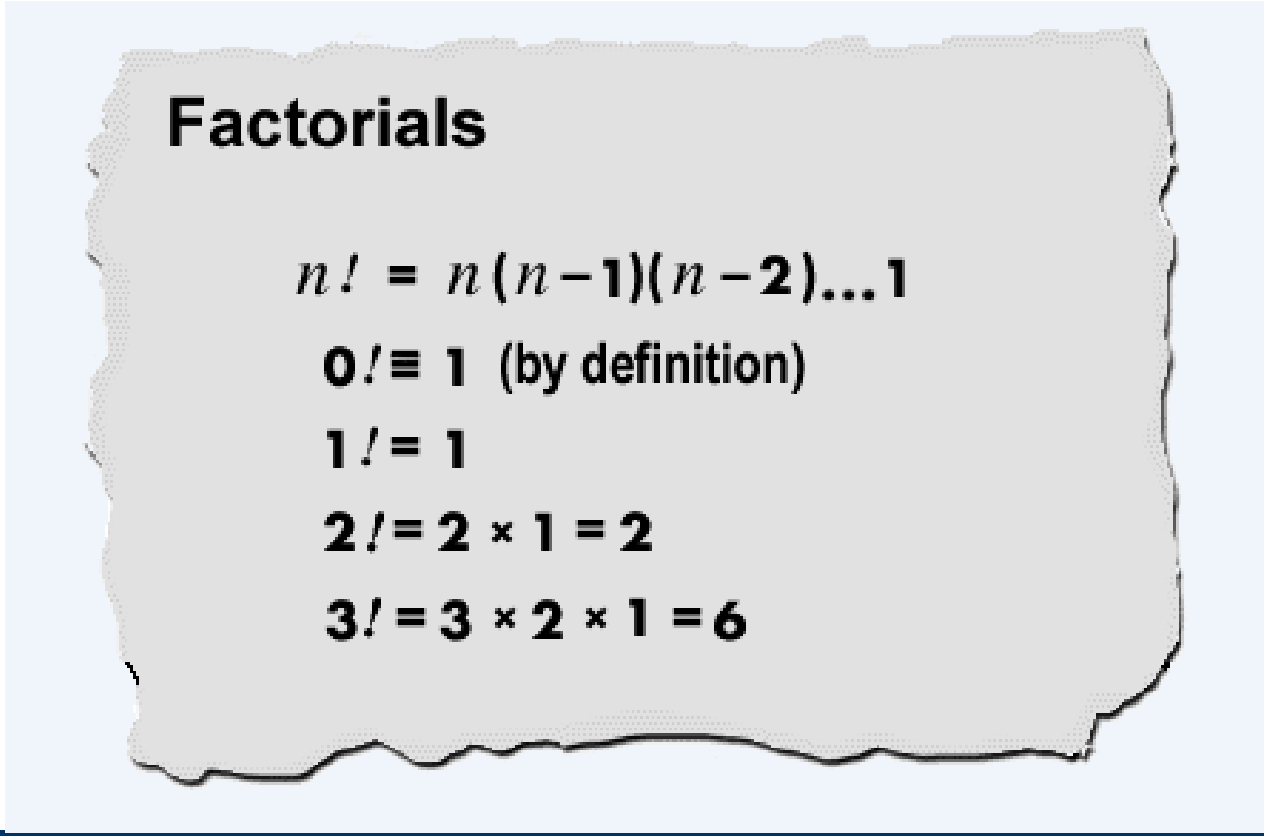
Recursion

- Recursion is a process in which a function **calls itself** as a subroutine.
- This allows the function to be repeated several times, since it calls itself during its execution.

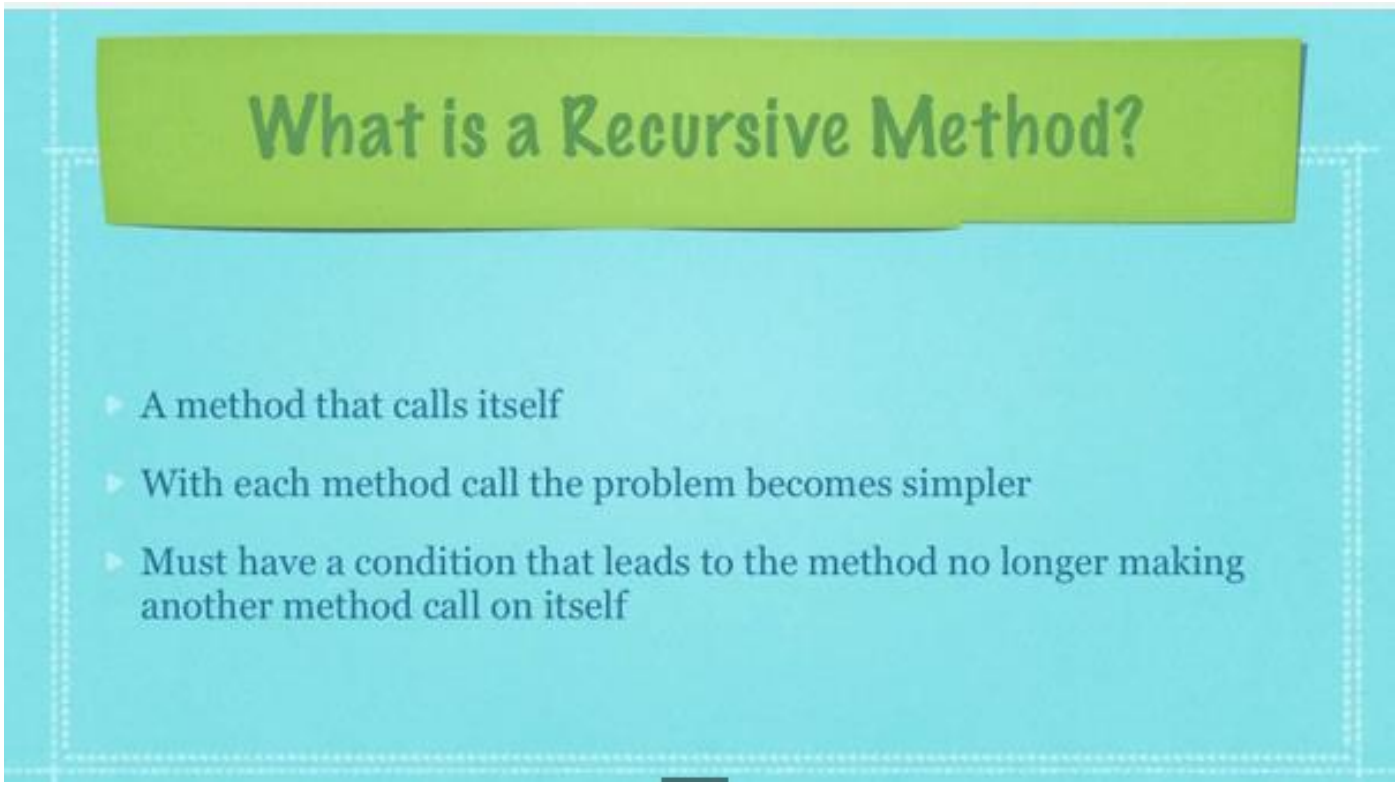


The Classic Example: n!

- **n!** calculates the factorial of an integer
- E.g. **4! = 4 x 3 x 2 x 1 = 24**



Videos to watch on YouTube



https://www.youtube.com/watch?v=Mv9NEXX1VHc

What will this do?

```
public static int factorial(int N)
{
    if (N == 1)
       {
            return 1;
       }
    return N * factorial(N-1);
}
```

Another example

```
class Factorial {

      int fact(int n) {

            int result;

      if ( n ==1) return 1;

      result = fact (n-1) * n;

      return result;
      }
}
```

Task:

Find **advantages** and **disadvantages** to recursion

Possible answer

- Recursive versions of many routines **may execute a bit more slowly** than the iterative equivalent because of the added overhead of the additional function calls.
- Many recursive calls to a method could cause a **stack overrun**. Because storage for parameters and local variables, it is possible that the stack could be exhausted. If this occurs, the java run-time system will cause an exception. However, you probably will not have to worry about this unless a recursive routine runs wild.
- The main advantage to recursive methods is that they can be **used to create clearer and simpler versions of several algorithms than can their iterative relatives**. For example, the QuickSort **sorting** algorithm is quite difficult to implement in an iterative way.

Possible exam type questions

A large company might have several hundred buses running. Each one has a unique id stored with the `Bus` instance.

(d) Explain how a binary tree could be used to store these ids such that they can be quickly retrieved (if they exist) by a search. *[3]*

The tree stores the ids 2045, 3474, 5877, 1099, 9644.

(e) Draw a diagram of an ordered binary tree containing these keys assuming they were inserted in the order given. *[5]*

A binary tree node may be inserted iteratively or recursively.

(f) Identify **two** disadvantages of the recursive algorithm. *[2]*