به نام خدا

جواب تمرین 1:

DIP یک اصل طراحی است وابستگی‌ها باید به سمت انتزاع (Interface/Abstraction) باشند نه پیاده‌سازی ، یعنی کلاس‌ها به یک قرارداد (interface) وابسته باشند، نه به کلاس مشخص.

IoC : یک الگوی طراحی است و به جای اینکه یک کلاس خودش وابستگی‌هایش را بسازد، کنترل ایجاد و مدیریت آن وابستگی‌ها به یک مکان دیگر (فریم‌ورک/کانتینر) واگذار می‌شود. در واقع پیاده‌سازی DIP را ساده‌تر می‌کند.

DI : یک تکنیک محبوب برای IoC ، به جای اینکه یک کلاس خودش وابستگی‌هایش را بسازد، آن وابستگی‌ها از بیرون (مثلاً توسط IoC Container) به آن تزریق می‌شوند.

IoC Container : فریم ورکی که DI را مدیریت می کند وظیفه ی آن ، ساختن اشیاء (object creation)مدیریت وابستگی‌ها (dependency management) و کنترل چرخه‌ی حیات سرویس‌ها را برعهده دارد.

مثال: Microsoft.Extensions.DependencyInjection

Factory as IoC Container : شکل ساده ای از IoC Container ، یک کلاس یا متد مسئول ساخت اشیاء است. اگر این Factory فقط کار ایجاد وابستگی‌ها را انجام دهد، می‌تواند به‌صورت ساده نقش یک IoC Container را ایفا کند.

اما IoC Container پیشرفته‌تر از Factory است (چون مدیریت چرخه‌ی حیات و injection خودکار را هم دارد).

Service Lifetime : نحوه ی مدیریت اشیا در IoC Container ، وقتی سرویسی را در IoC Container رجیستر می‌کنیم، باید مشخص کنیم چرخه‌ی عمر (lifetime) آن چه‌طور باشد.