Name: OTENE VICTOR.A.

Matric no: 17/SCI01/072

 Object oriented programming Modular programming

 1) OOP is a program structured as a 1) In modular programming procedure

A web of interactive objectives and of common functionality are grouped

These objects interact by sending together in to separate modules

Messages to each other

 2) In OOP the data is hidden and 2) In modular programing data is not

 Therefore, safe from accidental hidden and can be altered and easily

 Altercation and access accessed

 3) OOP is an approach to the overall 3) No matter how well structured large

 Organization of a program programs become excessively complex

 4) New data can easily be created 4) New data cannot be easily created

 5) It is easy for modelling real world 5) It is difficult for modelling real world

 Because attributes and behaviour Because attributes and behaviour are

 Are not separated

1. The following conditions listed below would satisfy the transformation of a modular program to an OOP design
* Encapsulation

This concept is also often used to hide the internal representation, or state, of an object from the outside. This is called information hiding. The general idea of this mechanism is simple. If you have an attribute that is not visible from the outside of an object, and bundle it with methods that provide read or write access to it, then you can hide specific information and control access to the internal state of the object.

In retrospect by encapsulating the modular code it would make the following possible:

- Information hiding to prevent accidental altercation

- Modelling real world entities if need be in the program

- Proper Organization

- Readability i.e making the code understandable and readable by other programmers. Therefore, making it easier for the programmers to easily write updates for the program without having to rewrite all the functions

- Creating new data easily

- Making it easier for each function/object to interact with each other

* Abstraction

Generally, abstraction is the concept of hiding internal operations of object and only revealing operations relevant for the proper operation of the objects. Applying abstraction means that each object should only expose a high-level mechanism

Without applying abstraction to the code each object used in the program would have to be explicitly defined with each showing the implementation details. And this can only result in making the code - inefficient

- unnecessarily long and complex

 - and slow to operate.

* Inheritance

Objects are often very similar. They share common logic. But there are not entirely the same. Inheritance allows the reuse of the common logic into separate class. Which generally means creating a child class by deriving from a parent class.

Inheritance makes it possible for:

- Creating new data

- Makes it easier for each function/object to interact with each other

- Helps to simplify code by building relationship between functions

* Polymorphism

Say we have a parent class and a few child classes which interact from it. Sometimes we want to use a collection e.g a list which contains a mix of all these classes. Or we have a method implemented for the parent class. But we’d like to use it for the children too. This can be solved by using polymorphism.

Polymorphism permits:

- New data to be created

- Makes it easier for each function/object to interact with each other

- Making code less complex