

# ASSIGNMENT 1 FRONT SHEET

<b>Qualification</b>	<b>BTEC Level 4 HND Diploma in Business</b>		
<b>Unit number and title</b>	<b>Unit 3: Human resource management</b>		
<b>Submission date</b>		<b>Date Received 1st submission</b>	
<b>Re-submission Date</b>		<b>Date Received 2nd submission</b>	
<b>Group number:</b>	<b>Student names &amp; codes</b>	<b>Final scores</b>	<b>Signatures</b>
	1.		
	2.		
	3.		
	4.		
<b>Class</b>		<b>Assessor name</b>	
<b>Student declaration</b> I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			

P1	P2	P3	P4	M1	M2	D1	D2

# OBSERVATION RECORD

Student 1				
Description of activity undertaken				
Assessment & grading criteria				
How the activity meets the requirements of the criteria				
Student signature:		Date:		
Assessor		Date:		

<b>signature:</b>			
<b>Assessor name:</b>			



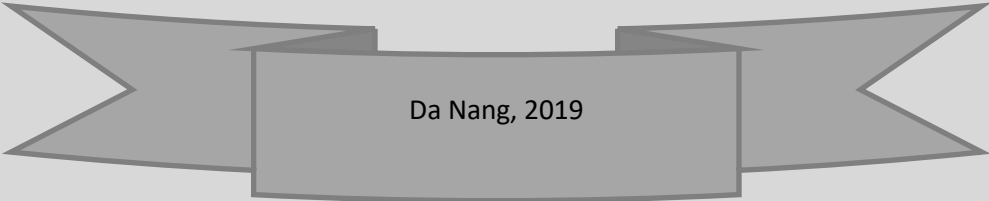
Alliance with  Education

# ***Data Structures & Algorithms***

Instructor: Phan Thi Thanh Tra

Student: Nguyen Van Hieu

Class: GCD0819



Da Nang, 2019

# Summary

This report will provide and designed:

- Explanation on how to specify an abstract data type using the example of software stack
- Explanation of the advantages of encapsulation and information hiding when using an ADT
- Discussion of imperative ADTs with regard to object orientation

## **P3. Explanation on how to specify an abstract data type using the example of software stack**

### **1. Abstract data type**

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.

### **2. Stack**

Stack is an instant data structure in which the extended and deleted exercises are only performed on a single end. In a stack, include and delete parts made in a position called "top". That shows, the newly combined section, the stack's best case scenario and a segment are removed from the stack's most incredible purpose. On the stack, extended and deleted exercises are performed according to the LIFO (Last In First Out) rule

### **3. How to specify an abstract data type?**

#### **Firstly, about abstract data type (ADT)**

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called “abstract” because it gives an implementation independent view. The process of providing only the essentials and hiding the details is known as abstraction.

#### **Secondly, about user of abstract data type**

The user of data type need not know that data type is implemented, for example, we have been using int, float,char data types only with the knowledge with values that can take and operations that can be performed on them without any idea of how these types are implemented.

#### **Conclusion**

So a user only needs to know what a data type can do but not how it will do it. We can think of ADT as a black box which hides the inner structure and design of the data type. Now we'll define three ADTs namely List ADT, Stack ADT, Queue ADT.

#### 4. Specify an abstract data type for a software stack

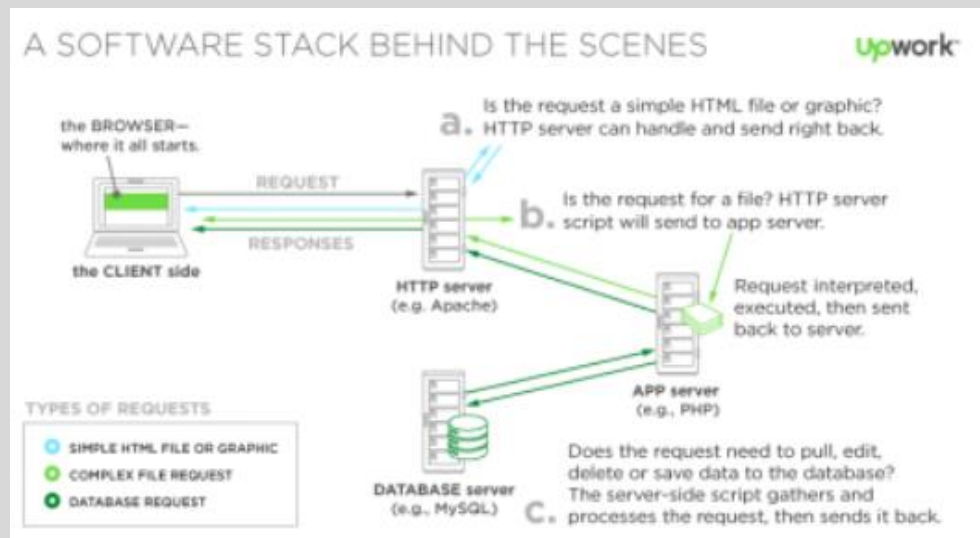
A software stack is a set of software subsystems or components needed to create a complete platform such that no additional software is needed to support applications (Inc, n.d.). Therefore, an application usually includes lots of small software at each layer of the software and each of them plays a different role. Moreover, in order to let them cooperate with each other, abstract data types are required such as Stack (LIFO) and Queue (FIFO).

##### **A stack is composed of components:**

- Operating
- Web Server
- Database Server
- Back-end Programming Language

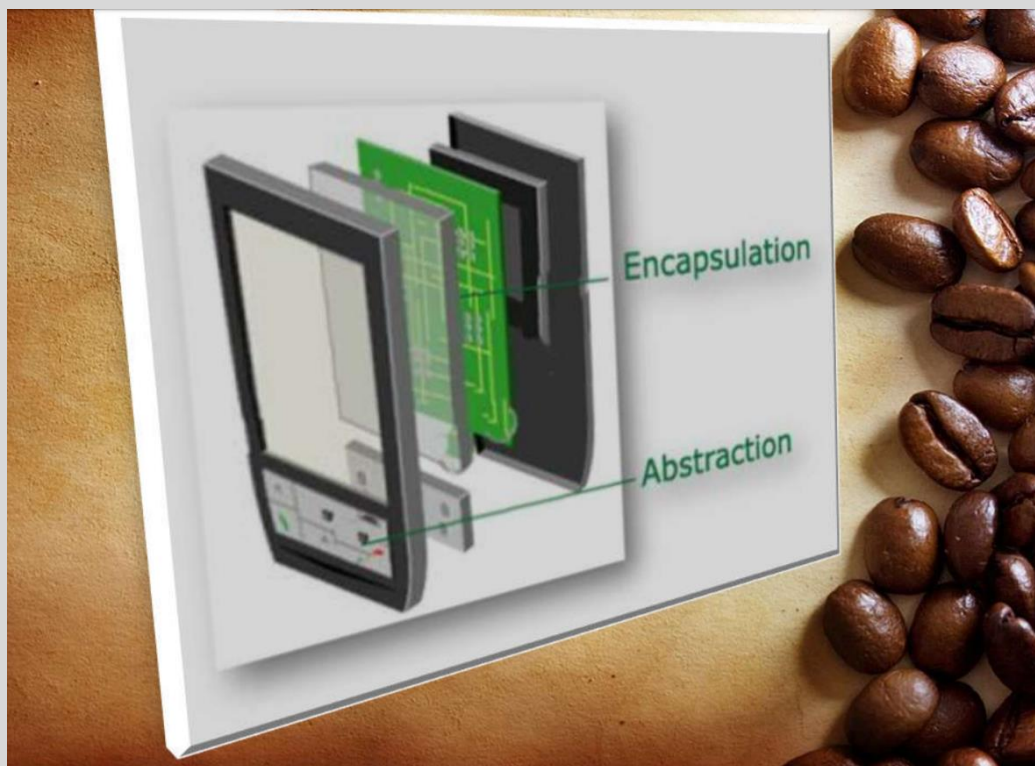
##### **And LAMP stack, the components are in turn:**

- ✓ L (Linux): The working framework (OS) makes up our first layer. Linux sets the foundation for the stack show. Each and every other layer continues running over this layer
- ✓ A (Apache): The second layer comprises of web server programming, ordinarily Apache Web Server. This layer lives over the Linux layer. Web servers are responsible for making an understanding of from web programs to their correct webpage.
- ✓ My sql (M): Our third layer is the place databases live. MySQL stores nuances that can be addressed by scripting to fabricate a site. MySQL ordinarily sits over the Linux layer near to Apache/layer 2. In top of the line setups, MySQL can be off stacked to a different host server
- ✓ PHP (P): Sitting over them all is our fourth and last layer. The scripting layer includes PHP or conceivably other practically identical web programming vernaculars. Sites and Web Applications keep running inside this layer.



Software Stack (Wodehouse, n.d.)

**M3. Examine the advantages of encapsulation and information hiding when using an ADT**





## Encapsulation

- The technique of hiding the used data structure and only provide a well-defined interface is known as Encapsulation.
- The wrapping up of data and functions into a single unit is known as Encapsulation.
- Through encapsulation the data is not accidentally corrupted by external world.
- The access specifiers defined the accessing of data and member functions

## Advantages of encapsulation

- Encapsulation is a very important O-O concept
- Each object has 2 views. An internal view and an external view
- Encapsulation is a form of protection
- Also called Information Hiding
- The outside world does not have direct access to the internal implementation or representation of an object
- As long as the external view does not change, the internal view can
- take on any form without affecting the outside world
- By hiding data and providing methods to gain access to it, an object can maintain high data integrity
- Methods have the responsibility of maintaining data integrity
- Private visibility offers full encapsulation
- Protected and default offer limited encapsulation
- Public offers no encapsulation

## Data hiding

- The hiding of design decisions in a computer program that are most likely to change, thus protecting other parts of the program from change if the decision is changed.
- In object oriented programming, access to the private part of object is restricted in the sense that the functions of the object can only access the data. Thus, private parts of the objects are not available outside the object and cannot be altered by external changes. This property of object is called as Information Hiding or Data Hiding.
- ❖ **Note:** In this, the object of class cannot initialize the private data member the public member function can only initialize and access the data member.

## **D2. Discussion of imperative ADTs with regard to object orientation**

A kind of dynamic information is basically any type described by which an event is not based on to create without any other person. For example, you can have a different type of 'Individual' and other 'Teachers' and 'Subscribers', both of whom are 'subclasses'. In case your application only uses Teachers and Students, then type That person is really a reasonable type. The type of Man can describe one of his own methods to provide typical usefulness for both 'Educators' and 'Underlings'; For example, techniques to classify the person's name and address for an envelope or to convey something clearly. It can describe the characteristics of the proposed systems replaced by sub-types. Therefore, for example, the type of person can describe the methodology's characteristics to give that individual a timetable, but the convenience of that strategy will be categorized between Teacher and Student. Different OOP languages have a special strategy to describe the proposed class is reasonable or describe the dependent procedures to be abolished. For example, in Python, a custom class is reasonably formal if any of its processes are declared as assumptions (according to linguistic rules, you cannot create events of a class in which any system Which system is played in a form) on how a class cannot be articulated is unique.

## References

- [1] Abelson and Sussman. *The Structure and Interpretation of Computer Programs*. MIT Press, 1985.
- [2] N. Adams and J. Rees. Object-oriented programming in Scheme. In *Proc. of the ACM Conf. on Lisp and Functional Programming*, pages 277–288, 1988.
- [3] P. America. A behavioral approach to subtyping object-oriented programming languages. In *Proc. of the REX School/Workshop on the Foundations of Object-Oriented Languages*, 1990.
- [4] M. P. Atkinson and R. Morrison. Procedures as persistent data objects. *ACM Transactions on Programming Languages and Systems*, 7(4):539–559, 1985.
- [5] M. R. Brown and G. Nelson. IO streams: Abstract types, real programs. Technical Report 53, Digital Equipment Corporation Systems Research Center, 1989.
- [6] P. Canning, W. Cook, W. Hill, J. Mitchell, and W. Olthoff. F-bounded polymorphism for objectoriented programming. In *Proc. of Conf. on Functional Programming Languages and Computer Architecture*, pages 273–280, 1989.
- [7] P. Canning, W. Cook, W. Hill, and W. Olthoff. Interfaces for strongly-typed object-oriented programming. In *Proc. of ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications*, pages 457–467, 1989.
- [8] P. Canning, W. Hill, and W. Olthoff. A kernel language for object-oriented programming. Technical Report STL-88-21, Hewlett-Packard Labs, 1988.