

Liskov Substitution Principle (LSP)

Subtypes must be substitutable for their base types.

That is, if class B is a substitute of A, it should behave like A when used in place of it.

Already applied in the previous principle. Let us make it automatic.

```
class ConfigParserFactory:
    @staticmethod
    def get_parser(path):
        ext = os.path.splitext(path)[-1].lower()

        if ext in [".yaml", ".yml"]:
            return YamlConfigParser()
        elif ext == ".json":
            return JsonConfigParser()
        else:
            raise ValueError(f"Unsupported extension: {ext}")
```

class B is `YamlConfigParser` and class A is `JsonConfigParser`. Therefore, no matters which parser is going to use, the code behaves the same. The reason behind it is that both class A and B used abstract base class and fulfills the abstract method requirements, i.e., implementation of `load()` function.