# Single Responsibility Principle

Application represents simple eshop, in which a user can order specified number of available products. It is accessible via a web browser at localhost:8080

### Tasks

1. Look at the source files and identify SRP violations
2. What are the negative impacts of these violations?
3. How would you solve the SRP violations (at least theoretically)?

Solution

Every class in a computer program should have responsibility over a single part of that program's functionality, which it should encapsulate. Classes in the task violates this definition.

Negative impact of the violations are in maintainability and reusability.

Solved by project refactoring.

# Open/Closed Principle

Application allows persisting of Comments into various types of files

### Tasks

1. Imagine, you want to extend the app with persisting into JSON. Find all places in the code, which you would have to modify to implement the extension.
2. Refactor the project so that it follows OCP.

Resolved in the code

# Liskov Substitution Principle

Project contains multiple serializers (CVS, JSON, Compression), which implement interface SimpleSerializer. In the Main class is a showcase of (de)serialization of two different objects for different combinations of serializers.

### Tasks

1. Read the contract in SimpleSerializer. Based on the contract, identify the LSP violations in the individual Serializers.
2. HINT: Try to run the Main method.

Solution

One of the violations shows to us when trying to deserialize the CSV of the object “Masterpiece Mona Lisa”. Documentation of the SimpleSerializer interface tells that the array of arrays is not permissible. This binds to the line 29 in Main.class. Also, it passed with JSONSerializer because CSVSerializer implements SimpleSerializer while JSONSerializer does not. We expect both to behave same way.

Next violation shows to us in serializer JSON with added compression on object Friend. As it is told in the documentation of SimpleSerializer interface on line 15 behaviour is not defined if conditions are not satisfied. Here we can see problem with the condition on line 18 and 19 – serialized and deserialized strings differs. Problem shows with the conversion of special characters between utf8 and ascii where we can see that the CompressingSerializer has more strict entry conditions while the SimpleSerializer has not defined those enough.

These who problems repeat again with both objects using the serializer CSV with added compression.

# Interface Segregation Principle

### Tasks

1. Discussion – can you give any examples of good and bad interfaces?

Solution

It’s nicely explained on this website <https://proandroiddev.com/interface-segregation-principle-3ce39a326dd>

It’s contra productive to rewrite it in this file :)

# Dependency Inversion Principle

Application implements recommendation services, which recommend how to dress/where to go for lunch based on the weather forecast.

### Tasks

1. Find violations of DIP
2. Rewrite the code so that it adhers to the DIP
3. What benefits does the DIP implementation brings?
4. (Optional) Explain the difference between DIP, Dependency injection and Inversion of Control

Solution

Problem is that the code is meant to use specific OpenWeatherService and DarkSkyForecastService objects which can be changed by the time with different object. We have to create separate interfaces and refactor the code the way, that the hardcoded parts will be used through constructor.

Unable to fully rewrite the code so that it adhers to the DIP.

Benefit is that the DIP reduces coupling between different pieces of code. If we extract an interface which represents the logging, this interface should be much more stable for the future manipulation.

Inversion of control means that the program delegates control to someone else who will drive the flow. Inversion of control is a parent term while Dependency injection is a subset of the Inversion of control which is also a concept where the flow of the application is inverted

Source: Google