Object-Oriented Programming: Basics Q&A

Question 1: What is Object-Oriented Programming (OOP)?

Object-Oriented Programming is a programming paradigm that organizes code around objects and classes rather than functions and logic. It models real-world entities as objects with properties (attributes) and behaviors (methods), making code more modular, reusable, and maintainable.

Question 2: What are the four main pillars of OOP?

The four main pillars of OOP are:

1. **Encapsulation** - Bundling data and methods together, hiding internal details from outside access.
2. **Abstraction** - Showing only essential features while hiding complex implementation details.
3. **Inheritance** - Deriving new classes from existing classes to promote code reuse.
4. **Polymorphism** - Allowing objects to take multiple forms or methods to behave differently based on context.

Question 3: What is a Class and an Object?

A **class** is a blueprint or template that defines the structure and behavior of objects. An **object** is an instance of a class - a concrete realization of that blueprint with actual data and specific values.

Example: Class = "Car" (blueprint), Object = "My Red Honda" (specific car instance).

Question 4: What is Encapsulation?

Encapsulation is the bundling of data (variables) and methods (functions) that operate on that data into a single unit (class), while hiding the internal implementation details. It protects data from unauthorized access and modification by using access modifiers like private, protected, and public.

Question 5: Explain Inheritance with an example.

Inheritance allows a class (child/derived class) to inherit properties and methods from another class (parent/base class). This promotes code reuse and establishes a hierarchical relationship.

Example: Class "Animal" is the parent. Classes "Dog" and "Cat" inherit from "Animal", so they automatically have properties like "name" and methods like "eat()".

Question 6: What is Polymorphism?

Polymorphism means "many forms". It allows objects or methods to take multiple forms. There are two types:

1. **Compile-time Polymorphism** - Method Overloading (same method name, different parameters).
2. **Runtime Polymorphism** - Method Overriding (child class provides different implementation of parent's method).

Question 7: What is Abstraction?

Abstraction is the process of hiding complex implementation details and showing only the necessary features. It reduces complexity and improves code readability. Abstract classes and interfaces are used to achieve abstraction.

Example: A user sees a button on a UI and clicks it. The complex code behind the button's functionality is hidden (abstracted).

Question 8: Difference between Composition and Inheritance.

Inheritance - "IS-A" relationship. A child class inherits from a parent class. Example: Dog IS-A Animal.

Composition - "HAS-A" relationship. A class contains objects of other classes. Example: A Car HAS-A Engine.

Composition is often preferred because it's more flexible and avoids tight coupling.

Question 9: What are Access Modifiers?

Access modifiers control the visibility and accessibility of class members (variables and methods):

1. **Public** - Accessible from anywhere, both inside and outside the class.
2. **Private** - Accessible only within the same class, not from outside or derived classes.
3. **Protected** - Accessible within the class and derived classes, but not from outside.
4. **Default/Package-Private** - Accessible only within the same package (varies by language).

Question 10: What are Constructors and Destructors?

Constructor - A special method called automatically when an object is created. It initializes object properties and performs setup tasks. Can be overloaded.

Destructor - A special method called automatically when an object is destroyed or goes out of scope. It performs cleanup tasks like releasing memory or closing resources (mainly used in languages like C++).

In Python, `__init__()` is the constructor and `__del__()` is the destructor.