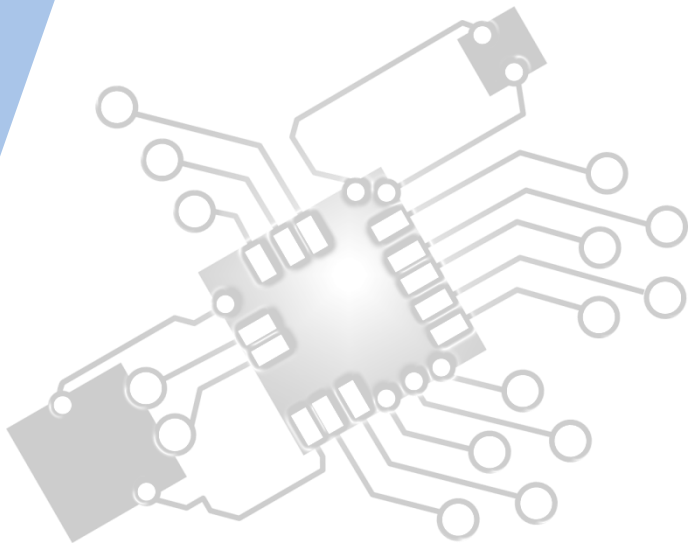
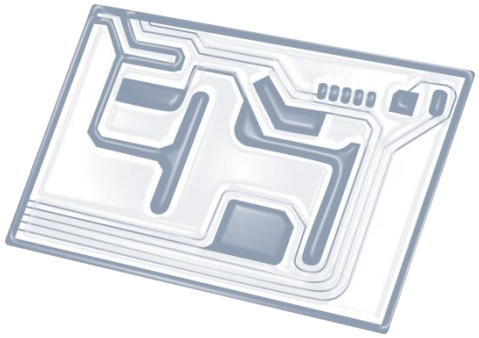
Computational thinking, problem-solving and programming: General Principals

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 4.1 Overview

Thinking procedurally

4.1.1 Identify the procedure appropriate to solving a problem

4.1.2 Evaluate whether the order in which activities are undertaken will result in the required outcome

4.1.3 Explain the role of sub-procedures in solving a problem

Thinking logically

4.1.4 Identify when decision-making is required in a specified situation

4.1.5 Identify the decisions required for the solution to a specified problem

4.1.6 Identify the condition associated with a given decision in a specified problem

4.1.7 Explain the relationship between the decisions and conditions of a system

4.1.8 Deduce logical rules for real-world situations

Thinking ahead

4.1.9 Identify the inputs and outputs required in a solution

4.1.10 Identify pre-planning in a suggested problem and solution

4.1.11 Explain the need for pre-conditions when executing an algorithm

4.1.12 Outline the pre- and post-conditions to a specified problem

4.1.13 Identify exceptions that need to be considered in a specified problem solution

Thinking concurrently

4.1.14 Identify the parts of a solution that could be implemented concurrently

4.1.15 Describe how concurrent processing can be used to solve a problem

4.1.16 Evaluate the decision to use concurrent processing in solving a problem

Thinking abstractly

4.1.17 Identify examples of abstraction

4.1.18 Explain why abstraction is required in the derivation of computational solutions for a specified situation

4.1.19 Construct an abstraction from a specified situation

4.1.20 Distinguish between a real-world entity and its abstraction

1: System design

2: Computer Organisation

3: Networks

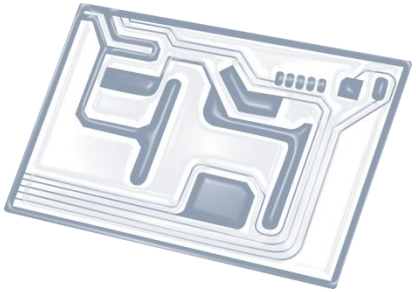
4: Computational thinking

5: Abstract data structures

6: Resource management

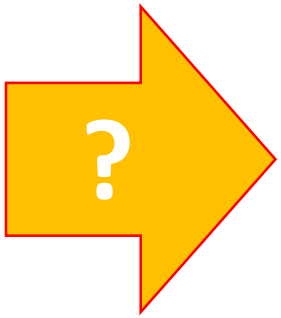
7: Control

D: OOP



Topic 4.1.18

Explain **why** **abstraction** is **required** in the **derivation** of computational solutions for a specified solution



Why use abstraction?

- *Abstraction* can be viewed both as a *process* and as an *entity*.
- *Abstraction* enables a person to concentrate on the essential aspects of the problem on hand, while ignoring details that tend to be distracting.
- The real world is sufficiently complex and presents to many items simultaneously to be dealt with.
- To solve problems well, we need to conquer this complexity.
- *Abstraction* is a convenient way to deal with complexity.

Lesson 5 - The Pillars of Object Orientated Modeling - Abstraction

Why Do We Need Abstraction ?

We have just seen that *abstraction* can be viewed both as a *process* and as an *entity*. Abstraction enables a person to concentrate on the essential aspects of the problem on hand, while ignoring details that tend to be distracting. Real world is sufficiently complex and presents to many items simultaneously to be dealt with. To solve problems well, we need to conquer this complexity. Abstraction is a convenient way to deal with complexity.

Consider maps in an atlas. The maps indicate various levels of detail. The world map, for instance, depicts the various countries, their location (in the latitude & longitude), the oceans, the major seas, international boundaries, and perhaps some capital cities. The map of a country may be available as a "political map" which may show the various states of the country, the borders, the capital cities of each state, etc. The "rivers" map of a country may show the major rivers and tributaries that flow the country. Here the rivers are their focus. Other details are suppressed. Thus, we can see that a map for a particular purpose shows the details required, while suppressing the unnecessary details. A road map, on the other hand, shows details of highways, major roads, motorable roads, and location of service stations and hotels. Of course, there are several other details that may have been given. For example, we could have depicted the nature of flora and fauna in specific places, areas of interest in tourism, climatic details, etc. But these tend to distract the user of a road map from essential details, and hence are ignored. Why should we ignore them if they are also important (in some context)? The human mind is only capable of handling so many details simultaneously. Complexity arises from the presentation of too many details simultaneously. One way of reducing complexity is to just have the required details focused upon while omitting the non-essential details for the problem on hand. The purpose for which the map is being made or used is critical. If the purpose is to have a map that aids hiking trips, perhaps a topographical map would be better to have since it would depict the contours of the landscape clearly. If tourism is the purpose, the items to be brought into focus are "places to see", "places to stay", "historical notes", etc. Therefore it is clear that abstraction must be made to suit a particular purpose.

As another example, consider the need to memorize numbers. Suppose you were given the number 91116925411 to memorize. You may initially find it difficult to memorize this and remember. Trying a bit harder, you may even achieve this. But what if you were to remember a dozen of such numbers. Difficult, isn't it? But suppose we informed you that the number 91116925411 is a telephone number. We can then look for certain groups, such as 91 depicting the country code for India, the 11 following the 91 as the city code for New Delhi and the 692 as the area code for Ashram (a locality in New Delhi). Now the number is easier to remember. Indeed to aid our memory, we can write the telephone number now as 91-11-692-5411, clearly focusing on the important constituents. We now treat the set of digits as a telephone number and not as just a set of digits. So what is the advantage? This method of abstraction enables one to memorize a dozen telephone numbers while a person would have found it difficult to memorize just a couple of 11 digit numbers. The capability to comprehend, memorize and recall are increased by orders of magnitude by the abstraction process.

Abstraction helps in reducing the complexity of systems being considered !

http://vu.bits-pilani.ac.in/Ooad/Lesson5/topic3.htm