Open/Closed Principle (OCP)

> Open for extension but closed for modification -- able to add new features without changing old code.

without OCP

```
class ConfigLoader:
    def load(self, path):
        with open(path, "r") as file:
            return yaml.safe_load(file)
```

It does not support other file extensions.

with OCP

```
class ConfigLoader:
    def __init__(self, parser: ConfigParser, path: str):
        self.parser = parser
        self.path = path

    def load(self):
        return self.parser.load(self.path)
```

An abstract interface:

```
from abc import ABC, abstractmethod

class ConfigParser(ABC):
    @abstractmethod
    def load(self, path: str) -> dict:
        pass
```

Implementing YAML loader:

```
import yaml
from config_parser import ConfigParser

class YamlConfigParser(ConfigParser):
    def load(self, path):
        with open(path, "r") as file:
            return yaml.safe_load(file)
```

Implementing JSON support:

```
import json
from config_parser import ConfigParser

class JsonConfigParser(ConfigParser):
    def load(self, path):
        with open(path, "r") as file:
            return json.load(file)
```

> If `@abstractmethod` unfamililar to new users, do not worry, An explanation is as follows:

In Python, abstract methods are defined using the `@abstractmethod` decorator from the abc (Abstract Base Classes) module. It is declared but not implemented in the base class. It only serves as a blueprint for child classes, forcing them to provide their own implementation.

In the code above, `ConfigParser` is an abstract base class. The function `def load(self, path)` is an abstract method since `@abstractmethod` decorator is used before the function decleration. Note that there is no implementation of this function, just a method signature.

This ensures that any subclass of `ConfigParser` must implement the `load()` method, or it will raise a TypeError at instantiation. As a result, subclasses `YamlConfigParser` and `JsonConfigParser` implemented the `load()` method and satisfies the requirement imposed by the abstract base class.