SOLID Principles: Good and Bad Examples with Code Snippets

This document contains good and bad examples of each SOLID principle, along with code snippets to demonstrate how to apply or violate the principles.

# 1. Single Responsibility Principle (SRP)

## Good Example:

In this good example, we separate the concerns of order processing and invoice printing into two distinct classes.

Code snippet:

class OrderProcessor:  
 def process\_order(self, order):  
 # Order processing logic here  
 pass  
  
class InvoicePrinter:  
 def print\_invoice(self, order):  
 # Invoice printing logic here  
 pass

## Bad Example:

In this bad example, a single class handles both order processing and invoice printing, violating SRP.

Code snippet:

class OrderProcessor:  
 def process\_order(self, order):  
 # Order processing logic here  
 pass  
  
 def print\_invoice(self, order):  
 # Invoice printing logic here  
 pass

# 2. Open/Closed Principle (OCP)

## Good Example:

In this good example, we extend the payment processing functionality using inheritance without modifying existing code.

Code snippet:

class PaymentProcessor:  
 def process\_payment(self, payment):  
 raise NotImplementedError("Subclasses should implement this method.")  
  
class CreditCardPayment(PaymentProcessor):  
 def process\_payment(self, payment):  
 # Credit card payment logic here  
 pass  
  
class PayPalPayment(PaymentProcessor):  
 def process\_payment(self, payment):  
 # PayPal payment logic here  
 pass

## Bad Example:

In this bad example, we modify the existing 'PaymentProcessor' class to add new payment types, violating OCP.

Code snippet:

class PaymentProcessor:  
 def process\_payment(self, payment, payment\_type):  
 if payment\_type == 'credit\_card':  
 # Credit card payment logic here  
 pass  
 elif payment\_type == 'paypal':  
 # PayPal payment logic here  
 pass

# 3. Liskov Substitution Principle (LSP)

## Good Example:

In this good example, subclasses extend the behavior of the base class without changing the expected behavior.

Code snippet:

class Bird:  
 def fly(self):  
 return "Flying"  
  
class Sparrow(Bird):  
 def fly(self):  
 return "Sparrow flying"  
  
class Penguin(Bird):  
 def fly(self):  
 return "Penguin can't fly"

## Bad Example:

In this bad example, the subclass 'Penguin' violates LSP by changing the expected behavior of 'fly' method.

Code snippet:

class Bird:  
 def fly(self):  
 return "Flying"  
  
class Penguin(Bird):  
 def fly(self):  
 # Penguin can't fly, so this breaks LSP  
 raise Exception("Penguin can't fly")

# 4. Interface Segregation Principle (ISP)

## Good Example:

In this good example, we separate the interface into smaller, more specific ones, so clients are not forced to implement methods they don't use.

Code snippet:

class Worker:  
 def work(self):  
 pass  
  
class Eater:  
 def eat(self):  
 pass  
  
class Robot(Worker):  
 def work(self):  
 pass  
  
class Human(Worker, Eater):  
 def work(self):  
 pass  
  
 def eat(self):  
 pass

## Bad Example:

In this bad example, a single interface forces clients to implement methods they don't need, violating ISP.

Code snippet:

class Worker:  
 def work(self):  
 pass  
  
 def eat(self):  
 pass  
  
class Robot(Worker):  
 def work(self):  
 pass  
 # Robot doesn't need to eat, but is forced to implement the eat method.

# 5. Dependency Inversion Principle (DIP)

## Good Example:

In this good example, high-level modules depend on abstractions instead of low-level modules.

Code snippet:

class DataStorage:  
 def save(self, data):  
 pass  
  
class DatabaseManager(DataStorage):  
 def save(self, data):  
 # Save to database  
 pass  
  
class UserManager:  
 def \_\_init\_\_(self, storage: DataStorage):  
 self.storage = storage  
  
 def save\_user(self, user):  
 self.storage.save(user)

## Bad Example:

In this bad example, high-level modules depend directly on low-level modules, violating DIP.

Code snippet:

class UserManager:  
 def \_\_init\_\_(self):  
 self.storage = DatabaseManager()  
  
 def save\_user(self, user):  
 self.storage.save(user)