**Solid**

مبدأ SOLID هو مجموعة من المبادئ التوجيهية في البرمجة الشيئية تهدف إلى تصميم البرامج بطريقة مرنة وقابلة للصيانة والتوسع. يُعتبر SOLID اختصارًا للمبادئ الخمسة التالية:

1. S - Single Responsibility Principle (مبدأ المسؤولية الواحدة):

كل فئة يجب أن تكون مسؤولة عن مهمة واحدة فقط، ولا يجب أن تتعدى هذه المسؤولية. بمعنى آخر، يجب أن يكون لكل فئة سبب واحد فقط للتغيير.

**// غير مطبقة**

class UserManager {

public function getUserData($userId) {

// استعلام لجلب بيانات المستخدم من قاعدة البيانات

}

public function sendEmailToUser($userId, $message) {

// إرسال بريد إلكتروني إلى المستخدم

}

}

**// مطبقة**

class UserDataManager {

public function getUserData($userId) {

// استعلام لجلب بيانات المستخدم من قاعدة البيانات

}

}

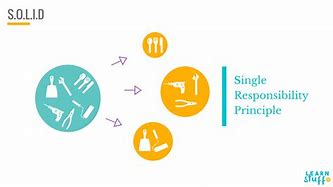
class EmailSender {

public function sendEmailToUser($userId, $message) {

// إرسال بريد إلكتروني إلى المستخدم

}

}



2. O - Open/Closed Principle (مبدأ التعديل/الإغلاق):

البرمجيات يجب أن تكون مفتوحة للامتداد (مفتوحة) ولكن مغلقة للتعديل. بمعنى آخر، يجب أن تكون الفئات مفتوحة للتمديد عن طريق إضافة مزايا جديدة، ولكن يجب أن تكون مغلقة للتعديل عن طريق تعديل المزايا القائمة.

**// غير مطبقة**

class Shape {

public function area() {

// حساب مساحة الشكل

}

}

class Circle extends Shape {

public function area() {

// حساب مساحة الدائرة

}

}

class Square extends Shape {

public function area() {

// حساب مساحة المربع

}

}

**// مطبقة**

interface Shape {

public function area();

}

class Circle implements Shape {

public function area() {

// حساب مساحة الدائرة

}

}

class Square implements Shape {

public function area() {

// حساب مساحة المربع

}

}

A diagram of different methods

Description automatically generated

3. L - Liskov Substitution Principle (مبدأ استبدال ليسكوڤ):

يجب أن يمكن استبدال الكائنات من نوع فئة مشتقة (أو نوع فرعي) مكان الكائنات من نوع الفئة الأساسية (أو نوع القاعدة) دون تعديل في سلوك البرنامج.

**// غير مطبقة**

class Bird {

public function fly() {

// طيران الطيور

}

}

class Ostrich extends Bird {

public function fly() {

throw new Exception("النعام لا تستطيع الطيران");

}

}

**// مطبقة**

interface Bird {

public function fly();

}

class Sparrow implements Bird {

public function fly() {

// طيران العصفور

}

}

class Ostrich implements Bird {

public function fly() {

throw new Exception("النعام لا تستطيع الطيران");

}

}

4. I - Interface Segregation Principle (مبدأ فصل الواجهات):

يجب عدم إلزام العملاء على استخدام واجهات يحتوي على الكثير من الوظائف التي لا تحتاج إليها. بدلاً من ذلك، يجب تقسيم واجهات كبيرة إلى واجهات أصغر وأكثر ملاءمة وفقًا لاحتياجات العميل.

**// غير مطبقة**

interface Worker {

public function work();

public function eat();

public function sleep();

}

class Engineer implements Worker {

public function work() {

// عمل المهندس

}

public function eat() {

// تناول الطعام

}

public function sleep() {

// النوم

}

}

**// مطبقة**

interface Workable {

public function work();

}

interface Eatable {

public function eat();

}

interface Sleepable {

public function sleep();

}

class Engineer implements Workable, Eatable, Sleepable {

public function work() {

// عمل المهندس

}

public function eat() {

// تناول الطعام

}

public function sleep() {

// النوم

}

A diagram of a book

Description automatically generated with medium confidence}

5. D - Dependency Inversion Principle (مبدأ تقليب التبعية):

يجب أن تعتمد الفئات على واجهات بدلاً من تفاصيل التنفيذ، ويجب أن يكون التوجه نحو التفاصيل المنخفضة من الأعلى إلى الأسفل (من الفئات العالية المستوى إلى الفئات المنخفضة المستوى).

**// غير مطبقة**

class Database {

public function connect() {

// اتصال بقاعدة البيانات

}

}

class UserRepository {

private $database;

public function \_\_construct() {

$this->database = new Database();

}

}

**// مطبقة**

interface Connection {

public function connect();

}

class Database implements Connection {

public function connect() {

// اتصال بقاعدة البيانات

}

}

class UserRepository {

private $database;

public function \_\_construct(Connection $connection) {

$this->database = $connection;

}

}

A diagram of a package and object

Description automatically generated

A diagram of a light bulb and a light bulb connected to a outlet

Description automatically generated

تطبيق هذه المبادئ يساعد في إنشاء برمجيات قابلة للصيانة، وسهلة الفهم، ومرنة للتوسع، وأقل عرضة للأخطاء.