A solid description for the SOLID Principles

1 Introduction

The SOLID principles are a set of guidelines in object-oriented programming that promote code readability, scalability, and maintainability. The acronym SOLID stands for Single Responsibility Principle (SRP), Open/Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP), and Dependency Inversion Principle (DIP).

2 Single Responsibility Principle (SRP)

Concept: A class should have only one reason to change.

Example:

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height
```

In this example, the `Rectangle` class has a single responsibility: to calculate its area.

3 Open/Closed Principle (OCP)

Concept: Software entities should be open for extension but closed for modification.

Example:

```
class Shape:
    def area(self):
        pass

class Rectangle(Shape):
    # ... (same as before)
```

Here, we can add more shapes without modifying existing shape classes, thus adhering to the OCP.

4 Liskov Substitution Principle (LSP)

Concept: Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.

Example:

```
def calculate_area(shape):
    return shape.area()

rectangle = Rectangle(2, 3)
print(calculate_area(rectangle))  # Output: 6
```

5 Interface Segregation Principle (ISP)

Concept: Clients should not be forced to depend on interfaces they do not use.

Example:

```
class Shape:
    def area(self):
        pass
```

6 Dependency Inversion Principle (DIP)

Concept: High-level modules should not depend on low-level modules. Both should depend on abstractions.

Example:

```
def calculate_area(shape):
    return shape.area()
```