Lab #10

Chapter 17 : Classes & Objects - The Basics.

This chapter introduces the core concepts of OOP. It explains how classes act as blueprints for creating objects, and how objects represent real-world entities with attributes and behaviors. Key topics include defining user-created classes, constructing and initializing objects with "__init__", and creating methods for functionality. The chapter also discusses using objects as parameters, returning instances from functions, and covering objects to strings. These basics lay the foundation for building organized, modular code.

Chapter 18: Classes and Objects.

This chapter builds on the basics by exploring more complex behavior of objects. It explains how objects are mutable, meaning their attributes can be changed after creation. It covers the concept of sameness (== vs is), and introduces arithmetic methods like "__and__" and "__sub__", which enable operator overloading. The chapter uses the "Fraction" as a case study, showing how to make user-defined objects behave like built in types. It emphasizes deeper control over how classes behave and interact.

Chapter 19 : Inheritance.

Inheritance allows one class to inherit attributes and methods from another, promoting code reuse and hierarchy. The chapter outlines the <u>three pillars of OOP:</u>

1. Encapsulation.
2. Inheritance.
3. Polymorphism.

It introduces subclassing, method overriding, and constructors in subclasses. A key concept is composition, which uses other classes as components rather than inheriting from them. The case study on structured postal addresses demonstrates these principles in practice, illustrating how to apply "inheritance" and "isinstance()" checks in real-world designs.

UML—Diagram PowerPoint Slides :-

Unified Modeling Language (UML) is a standardized visual language for modeling software systems. It helps design and communicate structure and behavior using diagrams. The slides emphasize class diagrams, showing classes, their attributes, methods, and relationships. Important relationship types include:

- Association (normal reference),
- Aggression (whole-part, but independent lifecycles),
- Composition (whole-part with shared lifecycles),
- Dependancy (temporary, weaker relationship),
- Inheritance (is-a relationship),
- Implementation (class fulfilling an interface).

Each relationship type uses different arrowheads, which are critical for understanding class interactions in UML Diagrams for a simple arcade game.