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)
```