UNIT ONE: INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING

Learning outcomes

Upon completing this topic you should be able to:

- Define object-oriented concepts that includes classes, objects, encapsulation, message passing encapsulation, polymorphism and inheritance
- Differentiate an object-oriented programs and an object-oriented programming language
- Describe properties of object-oriented programs

1.1 Introduction to object-oriented programs

Object-oriented programs (OOPs) are programs that are made of objects. These objects interact with one another to achieve specific objectives.

Objects interact with one another through message passing. The objects that make up an OOP are instantiated from classes.

Classes are building blocks of OOPs. As compared to structured programs, object- oriented programs are easier to develop and maintain.

Object oriented programming is the process of developing an object-oriented program.

The programming languages that are used in writing object-oriented programs are referred to as object-oriented programming languages. Examples include C++, Java, Python, etc

1.2 Fundamental Object-Oriented Concepts

Fundamental object-oriented concepts includes:- objects, classes, inheritance, encapsulation, message passing, information hiding, abstraction and polymorphism.

Object

An object is:

a. A run-time entity and it consumes computer memory;
b. An instance of a class;
c. An entity that has state, behavior and identity.

Class

A class is

a. A template that is used to construct objects.
b. A collection of objects that have similar state and behavior.
c. A collection of objects of similar type.

Inheritance

Inheritance is the ability to define new classes from already existing classes i.e. the ability of objects r classes to acquire the properties/characteristics of another. This is a powerful concept in programming. It facilitates development of programs that are of high quality.

A class that is defined, created or derived from another class is referred to as a *sub-class, child class* or a *derived class.*

A class that is used to create or derive another class is referred to as a super-class, base or a parent-class.

There are four types of inheritance:

a. **Single inheritance**: A sub-class has one parent class
b. **Multiple inheritance**: A sub-class has more than one parent class. There are some languages such as Java that does not support multiple inheritance.
c. **Multi-level inheritance**: A sub-class has one parent class and the parent class is a sub-class of another parent class
d. **Hybrid inheritance:** It is a combination of single-inheritance, multiple inheritance and multi-level inheritance.

A class that is derived from another class inherits properties of the base class.

Encapsulation

Encapsulation is:

a. Enclosing state and behavior of an object.
b. Putting together state and behavior of objects in single unit. Encapsulation is implemented through the use of classes.

Information hiding

Is the prevention of state and behavior of an object from direct access by other object. Information hiding is implemented through the use of access specifiers that includes: private, public and protected. Private, public and protected are keywords

Abstraction

Abstraction is the act of showing only the general properties of an object leaving out the detailed properties. Abstraction useful when designing complex programs. A class is an example of an abstract data type (ADT)

Message passing

Message passing is the way objects interact with one another in an object-oriented program. Objects interact by sending messages to one another. For example, if “person” and “window” are two objects in an object-oriented program, the “person” object may send a message to the “window” requesting it to close.

A message is a request for something to be done. Message passing has the following information.

a. The name of the object that the message is being sent to
b. The name of the message
c. Information that the receiver of the message requires in order to respond to the message.

Polymorphism

Polymorphism is the ability of two or more objects to respond differently when they are sent the same message. For example, the window and the door objects can be sent close message. However, the way the window object will respond to that message will be different from the way the door object will respond to the same message.

1.3 Writing an Object-Oriented Program using a High-level Language

The following are steps that are followed in order to execute an object-oriented program written in a high-level language

1. First, a text-editor is used to create a program in a high-level language such as Java or C++. The program is written following syntax and semantics of the language being used. This program is referred to as source code. The program source code is saved.
2. The source-code is compiled using a compiler. The compiler checks the source code for syntax errors, and if no error is found, it translates the program into equivalent machine code.
3. Linker combines machine code produced by the compiler with machine code from the libraries included in the program to produce an executable file.
4. The executable file is loaded into the main memory for execution. This is done by a program referred to as loader.

Revision Questions

EXERCISE 1. Differentiate an object and a class

Example ✐. Define a compiler

Solution: A compiler is a program that checks for syntax errors and translate pro- gram source code into machine code

EXERCISE 2. Compare and contrast an object-oriented program and a structured program

EXERCISE 3. Explain how inheritance improve quality of a program

EXERCISE 4. Explain the importance of polymorphism and encapsulation