CMPE 202 - Personal Project - UML Parser - 2017

Applying Kanban process model to create a parser that converts Java source code into UML Class diagram.

For Extra Credit :

* Collaborating with the CMPE281 team member to deploy this project in cloud.
* Generate sequence diagram from the java code provided.

To Run: The parser should be executable on the command line with the format -

**“umlparser <source folder> <output file name>”**

where, <source folder> is a folder name where all the .java source files will be and

<output file name> is the name of the output image file that the program will generate

Requirements: 1. Java Environment,

2.Graphviz (to generate class diagrams)

Tools Used:

* Graphviz – Open source software that provides graphical visualization. Takes text descriptions to generate the graph. Computation is generally done by an external program
* Java Parser – Java Parser is a light weight library that can be used parse, analyse, transform and generate java code. It parses through the java code to generate an AST(abstract Syntax Tree) that makes it easy to process. The parser has no dependencies at all and hence the ABT generation is fast.
* Plantuml – plantuml employs graphviz/DOT to figure out node positioning for the UML diagrams that it generates. To use it, it must be installed in the system. Moreover, Graphviz/DOT originally written in C. But plantuml needs only the DOT algorithm, since C syntax is close to Java syntax, conversion of just that would suffice. Drawing code and parsing of DOT is not necessary as it is done in java by plantuml.

Project Tracking:

Github – A repository hosting service with many of its own features. Provides a graphical user interface, access control and several collaboration features. It simplifies the task of the developers by enabling them to fork, pull request , merge.

WaffleIO – Is a free project management tool that enables the developers to keep tracl of their prohect. Right from the start it helps to plan every step along the way. It takes cues from the commits made into github to update the status on the waffle board.

Problems encountered: Initially was working on the project in cloud9. Cloud9 is an online integrated development environment that enables its users to start coding immediately without the need to set up the environment. It supports several languages like C, C++, PHP, Ruby, Perl, Python, JavaScript with Node.js, and Go. But the prolonged sessions were hard to handle so switched to local environment.

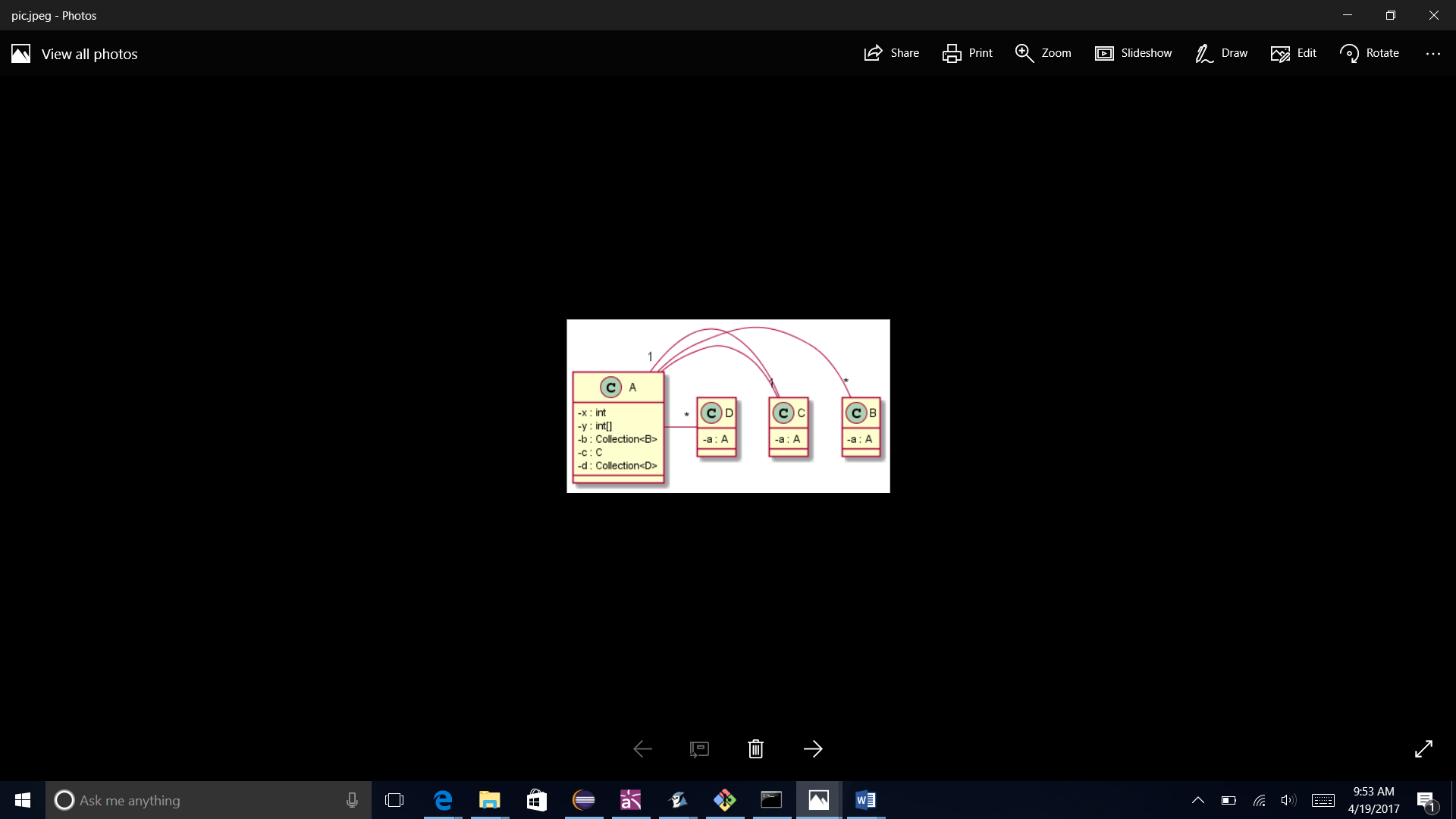
**Test Cases as provided:**

Test Case :1

Run in cmd giving :

java -jar ProjectUml "C:\\Users\T V Divyaa\\Desktop\\FinalTestCases\\test1\\” pic

the file path- location where the test cases are stored.



Op.txt content:

@startuml

skinparam classAttributeIconSize 0

A : - x : int

A : - y : int[]

A - "\*" B

A : - b : Collection<B>

A - "1" C

A : - c : C

A - "\*" D

A : - d : Collection<D>

class B

B : - a : A

class C

C - "1" A

C : - a : A

class D

D : - a : A

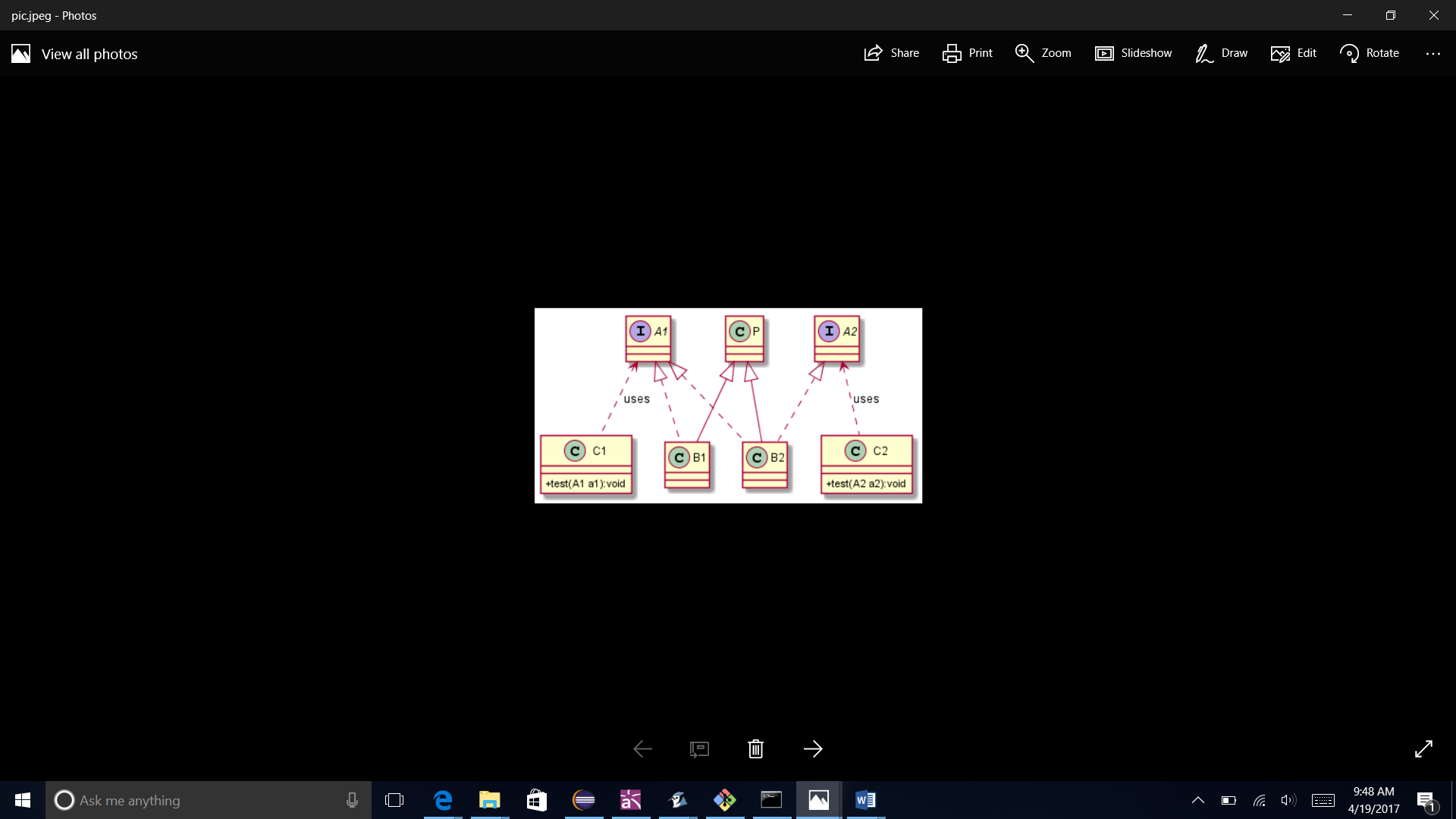
@enduml

Test Case :2

Run in cmd giving :

java -jar ProjectUml "C:\\Users\T V Divyaa\\Desktop\\FinalTestCases\\test2\\” pic

the file path- location where the test cases are stored.



Op.txt content:

@startuml

skinparam classAttributeIconSize 0

interface A1

interface A2

class B1

P <|-- B1

A1 <|.. B1

class B2

P <|-- B2

A1 <|.. B2

A2 <|.. B2

class C1

A1<.. C1:uses

C1 : +test(A1 a1):void

class C2

A2<.. C2:uses

C2 : +test(A2 a2):void

class P

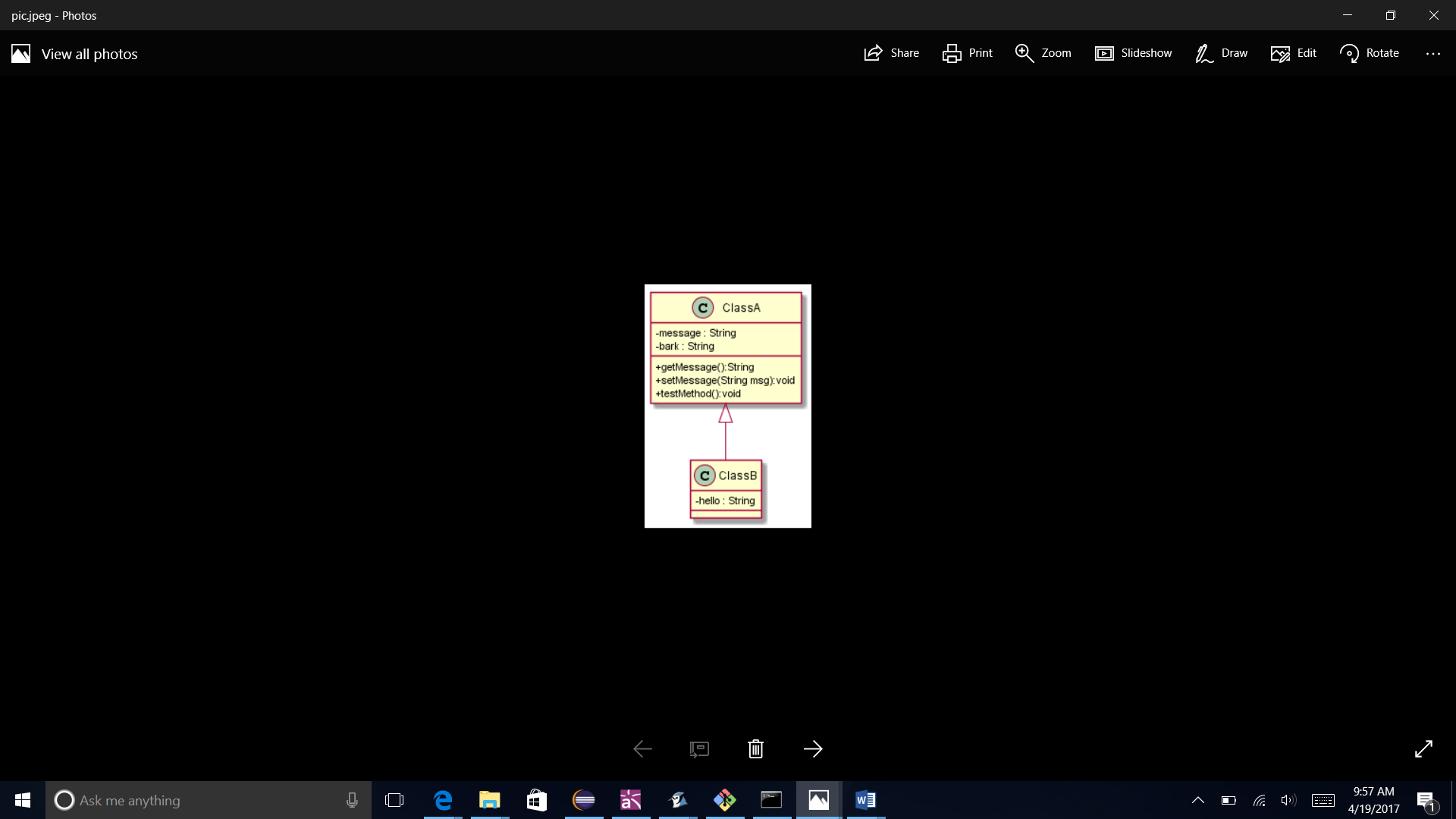
@enduml

Test Case 3:

Run in cmd giving :

java -jar ProjectUml "C:\\Users\T V Divyaa\\Desktop\\FinalTestCases\\test3\\” pic

the file path- location where the test cases are stored.



Op.txt Content

@startuml

skinparam classAttributeIconSize 0

ClassA : - message : String

ClassA : - bark : String

ClassA : +getMessage():String

ClassA : +setMessage(String msg):void

ClassA : +testMethod():void

ClassA <|-- ClassB

ClassB : - hello : String

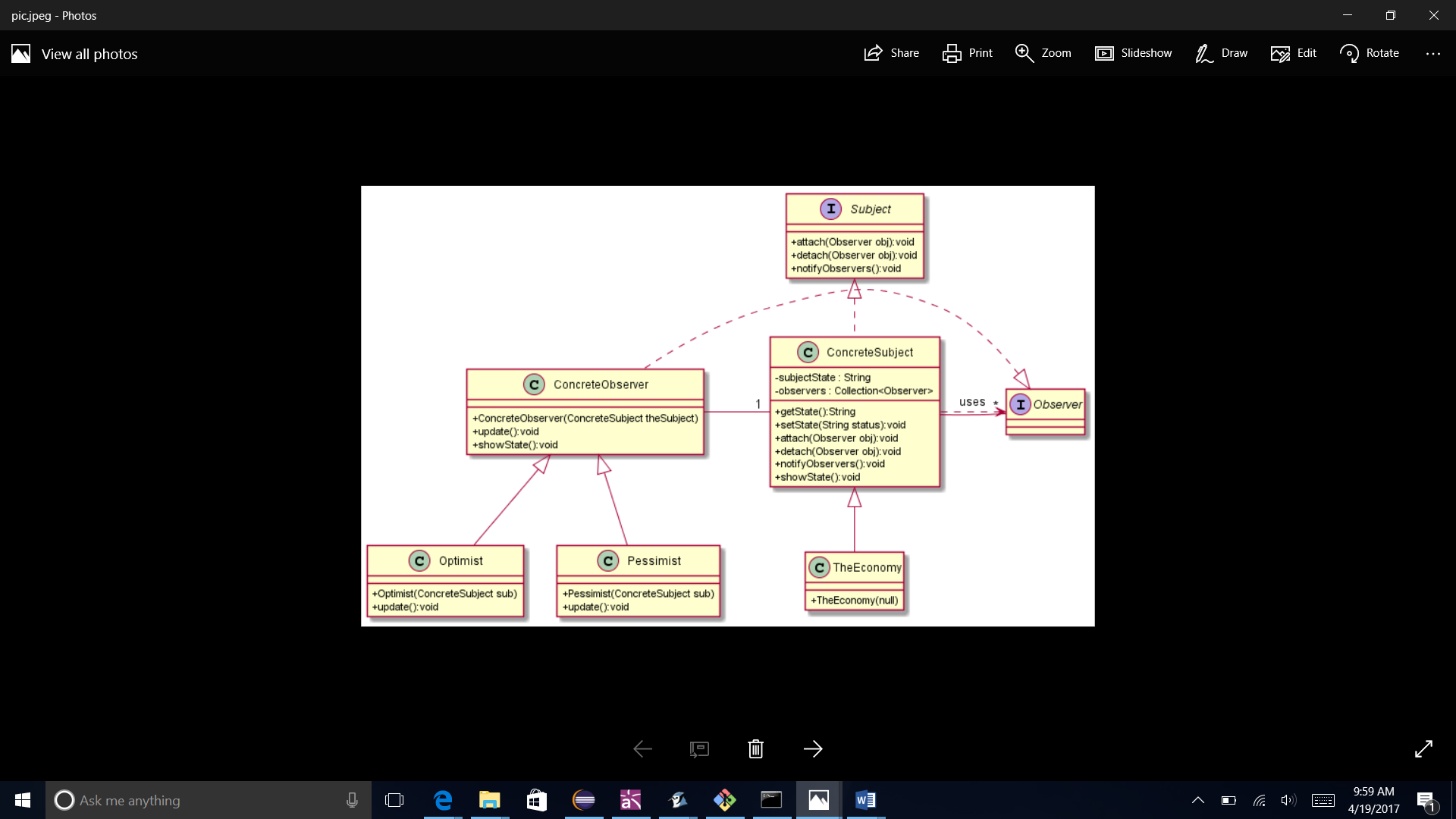
@enduml

Test Case 4:

Run in cmd giving :

java -jar ProjectUml "C:\\Users\T V Divyaa\\Desktop\\FinalTestCases\\test4\\” pic

the file path- location where the test cases are stored.



Op.txt content:

@startuml

skinparam classAttributeIconSize 0

class ConcreteObserver

ConcreteObserver : +ConcreteObserver(ConcreteSubject theSubject)

Observer <|.. ConcreteObserver

ConcreteObserver - "1" ConcreteSubject

ConcreteObserver : +update():void

ConcreteObserver : +showState():void

Subject <|.. ConcreteSubject

ConcreteSubject : - subjectState : String

ConcreteSubject - "\*" Observer

ConcreteSubject : - observers : Collection<Observer>

ConcreteSubject : +getState():String

ConcreteSubject : +setState(String status):void

Observer<.. ConcreteSubject:uses

ConcreteSubject : +attach(Observer obj):void

ConcreteSubject : +detach(Observer obj):void

ConcreteSubject : +notifyObservers():void

ConcreteSubject : +showState():void

interface Observer

class Optimist

ConcreteObserver <|-- Optimist

Optimist : +Optimist(ConcreteSubject sub)

Optimist : +update():void

class Pessimist

ConcreteObserver <|-- Pessimist

Pessimist : +Pessimist(ConcreteSubject sub)

Pessimist : +update():void

interface Subject

Subject : +attach(Observer obj):void

Subject : +detach(Observer obj):void

Subject : +notifyObservers():void

class TheEconomy

ConcreteSubject <|-- TheEconomy

TheEconomy : +TheEconomy(null)

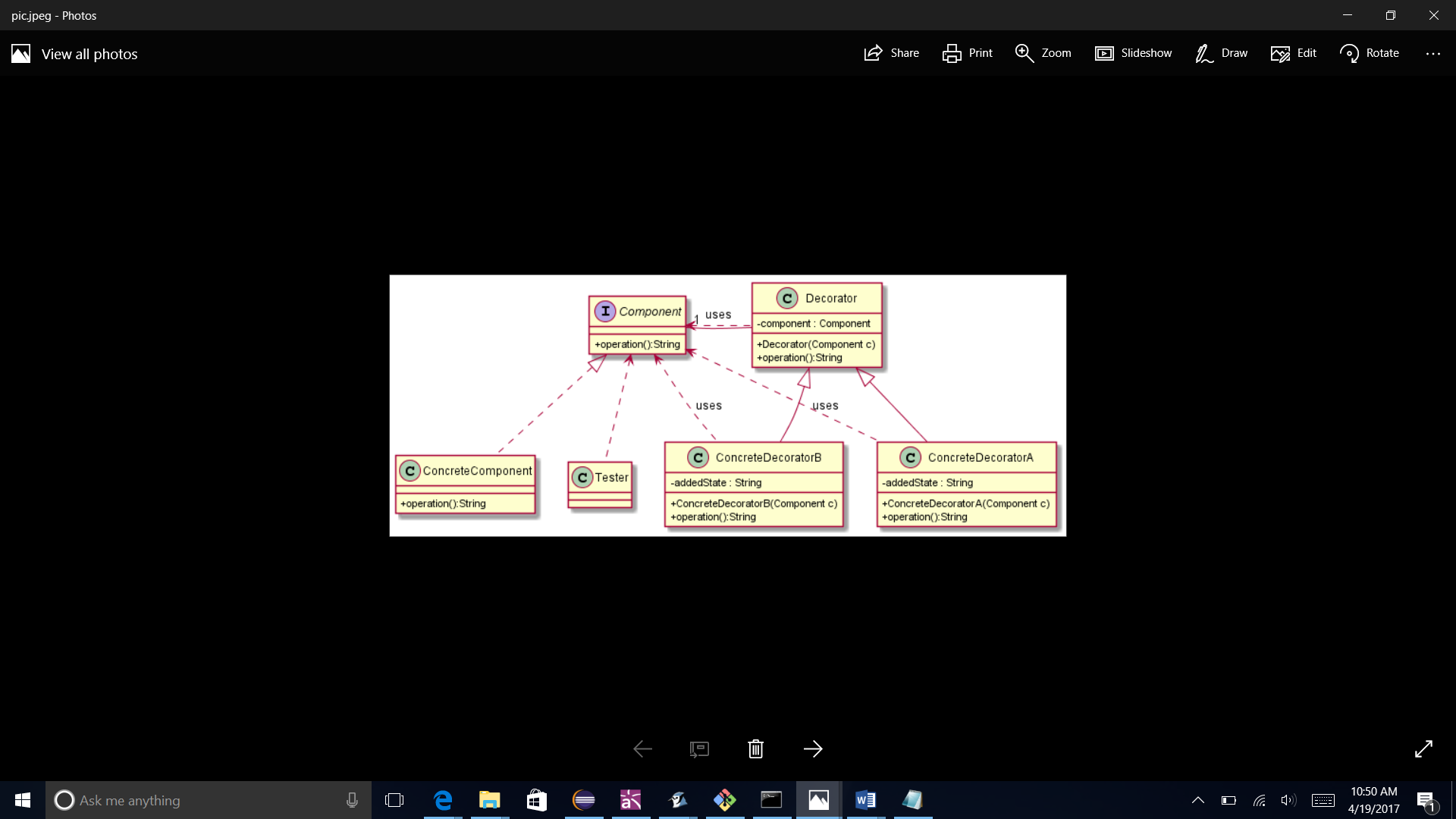
@enduml

Test Case 5:

Run in cmd giving :

java -jar ProjectUml "C:\\Users\T V Divyaa\\Desktop\\FinalTestCases\\test5\\” pic

the file path- location where the test cases are stored.



Op.txt content:

@startuml

skinparam classAttributeIconSize 0

interface Component

Component : +operation():String

class ConcreteComponent

Component <|.. ConcreteComponent

ConcreteComponent : +operation():String

class ConcreteDecoratorA

Decorator <|-- ConcreteDecoratorA

Component<.. ConcreteDecoratorA:uses

ConcreteDecoratorA : +ConcreteDecoratorA(Component c)

ConcreteDecoratorA : - addedState : String

ConcreteDecoratorA : +operation():String

class ConcreteDecoratorB

Decorator <|-- ConcreteDecoratorB

Component<.. ConcreteDecoratorB:uses

ConcreteDecoratorB : +ConcreteDecoratorB(Component c)

ConcreteDecoratorB : - addedState : String

ConcreteDecoratorB : +operation():String

class Decorator

Component<.. Decorator:uses

Decorator : +Decorator(Component c)

Decorator - "1" Component

Decorator : - component : Component

Decorator : +operation():String

class Tester

Component<.. Tester

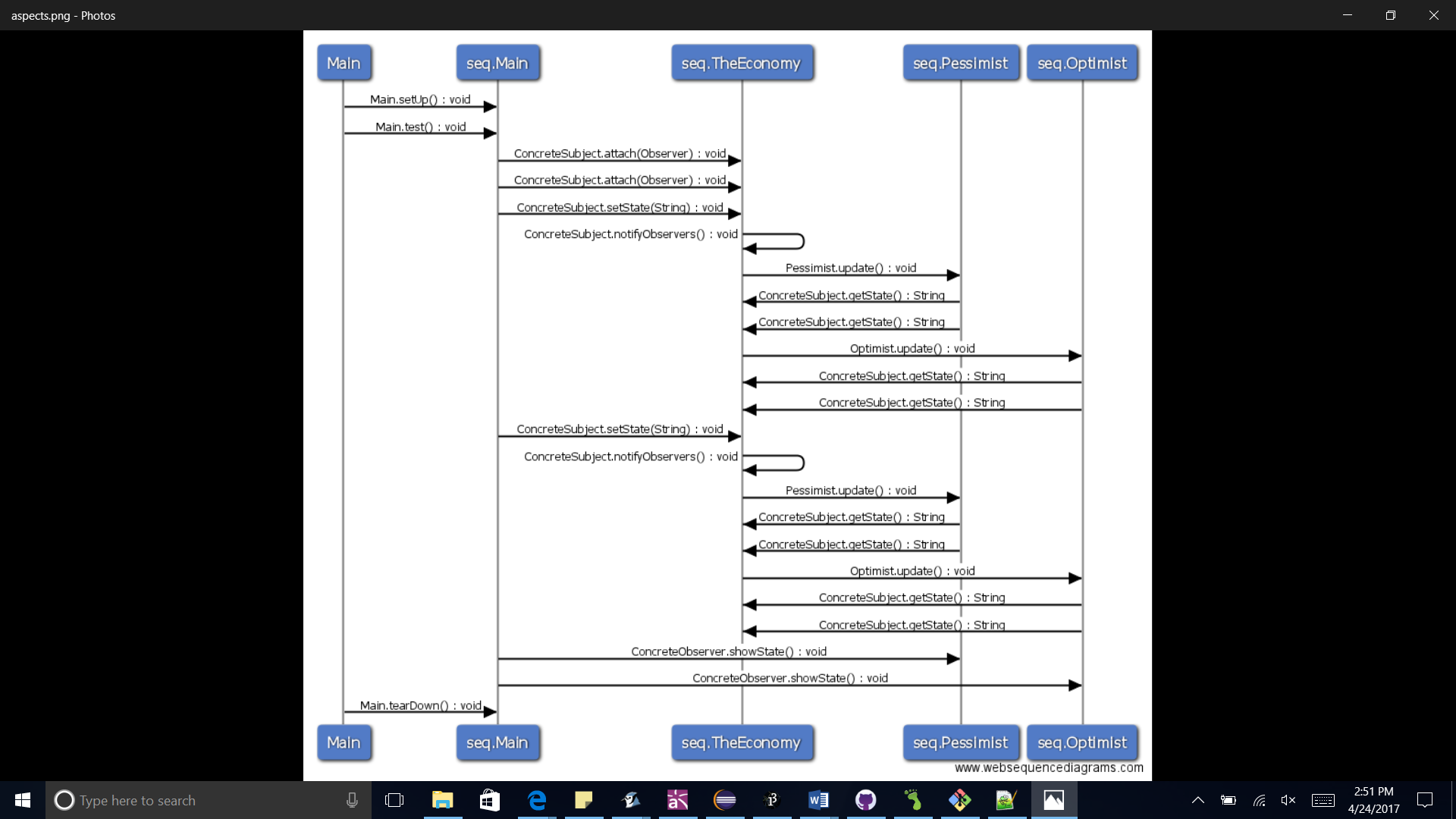
@enduml

**Extra Credit Attempt**

Extend the UML Parser to perform **dynamic analysis** of the Java Source code to generate a UML Sequence Diagram.  The UML Sequence Diagram must be generated by executing the Java Source Code. The Test Code will be executed from a Static Main class method in a ***Main.java*** class. (**10 points**)

Java Classes provided in Test Case:

* ConcreteObserver
* ConcreteSubject
* Main
* <<interface>>Observer
* Optimist
* Pessimist
* <<interface>>Subject
* Economy



Op.txt content:

Main ->seq.Main: Main.setUp() : void

Main ->seq.Main: Main.test() : void

seq.Main ->seq.TheEconomy: ConcreteSubject.attach(Observer) : void

seq.Main ->seq.TheEconomy: ConcreteSubject.attach(Observer) : void

seq.Main ->seq.TheEconomy: ConcreteSubject.setState(String) : void

seq.TheEconomy ->seq.TheEconomy: ConcreteSubject.notifyObservers() : void

seq.TheEconomy ->seq.Pessimist: Pessimist.update() : void

seq.Pessimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.Pessimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.TheEconomy ->seq.Optimist: Optimist.update() : void

seq.Optimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.Optimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.Main ->seq.TheEconomy: ConcreteSubject.setState(String) : void

seq.TheEconomy ->seq.TheEconomy: ConcreteSubject.notifyObservers() : void

seq.TheEconomy ->seq.Pessimist: Pessimist.update() : void

seq.Pessimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.Pessimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.TheEconomy ->seq.Optimist: Optimist.update() : void

seq.Optimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.Optimist ->seq.TheEconomy: ConcreteSubject.getState() : String

seq.Main ->seq.Pessimist: ConcreteObserver.showState() : void

seq.Main ->seq.Optimist: ConcreteObserver.showState() : void

Main ->seq.Main: Main.tearDown() : void

References:

* <http://javaparser.org/>
* <https://en.wikipedia.org/wiki/Cloud9_IDE>
* <http://plantuml.com/>
* <https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/>
* <https://www.crunchbase.com/organization/waffle-io#/entity>