**Đề bài: \* Tổng hợp lại tất cả các kiến thức :  
- Solution,Project là gì ?  
- Object là gì ? Class là gì ?  
- Các đặc tính OOP ?   
- Abstract vs Interfact như thế nào ?  
- Nêu các ví dụ thực tế ứng dụng OOP trong làm phần mềm.**

**1. Solution và Project là gì?**

* **Solution**
  + Là một “khung” chứa nhiều project liên quan đến nhau.
  + Cho phép quản lý và xây dựng các ứng dụng phức tạp, chia nhỏ thành nhiều module độc lập.
  + Trong Visual Studio, solution thường có phần mở rộng là *.sln*.
* **Project**
  + Là một đơn vị độc lập chứa mã nguồn, tài nguyên và cài đặt biên dịch.
  + Mỗi project có thể tạo ra một ứng dụng, thư viện hay dịch vụ riêng biệt.
  + Các project trong solution có thể tham chiếu lẫn nhau để tái sử dụng code.

**2. Object và Class là gì?**

* **Class**
  + Là một khuôn mẫu (blueprint) hoặc mô hình định nghĩa các thuộc tính (properties), phương thức (methods) và sự kiện (events) của một đối tượng.
  + Đóng vai trò như một thiết kế để tạo ra các đối tượng cụ thể.
* **Object**
  + Là một thể hiện (instance) của class.
  + Khi tạo ra một object từ class, nó sẽ chứa dữ liệu và hành vi được định nghĩa trong class đó.
  + Ví dụ: Class Car có các thuộc tính như màu sắc, tốc độ; một object cụ thể như myCar là một chiếc xe có màu đỏ, tốc độ 120 km/h.

**3. Các đặc tính của OOP (Lập trình Hướng đối tượng)**

* **Đóng gói (Encapsulation):**
  + Giúp ẩn dữ liệu nội bộ của đối tượng và chỉ cho phép truy cập thông qua các phương thức (getter/setter).
  + Bảo vệ dữ liệu tránh bị thay đổi không mong muốn từ bên ngoài.
* **Kế thừa (Inheritance):**
  + Cho phép tạo ra các lớp con (subclasses) kế thừa thuộc tính và phương thức của lớp cha (superclass).
  + Hỗ trợ tái sử dụng code và mở rộng chức năng.
* **Đa hình (Polymorphism):**
  + Cho phép đối tượng của các lớp con khác nhau có thể được xử lý thông qua cùng một giao diện của lớp cha.
  + Thực hiện thông qua việc ghi đè (override) phương thức hoặc nạp chồng (overload) phương thức.
* **Trừu tượng (Abstraction):**
  + Là quá trình ẩn đi các chi tiết cài đặt phức tạp và chỉ hiển thị các tính năng cần thiết.
  + Được thực hiện qua việc sử dụng lớp trừu tượng (abstract class) hoặc giao diện (interface).

**4. Abstract vs Interface**

* **Abstract Class**
  + Có thể chứa cả các phương thức có cài đặt và phương thức trừu tượng (không cài đặt).
  + Cho phép định nghĩa các trường dữ liệu (fields) và các thành phần bảo vệ dữ liệu.
  + Một lớp chỉ có thể kế thừa từ một abstract class.
  + Thích hợp khi các lớp có mối quan hệ “is-a” và có chung một số hành vi mặc định.
* **Interface**
  + Chỉ định nghĩa các phương thức, thuộc tính, sự kiện mà không có phần cài đặt (trong các phiên bản cũ; phiên bản mới của C# cho phép cài đặt mặc định nhưng vẫn giới hạn).
  + Không chứa trường dữ liệu (fields) và chỉ dùng để định nghĩa hợp đồng mà các lớp cần tuân theo.
  + Một lớp có thể triển khai nhiều interface cùng lúc.
  + Thích hợp khi bạn muốn đảm bảo các lớp khác nhau đều có cùng một giao diện hoạt động, dù chúng không có quan hệ kế thừa trực tiếp.

**5. Ví dụ thực tế ứng dụng OOP trong làm phần mềm**

* **Phần mềm quản lý khách sạn:**
  + Sử dụng OOP để định nghĩa các lớp như HotelStaff, Receptionist, Housekeeper, Customer,…
  + Áp dụng tính kế thừa và đa hình để xử lý các loại nhân viên khác nhau với cách tính lương riêng biệt.
* **Phần mềm quản lý nhân sự (HR):**
  + Các đối tượng như Employee, Manager, Intern được xây dựng từ cùng một class cơ sở và mở rộng thêm các thuộc tính riêng.
* **Phát triển game:**
  + Sử dụng các lớp như Character, Enemy, Player để định nghĩa hành vi, thuộc tính và tương tác trong game.
* **Phần mềm quản lý bán hàng:**
  + Các đối tượng Product, Order, Customer được định nghĩa rõ ràng, cho phép mở rộng chức năng, bảo trì dễ dàng.
* **Ứng dụng ngân hàng:**
  + Xây dựng các lớp như Account, SavingsAccount, CheckingAccount với các tính năng riêng biệt và đảm bảo tính bảo mật thông qua đóng gói.