Computational Thinking

As part of the introduction home work for computer science you are expected to complete that tasks on Boolean Logic (see work sheet) and the task on computational methods abstraction (see below)

In computing and computer science we use a number of ways to solve problems. One of the common ways is **computational thinking** which includes the following characteristics:

- decomposition,
- abstraction,
- Pattern recognition
- and the use of algorithms

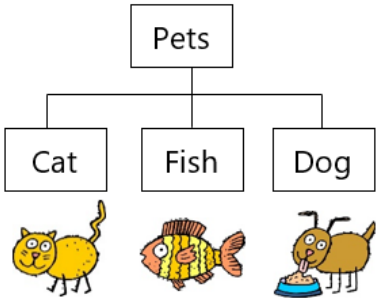
Abstraction is used to help us as computer scientist filter down the important parts of our problem.

The nature of abstraction

In computer science abstraction is the process of including only the important features when solving a problem; this reduces the complexity of the system by removing unnecessary details.

The benefits of using abstraction techniques are that it is easier for the programmer or user to view, to modify and to maintain the solution as they are not distracted by excessive detail which is hidden from them.

The hierarchy diagram below is an abstraction by generalisation for pets



Representational Abstraction is where all unnecessary details are removed, leaving only the information required to solve the problem

Generalisation Abstraction is where a problem is simplified by grouping similar parts into hierarchical structure

Ref Zig Zag Education

Abstraction and reality

Abstraction is one of the key features within object-oriented programming, where the programmer reduces complexity by hiding all but the essential data about an object.

An object, which is based on a class, is a model of a concept, process or real-world item that is relevant to the application.

Variables are used in programs to perform calculations and can represent real-world values as well as intermediate values in a calculation.

Object-oriented programming languages consist of a series of objects that interact with each other.

Java and C++ are commonly used object-oriented programs.

The abstraction method is widely used in object-oriented programming languages; for example, a program may have a method called 'CalculateTax' in their 'Worker' class, which uses WorkerID as an input parameter and returns the income tax paid as a real value. The programmer does not need to know how the calculation is carried out; they simply need to be aware of the name of the method and the input parameter and the format of the output.

Note that for function abstraction the exact computational method used is hidden, unlike procedural abstraction. The use of built-in or library functions is an example of functional abstraction; the user simply calls the function and a value is returned with no knowledge of the internal code within the function.

For example, using a built-in square root function:

SQRT(16) will return the value 4, so x = SQRT(16) will become x = 4

Ref Zig Zag Education

Computational Thinking

Devising abstract models

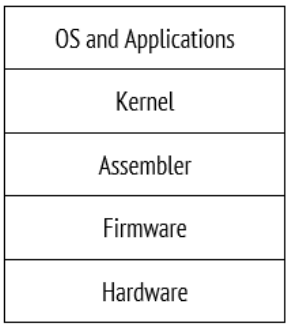
Abstraction is a key feature in computer science where abstract models are created for real-world objects and phenomena. It is essential that sufficient detail is included from the abstraction process to create a model that can solve the problem to the required level of accuracy. This automation process is based on the following:

- Creation of algorithms – which includes the breaking up of the problem into sub-problems and the listing of the steps needed to solve each sub-problem.
- Implementing the algorithms in program code – which includes conversion between the pseudocode algorithm and the instructions of the programming language chosen.
- Implementing the models in data structures – chosen data structures should be suitable for specific model; commonly used data structures include arrays, files and record/ strcuts.
- Executing the code – once the code has been created it should be executed to ensure it runs and then tested/debugged to ensure it operates as expected.

Abstraction layers

Computer systems make use of the concept of layers (or levels) of abstraction to represent a complex problem as a series of layers. The functionality of the various layers is hidden from the other layers.

The diagram below shows a view of the abstraction layers for computer architecture.



Ref Zig Zag Education

Computational Thinking

TASKS

A software developer is creating a Virtual Pet game.

The user can choose the type of animal they would like as their pet, give it a name and then they are responsible for caring for that animal. The user will need to feed, play with, and educate their pet.

The aim is to keep the animal alive and happy, for example if the animal is not fed over a set period of time then the pet will die.

- The game tells the user how hungry or bored the animal is as a percentage (%) and the animal's intelligence is ranked as a number between 0 and 150 (inclusive).
- Hunger and boredom increase by 1% with every tick of a timer.
- When the feed option is selected, hunger is reduced to 0.
- When the play option is selected, bored is reduced to 0.
- When the read option is selected, the intelligence is increased by 0.6% of its current value.

An example of the game is shown:

```
What type of pet would you like? Fox or Elephant?
Fox
What would you like to name your Fox?
Joanne
Joanne's stats are
Hunger: 56%
Bored: 85%
Intelligence: 20
What would you like to do with your pet? Play, Read or Feed?
```

Fig. 1.1

The developer made use of abstraction when creating the Virtual Pet game (see figure 1).

Discuss the need for and purpose of abstraction and how abstraction will be used in the development of the game. (Write an essay of no less than 400 words).