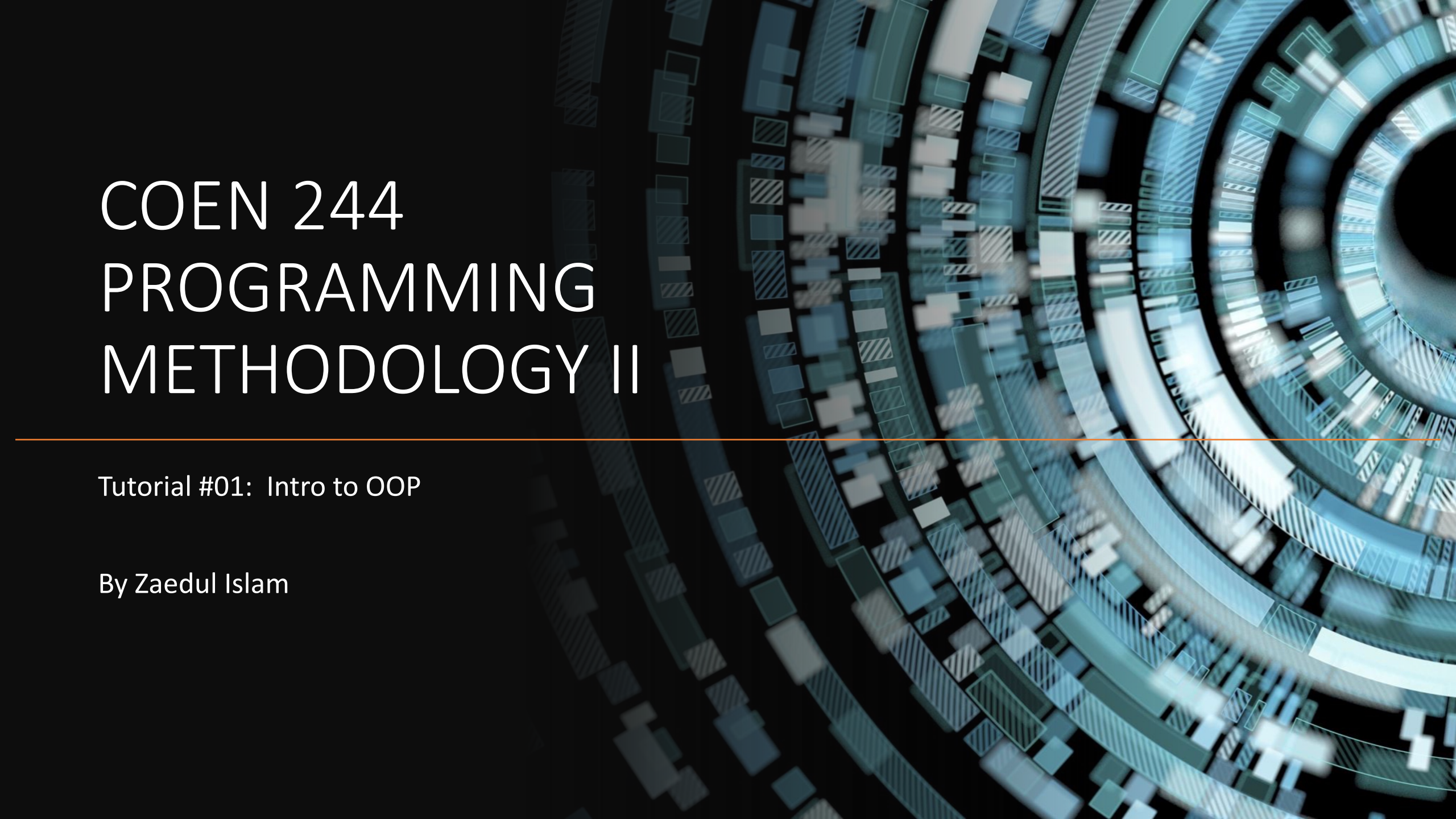


COEN 244 PROGRAMMING METHODOLOGY II

Tutorial #01: Intro to OOP

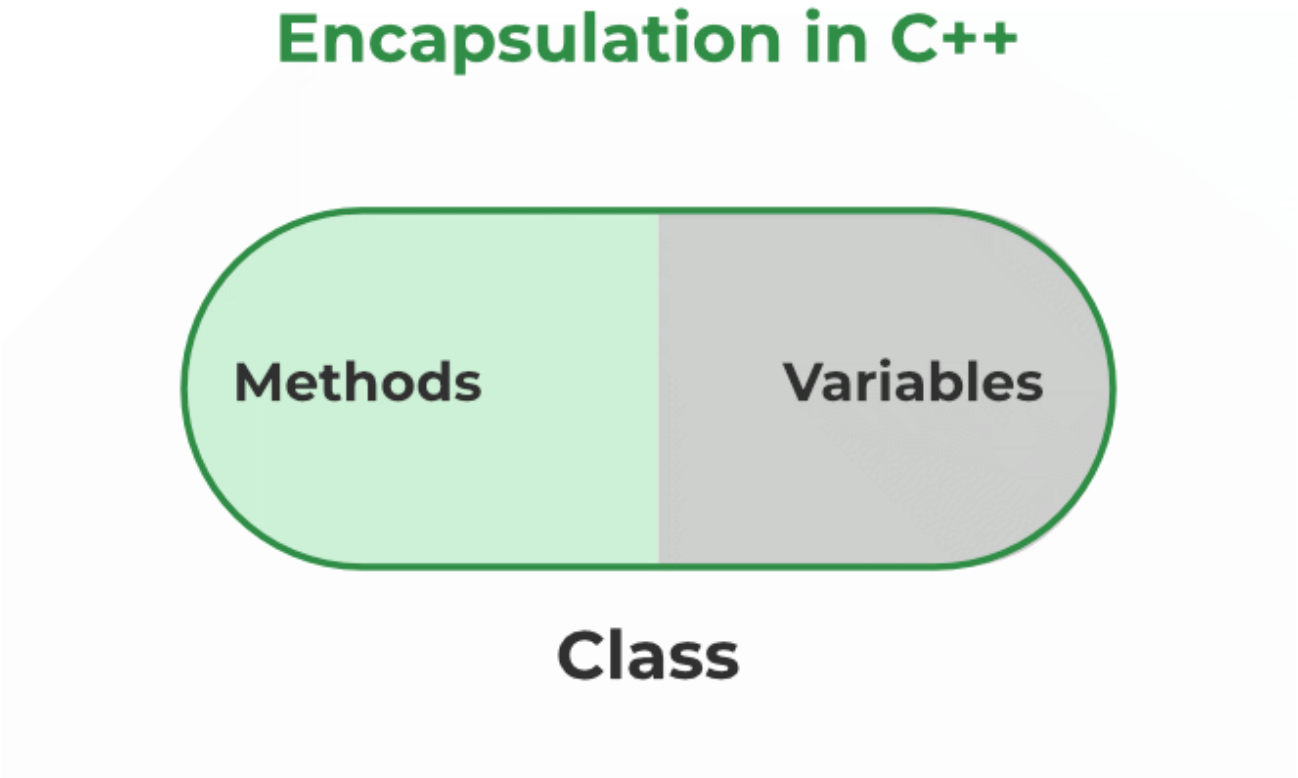
By Zaedul Islam

Philosophy behind Object Oriented Programming

- The philosophy behind object-oriented programming (OOP) in C++ is to model real-world objects and their interactions within a program. In OOP, a software system is viewed as a collection of interacting objects, rather than a collection of procedures and data.
- By using OOP in C++, developers can create software that is **more modular**, **easier to understand**, and **easier to maintain**. OOP allows them to organize the code in a more meaningful way, and to encapsulate complex functionality in separate objects, making the system more manageable.
- What is Object-oriented programming (OOP)?
  - It's is a programming paradigm that is based on the concept of classes and objects.

Object Oriented Programming (contd.)

- In other words, it is a way of organizing and structuring code that is based on the concept of classes and objects.
- OOP has several key principles:
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Abstraction



Classes

- OOP is a programming paradigm, and a class is a blueprint to create objects that follow that paradigm. Classes are the building blocks of OOP, as they provide the structure and behavior for the objects in the system. A class is like a blueprint for making something.
- For example, if we have a class called "Car", it will have the instructions on how to make a toy car - how it should look like, how many wheels it should have, what color it should be. And once we use that blueprint, we can make as many toy cars as we want each with their own features.
  - This class has two private member variables: "wheels" and "color", and three public member functions: a constructor, a function to set the color, and a function to get the color.

Objects

- We can create multiple objects of this class, each object will have their own values for the variables "wheels" and "color”
- In this example we create two car objects, car1 and car2 and each one of them is an instance of the Car class. We can then use the class functions to get or set their properties and they will not interfere with one another.

Review: Local and Global Variables

- Variables defined inside a function are local to that function. They are hidden from the statements in other functions, which normally cannot access them.
- Because the variables defined in a function are hidden, other functions may have separate, distinct variables with the same name.
- A global variable is any variable defined outside all the functions in a program.
- The scope of a global variable is the portion of the program from the variable definition to the end.
- This means that a global variable can be accessed by all functions that are defined after the global variable is defined.

Local Variables Lifetime

- A function's local variables exist only while the function is executing. This is known as the lifetime of a local variable.
- When the function begins, its local variables and its parameter variables are created in memory, and when the function ends, the local variables and parameter variables are destroyed.
- This means that any value stored in a local variable is lost between calls to the function in which the variable is declared.

Static Local Variables

- Local variables only exist while the function is executing. When the function terminates, the contents of local variables are lost.
- *static* local variables retain their contents between function calls.
- *static* local variables are defined and initialized only the first time the function is executed. *0* is the default initialization value.
- Example

Exercise 1

- Define a class to represent a bank account. Include the following members:
  - Data members:
    1) Name of the depositor
    2) Account number
    3) Type of account
    4) Balance amount in the account.

  - Member functions:
    1) To assign initial values
    2) To deposit an amount
    3) To withdraw an amount after checking the balance
    4) To display name and balance.
- Write a main program to test the program.
- Solution

Exercise 2

- Do the required modification on Exercise 1 to generate the account Number automatically with the help of **static data member**.
- The account number should follow the following pattern (10500100X), which x is a counter start from 1.
  - For example, the account number of first client will be 105001001, the second 105001002, and go on.
- Solution