SOLID Principles objective education syntax markdown

This project get code from https://github.com/mikeknep/SOLID, For **education** only. I recommend you to read this manual inside Github website for better view.

Table of contents

Description

Dependency Inversion Principle

> In high-level modules should not depend on low-level modules.

As you see in bad design, Inside WeatherTracker class have Phone and Emailer class, which bad design because if I try to create new receiver like SMS class, I need to modify code 2 place (in SMS.java and WeatherTracker.java). To fix this issue, in good code it use Notifier which contain action method if it receive message. Now instead of have client code (Phone and Emailer) inside server code (WeatherTracker), we use Notifier pass as parameter to notify. So that Notifier became to interface that allow another class to implement it.

Interface Segregation Principle

> Clients should not be forced to depend on any interfaces they do not use.

For example of this is Bird class, The bad design will auto appear since no all of bird can fly so like in the bad code design Eagle and Penguin that in Penguin fly method throw UnsupportedOperationException. In new design (the good one) it separate all of action as new interface.

For example:

- FlyingCreature
- FeatheredCreature
- SwimmingCreature

Liskov Substitution Principle

> Functions that references to base classes must be able to use objects of derived classes without knowing it.

As you see method upgrade in UnitUpgrader, there have a code to check class of parameter is it Studio or not. That is should not be. To fix this issue, it try to separate class that not have same role so the adapter class (like BedroomAdder) will don't have to check no matter it is.

Open Closed Principle

> Software program should be open for extension, but closed for modification.

In the bad example, the issue will appear when developer want to add new way to greet people developer need to modify Greeter, and later on Greeter class will became fatty. So to fix it reference to description of open closed principle, that is extension. Personality is the interface of greeting type, and that each of class refer to each of greeting type. so if need to create new type of greet, it just create new class that implement

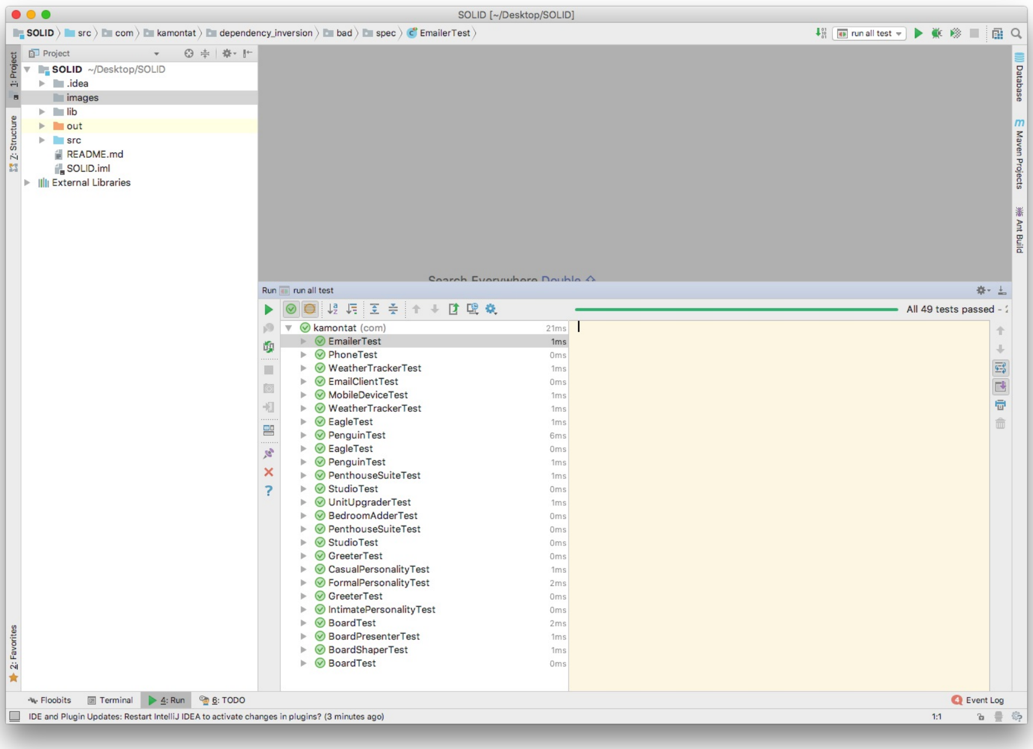
Personality and pass to constructor method in Greeter class.

Single Responsibility Principle

> Every class should have a single responsibility. There should never be more than one reason for a class to change.

For instance bad example, Board class have multiple responsibility such as getting board's rows, print the board to screen. so the **Single Responsibility Principle** tells us that this class is actually handling far too many. Consider the Board class in good example, this response only the values of its spots. The BoardShaper and BoardPresenter class are similarly focused on specific tasks. They also pass only needed attributes; for example, BoardShaper object are initialized with only size since they don't need to know whole board.

Images



```
All 49 tests passed
kamontat (com)              21ms
  EmailerTest                1ms
  PhoneTest                  0ms
  WeatherTrackerTest         1ms
  EmailClientTest            0ms
  MobileDeviceTest           1ms
  WeatherTrackerTest         1ms
  EagleTest                  1ms
  PenguinTest                6ms
  EagleTest                  0ms
  PenguinTest                1ms
  PenthouseSuiteTest         1ms
  StudioTest                 0ms
  UnitUpgraderTest           1ms
  BedroomAdderTest           0ms
  PenthouseSuiteTest         0ms
  StudioTest                 0ms
  GreeterTest                0ms
  CasualPersonalityTest      1ms
  FormalPersonalityTest      2ms
  GreeterTest                0ms
  IntimatePersonalityTest    0ms
  BoardTest                  2ms
  BoardPresenterTest         1ms
  BoardShaperTest            1ms
  BoardTest                  0ms
```

IDE and Plugin Updates: Restart IntelliJ IDEA to activate changes in plugins? (3 minutes ago)

Creator