Contents

[Day\_5 1](#_Toc71531168)

[Solid 1](#_Toc71531169)

[Discussion 1](#_Toc71531170)

[Regular Expression 1](#_Toc71531171)

[RegEx 1](#_Toc71531172)

[Unit Testing 2](#_Toc71531173)

[Unit Testing 2](#_Toc71531174)

[Parts of a unit test 2](#_Toc71531175)

[Test Driven Development (TOD) 2](#_Toc71531176)

Revature

# Day\_5

## Solid

**Create Flexible, Maintainable, and Modular code**

* Single Responsibility
  + Classes should be responsible for one thing, and one thing only.
  + Follows Encapsulation by putting individual functions into individual class.
* Open-Close
  + When code is updated, you should minimize refactoring.
  + If necessary, changes are not relevant to your code, it should not require any refactorization.
* Liskov Substitution Principle
  + Child types should follow the basic rules or data structure that the parent class features.
  + This allows for clean Covariance and Contravariance.
* Interface Segregation
  + Do not create a super Interface that does everything.
* Dependency Inversion
  + Do not tightly couple your classes together.
  + Similar to Open-Close Principle

## Discussion

Q: What is SOLID? Give a brief description

A: Single Responsibility: Cannot be a super class, break classes down into individual functions.

Open-Close: Objects should only have one purpose. Encapsulation is a tool to achieve Open-Close principle, just like Dependency Inversion can affect it as well.

Liskov Sub. Princp.: If a class inherits from another class, the idea or application of the parent type should be.

Interface Segregation: Each Interface should implement one general idea, and not handle any other ideas that fall outside of this general idea. Ex. IMenu’s general idea is to handle the Menu Screens, and shouldn’t handle any Data Manipulation, or Business Logics.

Dependency Inversion:

# Regular Expression

## RegEx

* Very Helpful in all around programing
* A Good way to standardize your inputs
* Primarily used in pattern matching

# Unit Testing

## Unit Testing

* Test small units of your code
  + This would be your methods
* Done in XCode

## Parts of a unit test

* Arrange
  + Any setup necessary to prepare for the behavior test
* Act
  + You do the thing you wanna test
    - Testing City Validator, run the setter for the City Validator

## Test Driven Development (TOD)

* Write test that fail
* Build code to test
* Initialize what your program will accept and what it wont accept