قواعد SOLID

1. اصل مسئولیت واحد (Single Responsibility Principle):

هر کلاس باید وظیفه انجام یک کار را داشته باشد. (مثال گزارش‌گیری و ارسال ایمیل)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. اصل باز و بسته (Open/Close Principle):

کلاس‌ها باید برای توسعه باز و برای تغییر بسته باشند.

یعنی طراحی باید به گونه‌ای انجام شود که اضافه کردن متودها و پراپرتی‌های جدید، بدون تغییر کدهای قبلی صورت بگیرد. (مثال محاسبه مساحت)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. اصل جایگزینی (Liskov Substitution Principle):

اشیاء ساخته شده از زیرکلاس‌ها باید بتوانند به راحتی جاگزین اشیاء کلاس والد شوند بدون اینکه رفتار برنامه را تغییر دهند. (مثال پرواز پنگوئن)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. اصل جداسازی رابط‌ها (Interface Segregation Principle):

کلاس‌ها نباید متودی را پیاده سازی کنند که به آن نیازی ندارند. (مثال غذا خوردن ربات)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. اصل وارونه سازی وابستگی‌ (Dependency Inversion Principle):

کلاس‌های سطح بالا نباید به کلاس‌های سطح پایین وابسته باشند. هر دو باید به انتزاع‌ها وابسته باشند. (مثال Sql بودن دیتابیس و وابسته بودن کلاس به نوع دیتابیس و عدم استفاده از DI)