**Section2 SOLID Design principles:**

1. **Single Responsibility Principle (SRP)**

Keep the class as simple as possible. E.g. a class to store properties, method related to the properties. A separate class to manipulate the previous class like CRUD behaviors.

1. **Open-Closed Principle(OCP) + Sepcification**

Want the code open to extension but close to modification after it is being tested/delivered

Sepcification Pattern

Interface Sepcification<T>{ Boolean isSatisfied(T item);}

Interface Filter<T>{ Stream<T> filter(List<T> items, Specification<T> spec);}

ColorSpecification extends Sepcification

A general filter which extends Filter interface to filter a particular class.

To satisfy to condition, we make a new class AddSpecification<T> implements Specification<T>

1. **Liskov Substitution Principle(LSP) + Factory Pattern**

Should be able to substitute a base type for a subtype in any api calls without breaking things.

e.g. Rectangle <- square all have width and length, set length of square changes width of square which makes the area of square change too.

1. **Interface Segregation Principle(ISP)**

Should break an big interface to small ones, so that class can choose to inherit part or all the them so that we don’t have class inherit methods that are not implemented.

1. **Dependency Inversion Principle(DIP)**

High-level modules shouldn’t depend on low-level modules. Both should depends on abstractions.

Abstractions should not depend on details. Details should depend on abstractions.

**Things to do research on:**

**sequence processing (streams/Rx)**

**Stream Class Java 8**

Tutorial:

<https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>

Java 8 Stream Tutorial.pdf

Java Docs:

<https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html>

concurrency, dependency injection