مبادئ SOLID هي مجموعة من المبادئ الأساسية في تصميم البرمجيات، تهدف إلى كتابة كود نظيف، قابل للصيانة، وقابل للتوسيع. تم تقديم هذه المبادئ من قبل المهندس "روبرت سي. مارتن" (Robert C. Martin) وهي تُستخدم على نطاق واسع في هندسة البرمجيات، خاصة في البرمجة الكائنية (Object-Oriented Programming).

فيما يلي شرح لكل مبدأ من مبادئ SOLID:

---

1. Single Responsibility Principle (SRP)

مبدأ المسؤولية الواحدة

كل صنف (Class) يجب أن تكون له مسؤولية واحدة فقط، ويجب أن تتغير لأجل سبب واحد فقط.

مثال:

صنف يقوم بإدارة المستخدمين يجب ألا يحتوي أيضًا على منطق إرسال البريد الإلكتروني.

---

2. Open/Closed Principle (OCP)

مبدأ الانفتاح/الانغلاق

يجب أن تكون الكائنات (Classes, Modules) قابلة للامتداد ولكن مغلقة للتعديل.

بمعنى: يمكن إضافة وظائف جديدة دون تعديل الكود الموجود.

مثال:

بدلاً من تعديل صنف ReportPrinter لإضافة طريقة طباعة جديدة، يمكن إنشاء صنف جديد يرث منه ويضيف تلك الميزة.

---

3. Liskov Substitution Principle (LSP)

مبدأ استبدال ليسكوف

يجب أن تكون الكائنات المشتقة قادرة على استبدال الكائنات الأساسية دون أن تُغير من صحة البرنامج.

مثال:

إذا كانت Bird هي صنف أساسي وPenguin هو صنف مشتق، فيجب أن يتمكن Penguin من العمل في أي مكان تُستخدم فيه Bird دون مشاكل.

---

4. Interface Segregation Principle (ISP)

مبدأ فصل الواجهات

لا يجب على الأصناف أن تعتمد على واجهات لا تستخدمها.

بمعنى: من الأفضل وجود عدة واجهات صغيرة بدلاً من واجهة واحدة كبيرة.

مثال:

بدلاً من واجهة IMachine تحتوي على Print, Scan, Fax، يمكن تقسيمها إلى IPrinter, IScanner, IFax.

---

5. Dependency Inversion Principle (DIP)

مبدأ عكس التبعية

يجب أن تعتمد الوحدات العليا على التجريدات وليس على التفاصيل.

يجب أن تعتمد التفاصيل على التجريدات وليس العكس.

مثال:

بدلاً من أن يعتمد صنف OrderService على صنف MySQLDatabase مباشرة، يجب أن يعتمد على واجهة IDatabase