1. **Abstract Class vs. Interface**

Question:

a. What is the main difference between an abstract class and an interface in PHP?

Abstract Class	Interface
Contains declaration and implementation	Contains only declaration part
Not allows multiple inheritance	Allows multiple inheritance
Contains Constructor	Does not contain constructor
Allows static members	Does not allow static members
Only contains public access modifier	Only contains public access modifier
The performance of an abstract class is fast	The performance of an interface is slow
Can contain: **methods** and **fields**	Can contain: **methods** only
It can be fully, partially or not implemented	It **should** be fully implemented
Good choice when you are sure some methods are concrete/defined and must be implemented in the **SAME WAY** in all derived classes.	Good choice when you know a method has to be there, but it can be implemented **DIFFERENTLY** by independent derived classes.

b. Provide an example of a scenario where you would prefer to use an abstract class over an interface.

Example:
Developing an app with a built-in user management system. Everyone, from regular users to admins, can sign in and update their profiles. But admins have additional capabilities like managing user permissions and accessing administrative reports.

Solution using an Abstract Class:
An abstract class proves ideal here. It enables us to capture (encapsulate) the common functionalities (login, profile update) shared by both user types. At the same time, it enforces the implementation of specific behaviors (permission management) by the Admin class.