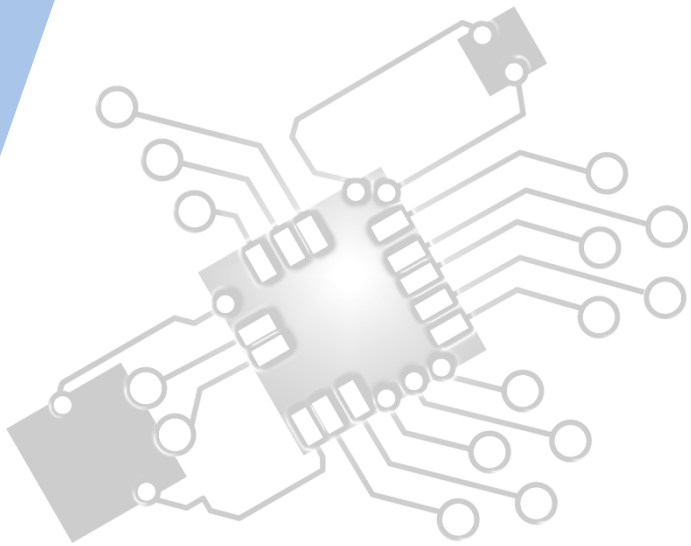
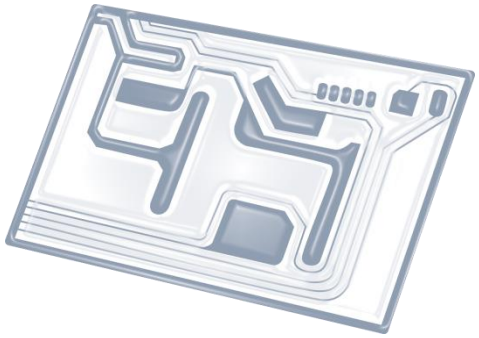
Computational thinking, problem-solving and programming: Introduction to programming

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 & SL 4.3 Overview

Nature of programming languages

4.3.1 State the fundamental operations of a computer

4.3.2 Distinguish between fundamental and compound operations of a computer

4.3.3 Explain the essential features of a computer language

4.3.4 Explain the need for higher level languages

4.3.5 Outline the need for a translation process from a higher level language to machine executable code

Use of programming languages

4.3.6 Define the terms: variable, constant, operator, object

4.3.7 Define the operators =, ., <, <=, >, >=, mod, div

4.3.8 Analyse the use of variables, constants and operators in algorithms

4.3.9 Construct algorithms using loops, branching

4.3.10 Describe the characteristics and applications of a collection

4.3.11 Construct algorithms using the access methods of a collection

4.3.12 Discuss the need for sub-programmes and collections within programmed solutions

4.3.13 Construct algorithms using predefined sub-programmes, one-dimensional arrays and/or collections


1: System design

2: Computer Organisation



3: Networks

4: Computational thinking


5: Abstract data structures

6: Resource management



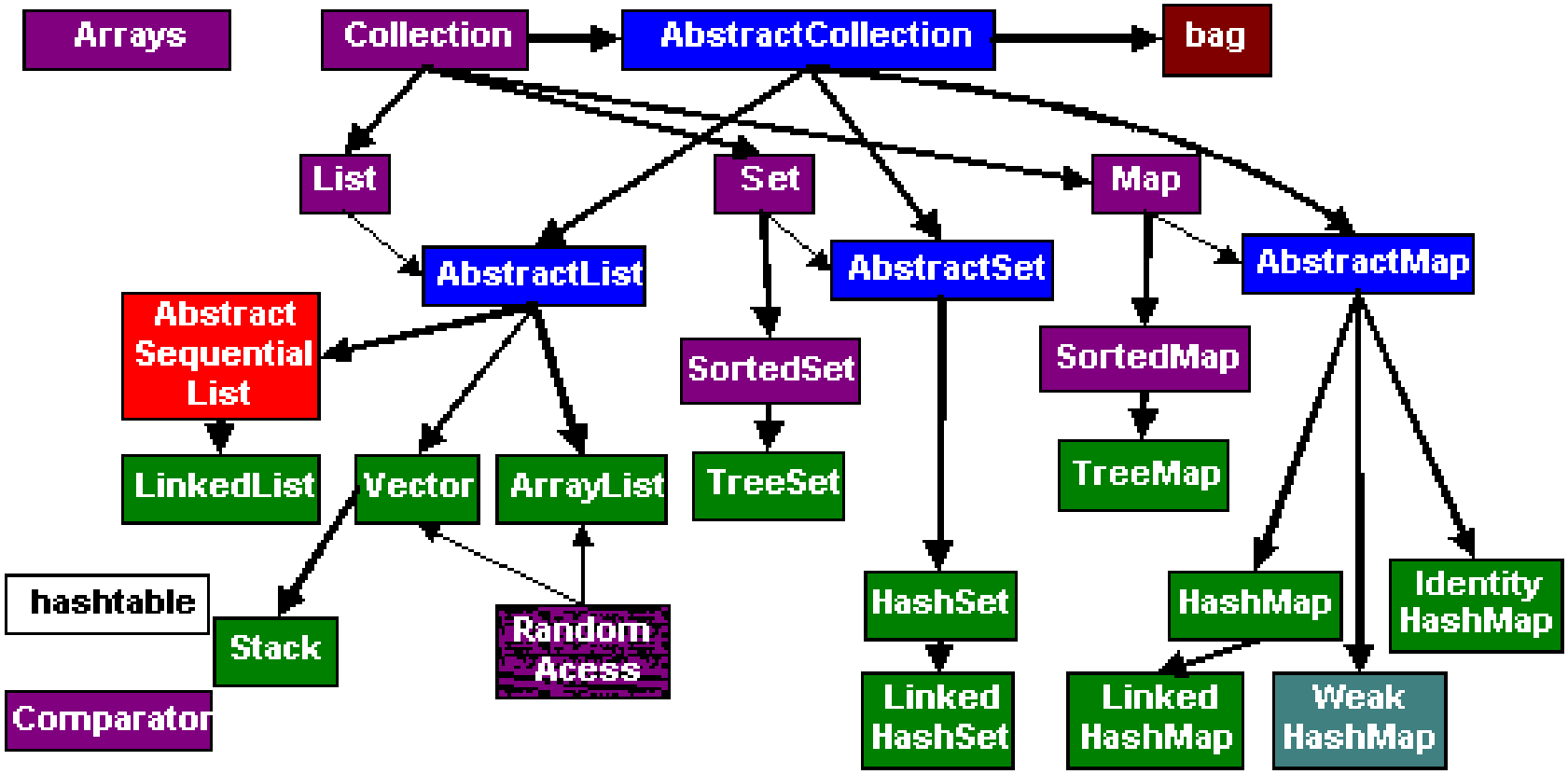
7: Control

D: OOP


Topic 4.3.10

Describe the **characteristics** and **applications** of a **collection**

In **'real life' Java**, there are **many** types of collections.
In **IB land Java**, think of a collection as a **Linked List** with an unknown size/length.



Important: the order is not known

A collection is like a linked-list, but the order of elements is not guaranteed.

Collection methods in **Pseudocode** are:

- `.addItem( new data item )`
- `.resetNext( )` start at beginning of list
- `.hasNext( )` checks whether there are still more items in the list
- `.getNext( )` retrieve the next item in the list
- `.isEmpty( )` check whether the list is empty

Example of collection in Pseudo code

```
NAMES = new Collection()

NAMES.addItem("Bob")
NAMES.addItem("Dave")
NAMES.addItem("Betty")
NAMES.addItem("Kim")
NAMES.addItem("Debbie")
NAMES.addItem("Lucy")

NAMES.resetNext()

output "These names start with D"

loop while NAMES.hasNext()
    NAME = NAMES.getNext()
    if firstLetter(NAME) = "D" then
      output NAME
    end if
end loop

method firstLetter(s)
   return s.substring(0,1)
end method
```

Good idea:

Get lots of practice using the online pseudo code engine and looking at Java tasks you've done involving LinkedLists

Some applications for lists

- Useful for group of items when you **don't know how many** items you'll be needing/using (**contrast** to **arrays** where the size is set in stone at creation)
- Because the collection is only as big as you need it to be, it is an **efficient use of RAM** (memory)
- Can be of **any data type** (primitive or even your own object)

Extra reading that explains when to use which type of collection:
http://docs.oracle.com/javase/tutorial/collections/interfaces/index.html