**Introduction to Object Oriented Programming - Text directions, refs and code**

Object-oriented programming (OOP) is a programming paradigm that uses objects and their interactions to design and program applications

- Allows the program to block off areas of code that perform certain tasks independently of other areas in the application.

Encapsulation - concept of blocking off areas of code and not making it available to the rest of the program

Abstraction - is simplifying a complex process of a program, an enterprise software solution for example by modeling classes appropriate for it

Inheritance - is used where a class inherits the behavior of another class, referred to as the superclass

Polymorphism - is when a class inherits the behaviors of another class, but has the ability to not inherit everything and change some of it’s inherited behaviors. For example to write a method that does something differently from the inherited method

Classes - It is a blueprint that describes the state and behavior that the objects of the class all share. A class can be used to create many objects. Objects created at runtime from a class are called instances of that particular class.

Example of a user class

* class User
* attr\_accessor :name, :email
* def initialize(name, email)
* @name = name
* @email = email
* end
* def run
* puts "Hey I'm running"
* end
* def self.identify\_yourself
* puts "Hey I am a class method"
* end
* end
* user = User.new("mashrur", "mashrur@example.com")
* user.run
* User.identify\_youself # to run this class method you don't need an instance of user
* # you can directly call the class User